A Survey on Training and Evaluation of Word Embeddings

François Torregrossa · Robin Allesiardo · Vincent Claveau · Nihel Kooli · Guillaume Gravier

This is a post-peer-review, pre-copyedit version of an article published in International Journal of Data Science and Analytics. The final authenticated version is available online at: https://doi.org/10.1007/s41060-021-00242-8

Abstract Word Embeddings have proven to be effective for many Natural Language Processing tasks by providing word representations integrating prior knowledge. In this article, we focus on the algorithms and models used to compute those representations and on their methods of evaluation. Many new techniques were developed in a short amount of time and there is no unified terminology to emphasise strengths and weaknesses of those methods. Based on the state of the art, we propose a thorough terminology to help with the classification of these various models and their evaluations. We also provide comparisons of those algorithms and methods, highlighting open problems and research paths, as well as a compilation of popular evaluation metrics and datasets. This survey gives: 1) an exhaustive description and terminology of currently investigated word embeddings, 2) a clear segmentation of evaluation methods and their associated datasets, and 3) high-level properties to indicate pros and cons of each solution.

Keywords Word Embeddings · Word Embedding Evaluation · Survey · Contextualised Embeddings · Non-Euclidean Embeddings

1 Introduction

Words are distinct elements of language carrying meaning. Characters on their own are not sufficient to retrieve the meaning of the words they compose. For instance, *restaurant* and *food* are both related, but it is impossible to quantify this relation only using the characters of these strings. Word embeddings are models for words answering to the particular use-case of providing meaningful representations for words. In other words, word embeddings are projections of words in a continuous space preserving the semantic relationships between them. Words are mapped to vectors sharing semantic properties through geometrical relations.

The strength of word embeddings relies on their ability to represent natural language with geometric relations. They enable efficient end-to-end architectures by transposing a discrete world representation into a continuous space. That is why they are widely used in Natural Language Processing (NLP) problems: they are easy to plug in Deep Learning architectures. Sentiment analysis, Named Entity Recognition (Bakarov, 2018; Schnabel et al., 2015; Pennington et al., 2014) and many other tasks outperformed their standard counterparts with these models.

We are mainly interested in word embeddings learnt over a training corpus *«from scratch»* (Collobert et al., 2011). This family of representations is trying to compile a plain text dataset into a continuous vector representation without any expert knowledge. All of them rely on a simple statement of Harris (1954) assuming that words appearing in similar

F. Torregrossa
Solocal/IRISA, 35000 Rennes
E-mail: ftorregrossa@solocal.com,francois.torregrossa@irisa.fr

R. Allesiardo
Solocal, Rennes, France
the author has moved to another location

V. Claveau
IRISA, CNRS
E-mail: vincent.claveau@irisa.fr

N. Kooli
Solocal, Rennes, France
the author has moved to another location

G. Gravier
IRISA, CNRS
E-mail: guig@irisa.fr

context must have similar meanings. Therefore, in this work we assume this *«distributional hypothesis»* claiming that the sense of a word highly depends on the words surrounding it. Other word embedding techniques refine this assumption and use a language model to build contextualised word representations, such as BERT (Devlin et al., 2019). Finally, a very different technique is to build vectors using knowledge bases or other sources of expert knowledge, an example being the TransE approach proposed by Bordes et al. (2013). We do not consider them on their own, but we will expose embedding techniques using the *«distributional hypothesis»* in combination with others, as done by Sun et al. (2018).

Different corpora, optimisation processes or basic geometries yield different sets of properties in the resulting projection. In this work, we are trying to emphasise these properties in order to provide a detailed analysis of these various embedding methods. Along with this analysis we also detail evaluation procedures.

The remainder of this paper is organised as follows. In Section 2, we explain elements of the underlying theory of embeddings. In Section 3, we discuss the different embedding evaluation metrics and methods. In Section 4, we present high-level features that characterise embeddings. In the final section, we conclude on suggestions for future research in the area.

2 Embeddings: from discrete variables to continuous semantic space

Word embeddings, in their original formulation, are *Lookup Tables* mapping words to vectors of real numbers (Collobert et al., 2011). They are functions giving continuous vector representations in $\mathbb{R}^D$ for elements of a set V (e.g. a set of words or tokens). In the following, $v_w^e \in \mathbb{R}^D$ is the representation of an element $w \in V$ in an embedding e. Formally a word embedding e can be represented as:

$$\begin{aligned} e: \quad V &\to \quad \mathbb{R}^D \\ w &\mapsto \quad e(w) = v_w^e. \end{aligned} \tag{1}$$

In order to estimate the word embedding function e, an optimisation process is performed on a large sample of language data on an arbitrary task. During this optimisation process, word vectors are coalesced given a criterion in charge of, for instance, extracting the similarity between words from a corpus. At the end, the embedding has accumulated information from the corpus for each word vector leading to noticeable geometric relationships. An example is shown on Figure 1: countries and their main cities are separated by distance and a direction almost identical for every pair. Automatic systems benefit from these relations since they explore numerical features and hyperplanes.

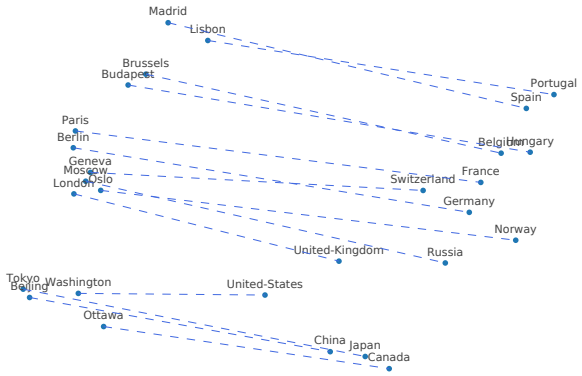


Fig. 1: Main cities and countries in a FastText (Bojanowski et al., 2017) embedding of dimension 300, projected onto a 2d plane using PCA. Dashed lines join each country with its main city.

In the following we separate static from contextualized embeddings:

- **Static word embeddings** use the *«distributional hypothesis»* to learn global and constant vectors. In other words, they represent a word by a unique vector condensing every local usage of the word in the training corpus. We distinguish Euclidean embeddings from others.
  - Euclidean word embeddings use the Euclidean geometry as mathematical support to embed vectors. Yet, due to intrinsic properties of this geometry (Sun et al., 2015), it may be difficult to embed asymmetric information. In fact, these kinds of embeddings (Pennington et al., 2014; Mikolov et al., 2013b; Bojanowski et al., 2017) preferably incorporate symmetric knowledge such as semantic synonymy or analogy.
  - Non-Euclidean geometries provide a solution to embed asymmetric relations such as entailments or logical thinking.
- **Contextualised word embeddings** are models providing variable vectors. In this work, we only review contextualised word embeddings using a language model to compute adaptative word representations. In this particular case, word representations are multiple and are directly computed from their context. The context of a word is usually composed by the words surrounding it. For contextualised embeddings, we only consider the Euclidean space as non-Euclidean contextualised word embeddings have not yet been investigated.

On Figure 2, a graph gives the main classes of word embedding techniques explained by basic properties of those.

2.1 Static Word Embeddings

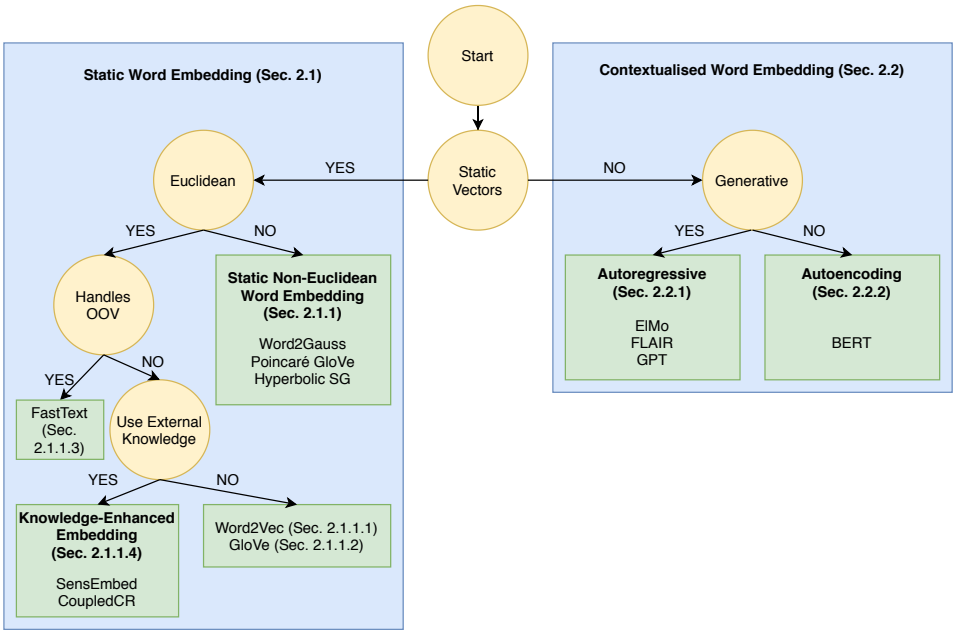


Fig. 2: Relations between word embeddings based on some basic properties.

Let a static word embedding be a word embedding where each word representation does not vary among contexts. For instance, the word *«bank»* in *«I am going to the bank»* and *«I want to open a bank account»* has two different meanings: a location or a financial organisation. However, a static word embedding gives the same word representation in both sentences without considering the context. Thus, the term *«static»* refers to this particularity: a word vector is always the same and can be statically pre-computed and stored in a matrix.

Static word embeddings are commonly represented with a matrix $M = (m_{ij}) \in \mathbb{R}^{|V| \times D}$. The i-th row of M is then a vector of $\mathbb{R}^D$ representing the i-th word of the vocabulary V. In other words, $v^M_{w_i} = (m_{ij})_{j \in [\![1,D]\!]}$, with w_i being the i-th word of V.

2.1.1 Euclidean Static Word Embeddings

Euclidean Static Word Embeddings are using the Euclidean geometry to build word representations. For instance, both Word2Vec Skip-Gram (SG) or Continuous Bag of Word (CBoW) proposed by Mikolov et al. (2013b) and FastText SG or CBoW from Bojanowski et al. (2017) belong to this class of models.

2.1.1.1 Word2Vec

Mikolov et al. (2013b) proposed a static word embedding algorithm named Word2Vec producing static word embeddings. It incorporates paradigmatic and syntagmatic relations between words in the vocabulary into word vectors, using two different word embedding models: SG or CBoW. *Paradigmatic* refers to individual concepts represented by the different words composing the vocabulary, while *syntagmatic* corresponds to the meaning inferred by the structure or syntax of language. Typically, idiomatic expressions are syntagmatic elements since their sense is derived from the whole sentence and not from the individual words in it. The Vocabulary V is composed of $|V|$ words extracted from a set of sentences in a corpus T (see Figure 3). In the end, an embedding vector is associated with each word.

This process relies on the *«distributional hypothesis»* from Harris (1954), claiming that word sharing similar context has similar meanings. Following Pennington et al. (2014) and Mikolov et al. (2013b), the context of a word can be defined by the k words before and the k words after this word in a sentence. k is called the window size and bounds the $2k$ words surrounding a target word (see Figure 3). Let C_{w_1}

$$T = \ldots \; w_{i-(k+1)} \; \boxed{w_{i-k} \;\; w_{i-(k-1)} \;\; \ldots \;\; w_{i-1} \;\; \boxed{w_i} \;\; w_{i+1} \;\; \ldots \;\; w_{i+(k-1)} \;\; w_{i+k}} \; w_{i+(k+1)} \; \ldots$$

Fig. 3: Window of size k around a target word w_i in a corpus T. The context words are then $\{w_{i-k}, w_{i-(k+1)}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+(k-1)}, w_{i+k}\}$.

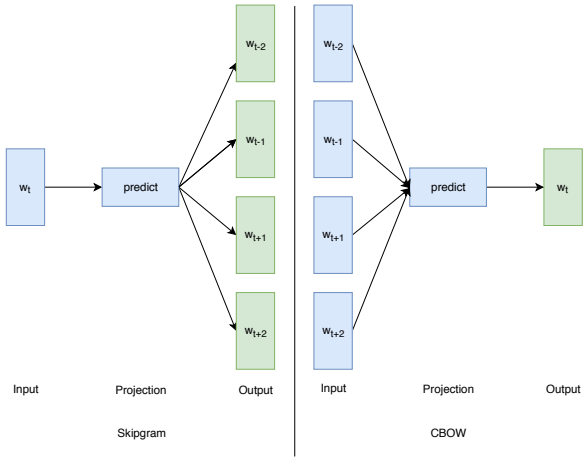


Fig. 4: The Skip-Gram (SG) and the Continuous Bag-of-Words (CBOW) models. On the left, the Skip-Gram model which predicts context words in the window given a target words. On the right, the CBOW model which predicts a target word given the context words of the window (adapted from Mikolov et al. (2013a)).

(resp. C_{w_2}) be the set of words appearing in the context of a word w_1 (resp. w_2). The *«distributional hypothesis»* states that the semantic proximity between w_1 and w_2 is positively correlated to $|C_{w_1} \cap C_{w_2}|$.

Two word embedding models were proposed to proceed the Word2Vec algorithm: the Skip-Gram (SG) and the Continuous Bag-of-Words (CBOW), presented on Figure 4.

Word2Vec uses two embeddings: a target word embedding (or *input matrix*) $W \in \mathbb{R}^{|V| \times D}$ and a context word embedding (or *output matrix*) $C \in \mathbb{R}^{|V| \times D}$. The word embedding C serves only for training and is mostly discarded after this process. One could argue that we could use a single matrix instead of two. However, context and target matrices play different roles. The context matrix C embed w.r.t. target words while the target matrix W embed w.r.t. context words. The aim of these two matrices is to reflect the difference between being a context and target word.

The Word2Vec Skip-Gram model tries to predict context words given a target word w_t. Therefore, for all w_t in the training corpus T, it maximises $p(w_c \,|\, w_t, T, W, C)$ (w_c, w_t respectively being a context and a target word), i.e. the probability to observe w_c in the window of w_t. Increasing this probability is equivalent to integrating significant information on w_c in the representation of w_t. Formally, from Mikolov et al. (2013a), the criterion to maximise is:

$$\frac{1}{|T|} \sum_{t=1}^{|T|} \sum_{\substack{c=t-k \\ c \neq t}}^{t+k} \log p(w_c \,|\, w_t, T, W, C), \tag{2}$$

where

$$p(w_c \,|\, w_t, T, W, C) = \frac{e^{\langle v^C_{w_c}, v^W_{w_t} \rangle}}{\sum_{w \in V} e^{\langle v^C_w, v^W_{w_t} \rangle}}. \tag{3}$$

The term $\sum_{w \in V} e^{\langle v^C_w, v^W_{w_t} \rangle}$ is not tractable in practice because the vocabulary contains millions of elements (Mikolov et al., 2013b). Actually, at each optimisation step it requires the computation of millions of dot products in a high dimensional space, which is not feasible in reasonable time. An alternative is to use Negative Sampling (NS) to turn the Skip-Gram objective into a computable one (Mikolov et al., 2013b). So far, we described Skip-Gram as a model trying to predict a context word given a target word. The idea of NS is to add random context words and detect them. Therefore, instead of predicting a context word given a target word, it classifies whether a context-target pair belongs to the corpus or was generated. In the end, it creates a representation where real pairs are geometrically close and noisy ones are scattered. Real pairs are called *positives* while noisy ones *negatives*, hence the name of the trick. Mathematically, NS maximises the dot product among positive pairs and minimises the dot product for negative pairs and is formulated as:

$$\frac{1}{|T|} \sum_{t=1}^{|T|} \sum_{\substack{c=t-k \\ c \neq t}}^{t+k} \Bigg[\log \sigma(\langle v^C_{w_c}, v^W_{w_t} \rangle) + \mathbb{E}_{w_{\text{neg}} \sim P_T(w)} \left(\log \sigma(-\langle v^C_{w_{\text{neg}}}, v^W_{w_t} \rangle) \right) \Bigg], \tag{4}$$

where $\sigma(x) = 1 + e^{-x}$ and $P_T(w) = \frac{|w|}{|T|}$.

The Continuous Bag-of-Words (Mikolov et al., 2013b) version of Word2Vec is similar to the Skip-Gram version but instead of predicting context words given a target word, it does the opposite: predicting a target word given the context words. Thus, the task is simpler since a larger set of words is known to infer the target word. In a sense, it might gather less semantic and more syntactic features, as shown by the experiment of Mikolov et al. (2013a). Yet for some languages, the CBOW version of Word2Vec leads to higher results (Grave et al., 2018). Therefore, choosing the version is still a parameter to optimise.

Depending on the window size, different aspects of language are represented. For instance, a short window is best suited for syntagmatic representations whereas a long window catches paradigmatic notions (Pennington et al., 2014). Levy and Goldberg (2014) have proven that Word2Vec relies on the fact that it is equivalent to factorising the pointwise mutual information matrix, i.e. the amount of shared information between a context and a target word.

2.1.1.2 GloVe

The Glove method (Pennington et al., 2014) reduces also this co-occurrences matrix in order to get vector representations. This procedure compiles the whole corpus into a large sparse matrix, each coordinate being the co-occurrences count between two words. Two words co-occur if they are in the context window of each other. This approach has two main advantages. First, the process is divided into two steps being the compression of the corpus into a sparse matrix followed by its factorisation by word vectors. The compression step is convenient since it has to be carried out once and produce a matrix summarising global information for each word in the corpus. Therefore, the optimisation step will benefit from global knowledge. The second point relies on the context window extracting local information on words. Consequently, GloVe cumulates both global and local information into its vectors leading to effective semantic representations, as proven by their experiments. In comparison, Word2Vec embeddings perform slightly less. An explanation is that Word2Vec focuses on local information and embed less easily global information.

Formally, let X_{ij} be the number of times word j appears in the context of word i, as done by Pennington et al. (2014). With these counters, authors defined

$$P_{ij} = \frac{X_{ij}}{\sum_k X_{ik}} \tag{5}$$

as the probability that the word j appears in the context of word i. This probability helps at quantifying the relevance of a word with regard to others. For instance, given three words i, j and k, the quantity

$$Q = \frac{P_{ik}}{P_{jk}} \tag{6}$$

acts as a discriminative feature:

- $Q \gg 1$ means that word k is more related to i than j,
- $Q \ll 1$ means that word k is less related to i than j,
- $Q \simeq 1$ means that word k is equally related to i and j.

The idea behind GloVe is to embed word probabilities into word vectors $v^W_{w_i}$, $v^W_{w_j}$ and $v^C_{w_k}$ all in $\mathbb{R}^D$ by trying to approximate Q using those vectors leveraging a function F, formally:

$$F(v^W_{w_i}, v^W_{w_j}, v^C_{w_k}) = Q = \frac{P_{ik}}{P_{jk}}, \tag{7}$$

where F must satisfy several criteria. First, F has to treat word vectors linearly without ambiguity over dimensions, thus, F is written $F((v^W_{w_i} - v^W_{w_j})^T v^C_{w_k})$. This property preserves the cosine similarity between vectors since dimensions are treated consistently. The second property is the symmetry between context word and target word which is artificial and implies that F must be able to exchange both roles. Therefore, authors (Pennington et al., 2014) added the constraint

$$F((v^W_{w_i} - v^W_{w_j})^T v^C_{w_k}) = \frac{F((v^W_{w_i})^T v^C_{w_k})}{F((v^W_{w_j})^T v^C_{w_k})} \tag{8}$$

which entails

$$F = \exp \tag{9}$$

and

$$P_{ik} = F((v^W_{w_i})^T v^C_{w_k}) \iff (v^W_{w_i})^T v^C_{w_k} = \log X_{ik} - \log X_i. \tag{10}$$

$\log X_i$ is considered as a bias parameter b^W_i and another bias b^C_k is added to the formula to have symmetry. In addition, as most of X_{ij} are zero value, the log is not well defined. To alleviate this problem, a weight function f will be used in the loss function with the following properties:

- $f(0) = 0$ and $\lim_{x\to 0} f(x)\log^2(x)$ is finite,
- f should increase,
- f should be small for large values.

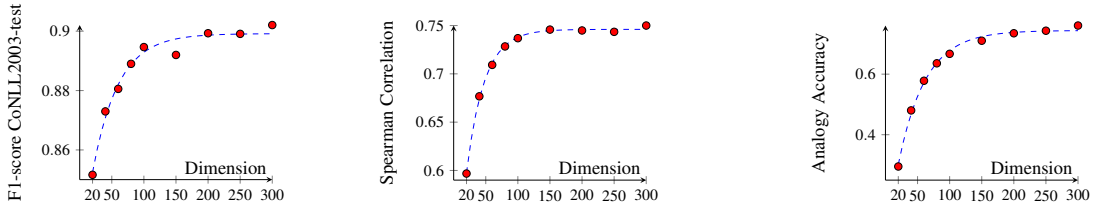
Those properties ensure that f do not overweight small or high values and smoothly cancel the logarithm. Finally, the GloVe loss function is given by

$$J = \sum_{i,j} f(X_{ij})((v^W_{w_i})^T v^C_{w_j} + b^W_i + b^C_j - \log X_{ij})^2. \tag{11}$$

GloVe is another way to produce word embedding vectors from plain text source using contextual information as SG and CBoW. Its main advantages are that it does not use NS for training, and as we mentioned earlier, that it globally compiles the corpus (less computation).

2.1.1.3 FastText

The main drawback of Word2Vec and GloVe (see the two preceding sections) is their inability to compute representations for words not seen during the training phase, called Out Of Vocabulary (OOV) words. To address this issue, the FastText method (Bojanowski et al., 2017) proposes to extend the embedding matrix with character n-grams. This is relevant since often the morphology of words influences their meaning. For instance, bi-gram *«ed»* carries a temporal notion: past. Other examples include: *«er»* for comparison, *«est»* for superlatives and *«ly»* for adverbs. FastText includes these subword information in the representation in order to provide pertinent vectors for unseen words. To do so, the $|V| \times D$ Word2Vec matrix is turned into a matrix W_{FT} of size $(|V| + H) \times D$, H being the size of the hash table of character n-grams.



(a) F1-score on the CoNLL2003 dataset (Named Entity Recognition task) from Sang and Meulder (2003), explained by the dimension of FastText Embeddings.

(b) Spearman Correlation with WordSim-353 (Finkelstein et al., 2002, semantic similarity task) explained by the dimension of FastText Embeddings.

(c) Analogy Accuracy (following Grave et al. (2018)) on Google Dataset (Mikolov et al., 2013a) explained by the dimension of FastText Embeddings

Fig. 5: Evaluation metrics explained by the dimension FastText Embeddings. Dashed lines are trend lines calculated with the model $f_\lambda(x) = \lambda_1 - \lambda_2 e^{-x/\lambda_3}$, $\lambda = (\lambda_1, \lambda_2, \lambda_3)$ being parameters for f_λ. This model was chosen because we claim that, with fixed parameters, when the dimension is high enough the accuracy can not increase further a threshold λ_1. λ_2 controls x-axis offset and λ_3 the convergence speed to λ_1. Red points represent observations.

For FastText, a character n-gram is a subset of n consecutive characters in a word w_i. Every n-gram is hashed[1] in order to get the index of the row of W_{FT} representing it. In the implementation of FastText (Bojanowski et al., 2017), vector representations are computed as follows:

$$\underbrace{v_{w_i}^{W_{FT}}}_{\text{FastText representation}} = \underbrace{W_{FT}^{i,*}}_{\text{word representation}} + \underbrace{\sum_{n=\text{minn}}^{\text{maxn}} \sum_{z \in G^n(w_i)} W_{FT}^{N+h(z),*}}_{\text{sum of n-gram representations}}, \tag{12}$$

where:

- minn and maxn are respectively the minimum and maximum size of character n-grams.
- $G^n(w_i)$ is the set of n-grams in w_i, $G^n(*)$ being the set of all possible n-grams, such that $G^n(w_i) \subset G^n(*)$.
- $W_{FT}^{k,*}$ is the k-th row of a FastText storage matrix W_{FT}.
- $h : G^n(*) \to [\![1, H]\!]$ the hashing function: mapping a n-gram to an index.

FastText enables the computation of a word embedding via the sum of its n-gram embeddings. For instance, 3-grams of the word « *<HELLO>* » are: «*<HE*», «*HEL*», «*ELL*», «*LLO*», and «*LO>* ». Thus, the representation for « *<HELLO>* » is: $v_{<\text{HELLO}>} = v_{\text{HELLO}} + v_{<\text{HE}} + v_{\text{HEL}} + v_{\text{ELL}} + v_{\text{LLO}} + v_{\text{LO}>}$.

2.1.1.4 *Knowledge Enhanced Word Embeddings*

Word embedding techniques presented so far are not able to deal with word polysemy. Indeed, static word embeddings map each word to a vector which is unable to change for different word usage. Some researchers (Iacobacci et al., 2015; Lu et al., 2020) proposed to use external tools and knowledge bases to alleviate this problem and embed words with regard to their different meanings. SensEmbed (Iacobacci et al., 2015) and CoupledCR (Lu et al., 2020) are examples of such solutions and both use the following workflow.

First, dealing with polysemy is achieved by a slight modification of the word embedding model. A vector is no longer mapped to a word but to a word sense. For instance, the word *spring* has two word senses: the season and the metallic object, which leads to two word vectors, one for each meaning. In the literature, those vectors are called concept vectors because they represent the concept of the word.

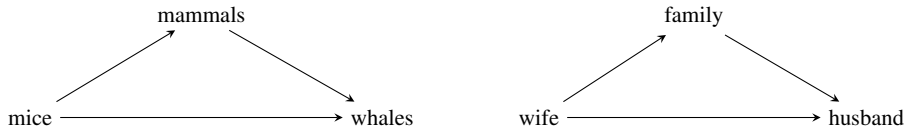
Then, the textual data is preprocessed by replacing each word by the corresponding concept when possible (words are left if no matching concept is found): this is obtained by using a word disambiguation tool (such as Babelfy used in Iacobacci et al. (2015)) or annotations in Wikipedia (Lu et al., 2020). The training of concept/word vectors is performed similarly to word2vec SG, CBoW or GloVe. In addition to that, CoupledCR embeds the Wikipedia hyperlink graph and adds it to the concept vector through concatenation followed by an SVD. Once the concept/word vectors are optimised, it is possible to recover the similarity between two words by taking the most similar concept vectors (w.r.t. cosine similarity) attached to those words (Iacobacci et al., 2015).

This technique is convenient because it enables word polysemy with word embeddings and it has been shown more effective than previous methods at recovering human-like similarities between words. The main disadvantage is that it relies on third-party tools and good annotations which breaks the end-to-end fashion of word embeddings.

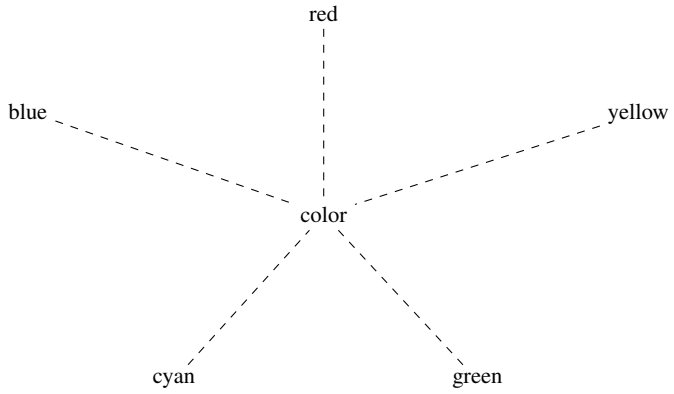
2.1.2 *Non-Euclidean Static Word Embeddings*

Entailment is a term commonly used to refer to asymmetric relations such as *is_a* or *type_of*. For instance a *dog* is a *mammal* but a *mammal* is not always a *dog*. As mentioned ear-

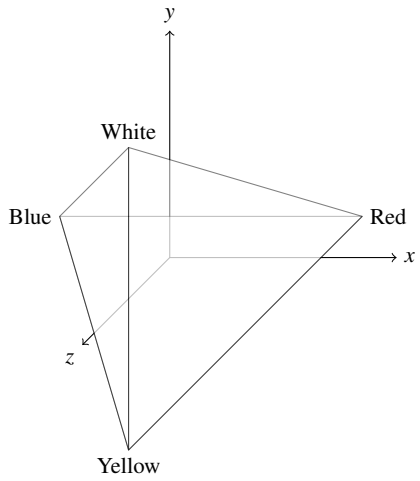
[1] Using the FNV algorithm (Bojanowski et al. (2017))



(a) Triangular inequality. This property constrains hierarchy hyponyms to be close to each others in Euclidean embedding space. *Mice* and *whales* do not share common meaning but they must be close because they share a common hypernym *mammals*. It can be virtuous, as shown by the *family* example.



(b) In $\mathbb{R}^2$, the maximum number of points sharing the same nearest neighbor is 5, as represented by the pentagon. Therefore, more dimension is required to represent complex relationships Sun et al. (2015) (e.g. a Hierarchy).



(c) The maximum number of equidistant points is 4 in $\mathbb{R}^3$, as represented by this tetrahedron. Therefore, it is not possible to represent families of words with homogeneous semantic similarities and more than $d+1$ elements in $\mathbb{R}^d$ Sun et al. (2015).[2]

Fig. 6: Example of properties in $\mathbb{R}^d$

lier, projecting asymmetric relations using Euclidean embedding is complex. Actually, the Euclidean distance and most of Euclidean metrics are symmetric, i.e. $f(w_i, w_j) = f(w_j, w_i)$. For example, the cosine similarity in Euclidean space does not consider the order of the input vectors. Thus, it is ambiguous to distinguish words as hyponyms or hypernyms of others (Sun et al., 2018).

Furthermore, representing complex data structures (such as language) is successfully achieved in Euclidean spaces with a large number of dimensions. To corroborate this statement, we evaluated the performances of Euclidean embeddings on three different tasks w.r.t. the embedding dimension, as shown in Figure 5. This behaviour is due to the underlying properties in Euclidean spaces as shown in Sun et al. (2015) and Laub and Müller (2004). Some of them are represented in Figure 6. More degrees of freedom are required to emancipate from those properties that constrain the representation power of Euclidean embeddings. Besides, high dimensional spaces require more parameters and thus complicate the optimisation, as pointed by Nickel and Kiela (2017).

[2] Source code for the tetrahedron by Ignasi `https://tex.stackexchange.com/questions/174317/creating-a-labeled-tetrahedron-with-tikzpicture`.

The point is not to discredit Euclidean embedding, but to claim that these properties are adapted for certain tasks (such as Analogy or Similarity) but fail to represent complex relations such as entailment (Nickel and Kiela, 2017, 2018). In the next section, we detail non-Euclidean geometries more adapted to this class of relations. Those geometries are often used to embed Knowledge Graph, but this usage is out of the scope of this survey. Instead, we mostly review works where these geometries are used with plain-text data.

2.1.2.1 Probabilistic Word Embeddings (Word2Gauss)

Probabilistic Word Embeddings are word embeddings that map each word to a distribution probability (e.g. Gaussian). Instead of modelling each word by a vector and then infer a probability for $p(w_c|w_t)$ as done with Word2Vec or GloVe, probabilistic embeddings (Vilnis and McCallum, 2015) directly parameterise a distribution probability and estimate $p(w_c|w_t)$ in the probabilistic embedding space. Then distance between words is estimated by metrics between distributions as Kullback- Leibler (KL) divergence, Expected Likelihood (EL) or Wasserstein (Sun et al., 2018). The first two are

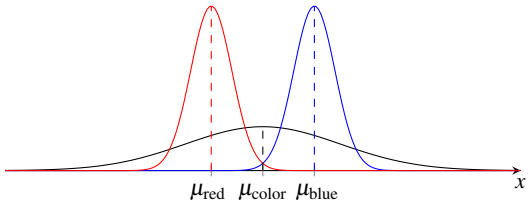


Fig. 7: Example of embedded hierarchy in the probabilistic Embedding space, as shown by Vilnis and McCallum (2015).

defined as

$$EL(p,q) = \int p(x) \cdot q(x) dx.$$
$$KL(p,q) = \int p(x) \cdot \log \frac{p(x)}{q(x)} dx. \quad \text{where } p, q \text{ are distributions.} \tag{13}$$

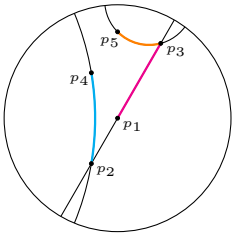
Specifically, EL is used to represent symmetric relations as synonymy or analogy while KL is used to build asymmetric ones (as entailment, or hierarchies).

Gaussian Word Embeddings or Word2Gauss (Vilnis and McCallum, 2015) is an implementation of this approach. Considering a Word2Gauss Embedding WG, each word is associated with a normal distribution: $v_{w_i}^{WG}(x) = \mathcal{N}(x, \mu_i, \sigma_i)$, where $\mu_i \in \mathbb{R}^D$ and $\sigma_i \in \mathbb{R}^{D \times D}$ are respectively the mean and the covariance matrix of the normal law corresponding to w_i. The mean interpretation is comparable to Word2Vec vectors. However, the variance encodes the uncertainty of a word. Here, uncertainty refers to how generic the word is. In other terms, words with high uncertainty (high variance) are very generic while words with low uncertainty (low variance) are very specific. For instance, the distribution of *fruit* must be very uncertain since there is a large variety of fruit hyponyms. Thus, we are not sure of what kind of fruit this *fruit* is. Now, an *apple* is more certain because it has less hyponyms. Therefore, the variance is an important addition of Word2Gauss w.r.t. Word2Vec as it enables hierarchical representations. Figure 7 shows how hierarchies can be represented in the probabilistic embedding space, using the uncertainty.

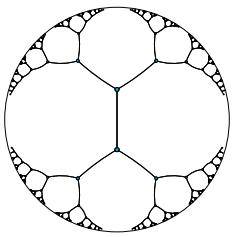
The optimisation process is similar to Word2Vec and relies on distances to embed asymmetric (KL) or symmetric (EL, Wasserstein) relations. Some experiments show that this advantageous ability to combine asymmetric and symmetric metrics in the same space is beneficial. Actually, it can incorporate information from taxonomies / knowledge graphs and plain text data simultaneously and outperforms the standard approach (Sun et al., 2018).

2.1.2.2 Hyperbolic Word Embeddings (Poincaré / Lorentz)

Hyperbolic geometry is anisotropic: the distance depends on the direction of the movement. Euclidean shortest paths are straight lines where hyperbolic ones are curved: the curvature intensity depending on the location in hyperbolic space. Hyperbolic Word Embeddings are representations for words using the anisotropic attitude of the hyperbolic distance to embed structured data, as shown in Nickel and Kiela (2017) with a tree. Figure 8 (taken from their work) shows how this can be done: root elements are placed in the centre and most specific elements are positioned on the edges. Hierarchies are preserved using the hyperbolic distance in order to create tree-like structure.



(a) Geodesics in the Poincaré disk represented by black lines. It is noticeable that distances between points depends on the location of those.



(b) Tree embedding in the Poincaré disc. Shortest paths between points are represented by black lines.

Fig. 8: Representations of the Poincaré disc, taken from Nickel and Kiela (2017)

Table 1: Mean Average Precision (MAP) of Hyperbolic and Euclidean word embeddings on two WordNet tasks (reconstruction and link prediction). Reconstruction consists of learning every WordNet links and predict them based on distance. Link prediction evaluates generalisation: WordNet links are randomly separated in a train and test sets. Word embeddings are trained on the train set and try to predict links from the test set. Experiments and results are taken from the work of Nickel and Kiela (2017).

WordNet Task	Embedding kind	Embedding dimensionality					
		5	10	20	50	100	200
Reconstruction	Euclidean	**0.024**	0.059	0.087	0.140	0.162	0.168
	Hyperbolic	0.823	0.851	0.855	**0.86**	0.857	0.87
Link Prediction	Euclidean	**0.024**	0.059	0.176	0.286	0.428	0.490
	Hyperbolic	0.825	0.852	0.861	**0.863**	0.856	0.855

There are two main hyperbolic models used to embed data in the literature (Nickel and Kiela, 2018; Leimeister and Wilson, 2018; Tifrea et al., 2019; Nickel and Kiela, 2018; Sun et al., 2015): Poincaré and Lorentz models. The major difference relies on the fact that the Poincaré model is bounded by the unit sphere while the Lorentz model is infinite. Therefore, the Poincare model is harder to implement than the Lorentz one since it generates numerical instabilities as 1 represents infinity. Yet, its geometry is easier to comprehend, as confirmed by Nickel and Kiela (2018), because concepts of a similar hierarchy tend to coalesce along a single radius. Also, the specificity of concepts grows as the norm increases. In other words, central area should contain generic concepts whereas edges should have the most specific ones as shown in Nickel and Kiela (2017). Such characteristics are harder to observe on the Lorentz model. However, numerical stability of Lorentz and visualisation simplicity of Poincaré can be used simultaneously since there is a diffeomorphism between both models, as presented by Nickel and Kiela (2018).

Complex hierarchy such as WordNet (Miller, 1995) was projected into hyperbolic spaces (both Lorentz and Poincaré, Nickel and Kiela (2018, 2017)). In all experiments, hyperbolic embeddings lead to a higher reconstruction accuracy even with small dimensionality. In comparison, best reconstruction performance for Euclidean embeddings is reached with high dimensionality and is low, as shown in Table 1.

The optimisation process is the following. Considering a vocabulary of concepts V, a set of edges $\mathscr{E} = \{(w_a, w_b), w_a \in V, w_b \in V\}$ extracted from a hierarchy and a hyperbolic embedding H, the objective is to minimise the hyperbolic distance $d_{\mathscr{H}}$ between existing edges while maximising $d_{\mathscr{H}}$ for non-existent ones. Then, following Nickel and Kiela (2017), the objective is to maximise:

$$\sum_{(w_a,w_b)\in\mathscr{E}} \log \frac{e^{-d_{\mathscr{H}}(v^H_{w_a}, v^H_{w_b})}}{\sum\limits_{w'_b \in \overline{\mathscr{E}}(w_a)} e^{-d_{\mathscr{H}}(v^H_{w_a}, v^H_{w'_b})}} \tag{14}$$

where: $\overline{\mathscr{E}}(w_a) = \{w'_b, (w_a, w'_b) \notin \mathscr{E}\} \cup \{w_a\}$, the set of concepts not connected to w_a including w_a.

Another possible use-case for hyperbolic embeddings is to discover hierarchies from raw pair-wise similarities (Sun et al., 2015; Nickel and Kiela, 2018). The idea is to derive global structures from local similarity relations between concepts. Then the distance in the embedded space should encode the compatibility between concept and entailment information.

We are more interested into Hyperbolic space version of SG or GloVe (Tifrea et al., 2019; Leimeister and Wilson, 2018), replacing the dot product by the hyperbolic distance. For similarity and analogy tasks, it did not improve performance compared to high dimensional Euclidean equivalent, except for Glove with a noticeable difference. This confirms our point on the fact that Euclidean embeddings are more adapted for analogy and similarity tasks. Yet, hyperbolic embeddings still exhibit striking performances with lower dimensions.

2.2 Contextualised Word Embeddings

Contextualised Word Embeddings are directly opposed to static word embeddings since they produce variable word representations. Here, contextualised word embeddings propose to represent a given word based on the sentence in which it occurs. Therefore, a context-independent representation for a given word is not compiled from a dataset. Instead, local information is extracted using a language model and a contextualised word embedding vector is computed depending on the local usage.

Contextualised word embeddings handle three major problems of their static counterparts. Firstly, in the original modelling, static Euclidean embeddings cannot cope with polysemy since they use a unique global vector for each word. In the best case, knowledge enhanced word embeddings (see section 2.1.1.4) are able to cope with polysemy using concept but they require third-party tools and are not end-to-end. On the contrary, contextualised word embeddings adapt representations to current situations, theoretically leading to more relevant vectors. Indeed, the meanings of words highly depend on the context in which they occur. Secondly, it enables representations for n-gram of words. In some cases, the unigram language model is not enough to infer the meaning of a group of words. For instance, the sense of the English expression *'Speak of the devil'* is very different from the individual meaning of the words composing it. Static word embeddings are theoretically adapted to unigram representation, thus, they are not convenient in this situation. The third and last point concerns the distributional hypothesis stating that the meaning of a word is distributed in its context. In practice, static Euclidean embeddings assume this distribution to be uniform. In other words, each context word equally participates in the meaning of target words. However, this is not totally right for each context window, as highlighted by the work of Ling et al. (2015): some context words do not participate (as stop words), some others participate more, or worse, have an opposite meaning.

An important remark is that contextualised word embeddings still employ static word embeddings to project the text to a continuous space. Then the projection is fed to sequence processing model that combines static features or enrich those to mutually embed contextual information.

According to Yang et al. (2019), there are two main approaches: autoregressive or autoencoding. For each of them, we will dedicate a section and examples. In the following, we consider a sentence (i.e. sequence of words) $S = w_1, ..., w_T$.

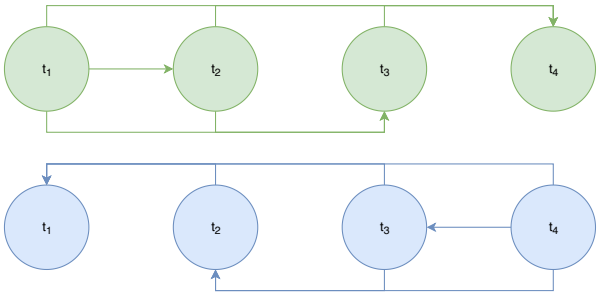


Fig. 9: Autoregressive modelling for a sequence of tokens. The first model is the forward factorisation and the second one is the backward factorisation.

Then, the final goal of contextualised word embedding is to compute the representation of w_i given the context of the sentence: $(w_j)_{j\in[\![1,i-1]\!]}$ for the autoregressive model, and $(w_j)_{j\in[\![1,T]\!]\setminus\{i\}}$ for the autoencoding model.

2.2.1 Autoregressive

Autoregressive models are models inspired from n-gram Language Modelling (Józefowicz et al., 2016). Indeed, n-gram Language Models (LM) try to estimate $p(w_t|w_{t-1},...,w_{t-n})$ (see Figure 9), i.e. anticipate the upcoming word in a sentence given the n preceding words. Autoregressive models extend the limit of n-gram models by representing previous words by context vectors. Thus, they only consider relevant words in the preceding context. They seek to maximise (Gu et al., 2018)

$$p(S) = p(w_1,...,w_T) = \prod_{i=1}^{T} p(w_i|w_{j<i}) \tag{15}$$

for all sentences S in the corpus, hence the name of these models. Actually, each word is exclusively explained by words previously seen in the sequence.

The first autoregressive models often use the combo pre-trained static embeddings + RNN + softmax (Akbik et al., 2018; Peters et al., 2018; McCann et al., 2017; Józefowicz et al., 2016) to represent the temporal sequence with a hidden context vector and then create an output distribution for words. Also, the sequence is reversed to perform bidirectional language understanding. However, as mentioned by Devlin et al. (2019); Yang et al. (2019), the resulting contextualised embeddings is divided, in the sense that the past and post context are not merged but considered independently. If it were not the case, then the model would see upcoming words, thus leading to a useless or trivial representation. As we will show later, autoencoding architectures achieve inference with a unified context which can lead to better understanding of natural language, as proven by Devlin et al. (2019) and Yang et al. (2019).

More recent autoregressive model widely use transformer decoder, as GPT (Radford, 2018; Radford et al., 2019; Brown et al., 2020). Using transformer instead of LSTM or recurrent network has shown to be beneficial. Even when the number of parameters is increased, transformer-based models tend to be more effective.

The OOV issue can arise in autoregressive embeddings as well, since they use a preset vocabulary. This is why, character CNN, proposed in Józefowicz et al. (2016) and McCann et al. (2017), as well as byte pair encoding vocabulary, in Radford (2018), are used in complement with the contextualised representation to comprehend the morphology of words and extend representation to unknown words. Another solution is to train bidirectional systems with characters instead of words and then concatenate static pre-trained representations, as FLAIR method proposed in Akbik et al. (2018). The two following paragraphs detail ElMo and FLAIR approaches.

2.2.1.1 ELMo

ELMo (Embedding for Language Modelling), proposed by Peters et al. (2018), is an autoregressive contextualised word embedding based on words, using a RNN. The idea is to create contextualised features by taking the hidden values of a multilayer bidirectional RNN learning a language model: predicting the upcoming word knowing the previous ones in a sentence. In the end, stacked hidden layers of the model should encode information of this language model.

2.2.1.2 FLAIR

FLAIR, proposed by Akbik et al. (2018), is an autoregressive contextualised word embedding based upon characters, using a RNN. It considers sequence of characters instead of sequence of words. Word representations are finally extracted using the space character to determine boundaries of words. For instance, the forward (resp. backward) FLAIR model uses hidden state of the last (resp. first) letter of a word. Both hidden states (forward and backward) are concatenated in order to unify the context into a single vector. It is beneficial to add to this representation a static word embedding, according to Akbik et al. (2018). An explanation for this behaviour is that static representations add word-level information to the character-based FLAIR representations.

2.2.1.3 GPT

GPT (Radford, 2018; Radford et al., 2019; Brown et al., 2020) is a model that has been improved over time. It is an autoregressive contextualised word embedding using a transformer decoder to comprehend word sequences. The third version GPT-3 (Brown et al., 2020) is probably paving

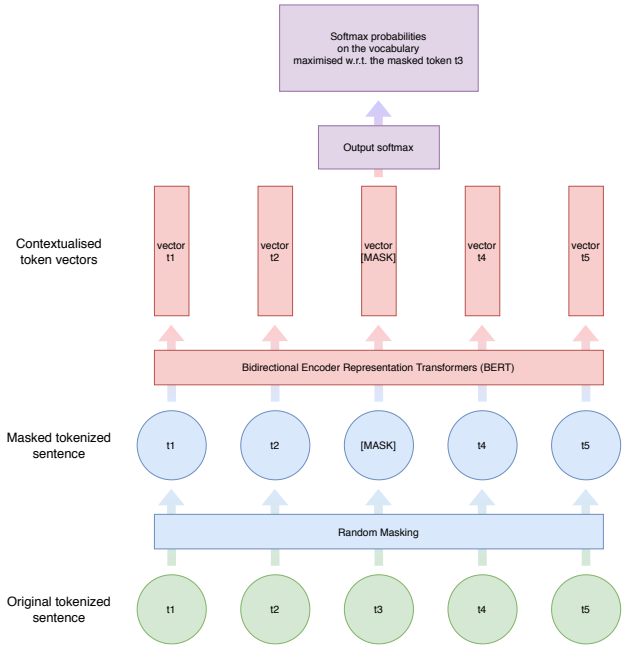


Fig. 10: Masked Language Modelling (MLM) task, used to train autoencoding contextualised word embeddings. Tokens are randomly replaced in sentence by the `[MASK]` token. At the end, the contextualised representation of `[MASK]` is fed to a classification layer over the vocabulary. The objective is to predict the original token.

the way to future contextualised word embedding, since it is able to perform NLP tasks with limited efforts spent on fine-tuning. Instead, the model is trained on a very large corpus and is then directly used to answer NLP tasks. Actually, any NLP task can be formulated as a question, and GPT-3, as a generative autoregressive model, is ideally able to generate the correct answer (for instance *«Translate to French: cheese.»*). When NLP task is proposed as sentences, then, the NLP task objective is close to the unsupervised autoregressive objective. Hopefully, with a sufficiently large amount of data and enough training, we will be able to reconcile both.

2.2.2 Autoencoding

Autoencoding-like approaches such as BERT approach, proposed by Devlin et al. (2019), try to encode bidirectional context to predict a target word. Of course, no information is given to the model on the word to predict. This is achieved by using a `[MASK]` token hiding a target word in a sentence. Therefore the objective is to reconstruct the original token based on this masked sentence. Figure 10 pictures this task. Generally, after this step, the model is fine-tuned on a downstream task.

Autoencoding models, particularly BERT, are suspected not to capture language complexity (Niven and Kao, 2019). Instead, it would overfit spurious statistics in the training set. Furthermore, the `[MASK]` token is supposed to introduce noise in the model because downstream tasks do not contain this one, as pointed by Yang et al. (2019). Likewise, the pre-training step is a reconstruction exercise where input sentences are corrupted by the `[MASK]` token. Therefore, the autoencoding embedding never observes real sentences and has to create representations based on a corrupted proxy. A solution to thwart those phenomena is to introduce new hyper-parameters controlling the amount of corruption or to replace the `[MASK]` token with randomly selected real words. However, those are not straightforward to choose nor to optimise. Finally, `[MASK]` tokens are supposed independent. This point, arose by Yang et al. (2019), consists of saying that the reconstruction of each masked token is done without considering the reconstruction of others. In the following, we explain BERT and XLNET embeddings.

2.2.2.1 BERT

BERT architecture proposed by Devlin et al. (2019) is an autoencoding contextualised word embedding using Transformer Encoder from Vaswani et al. (2017). It is trained in two steps. The first one involves two pre-training tasks: *Masked Language Modelling (MLM)* and *Next Sentence Prediction (NSP)*. The second one is a fine-tuning phase. MLM consists of (as said above) masking random token and try to predict the most probable word given the whole context. NSP is a classification problem where the model has to tell whether a sentence can follow another. After the training phase, the representation for words is taken from the output of the encoder. We do not detail here more recent version of BERT, such as ROBERTA Liu et al. (2019), which mainly focus on reducing the computational costs of training as well as its stability and global performance.

2.2.2.2 XLNET

XLNET from Yang et al. (2019) is a method trying to make the most of both autoregressive and autoencoding approaches. It reduces noise and approximations introduced by autoencoding models while benefiting from unified context encoding. Therefore, it benefits from a complete representation of the context without having to mask inputs and does not require masked tokens.

2.2.3 Training

Today, two training schemes are opposing. The first one is a two step training with a pre-training phase and a fine-tuning phase (FLAIR, ELMo, BERT, XLNET). The model is first pre-trained on a large corpus with unsupervised tasks (MLM, NSP, maximise Eq. 15) and then fine-tuned on a specific NLP task for real-world usage. The second scheme (GPT-3) is a

Table 2: Global, Intrinsic and Extrinsic Evaluations in term of properties.

	Global	Intrinsic	Extrinsic
Involve external data		✓	✓
Involve external models			✓

one-step training where an autoregressive model is massively trained using a huge amount of data and an enormous model. Then, answers of NLP tasks are generated by feeding the model with correct queries. As autoencoding models are not necessarily designed to be generative model, it is hard to use them in the second scheme since they will be unable to generate the answer to a query. Yet, the second scheme seems more advantageous because it does not require a large amount task-specific data or further training costs.

3 Evaluation

In this section, we propose to explore various evaluation solutions and metrics on Word Embeddings. As we mentioned in the previous section, different methods lead to different structures having pros and cons for solving specific tasks. In Table 2, we propose a partition , based on the work of Schnabel et al. (2015), of word embeddings evaluation measures that we will detail in the following.

3.1 Global

Based on Amsaleg et al. (2015) and Torregrossa et al. (2020), let global metrics be the set of measures quantifying the distribution of vectors in an embedding space. In other words, any metric that does not need external labelled dataset and gives information on the representation capacity of the embedding.

For instance, the Intrinsic Dimensionality (ID), coming from Information Retrieval and introduced by E. Houle et al. (2012), aims to be a local metric correlated with the ideal number of dimension required to project datapoints in an embedding space (Amsaleg et al., 2015; E. Houle et al., 2012). One could create a global metric by studying the distribution of these local metrics in the embedding space. Another example of a global measure would be the δ_{avg}-hyperbolicity, introduced by Tifrea et al. (2019), that measures whether the embedded objects have a tree-like structure.

Torregrossa et al. (2020) introduced other metrics used in different areas than NLP such as computer vision. They expect them to be simple and fast indicators on the representation capacity of embeddings. Those are particularly adapted to count significant dimensions in matrices. In other words, they estimate the minimal dimensionality required to compress matrices with a decent reconstruction error. Such metrics are also usable on word embedding matrices in order to discover the number of significative dimension of the representation: a high number of those being correlated with a good usage of the space dimensionality and potentially to effective representations.

Those two metrics are: *effective rank (erank)* (Roy and Vetterli, 2007) from signal processing and *empirical dimension (edim)* (Poling and Lerman, 2014) from computer vision. Their mathematic formulations are given by

$$\text{erank}(W) = \exp\left(-\sum_{i=1}^{D}\left[\frac{s_i}{\sum_{j=1}^{D} s_j}\cdot\log\left(\frac{s_i}{\sum_{j=1}^{D} s_j}\right)\right]\right), \quad (16)$$

$$\text{edim}(W,p) = \frac{||s||_p}{||s||_{\frac{p}{1-p}}}, \quad (17)$$

with $s = (s_i)_{i\in[\![1,D]\!]}$ being the singular values of a matrix W and $||x||_p = (\sum_i x_i^p)^{\frac{1}{p}}, p \in [0,1]$.

They both use singular values because those are related to the principal axis of variance of the matrix row vectors, the singular value itself being the value of this variance. The goal of erank and edim is to study the contribution of each singular value in the decomposition of the matrix row vectors. In the end, erank and edim return continuous values between $[1,D]$, corresponding to the number of singular values that matter in this decomposition. A value close to D means that singular values are equally important whereas a value close to 1 exposes a preponderant singular value that concentrate almost all information. Other values indicate that the embedding is compressible into a space with a lower dimensionality (almost equals to the erank or edim value).

3.2 Intrinsic

Intrinsic evaluations (Schnabel et al., 2015) reflect the coherence between word vectors and human judgement. They need external human-labelled data in order to compare the vector structures with a ground truth. The *intrinsic* term refers to the fact that it measures the structure of the vectors in the embedding space without adding external information to the model. Indeed, those evaluations assess the global quality of the language representation.

Such evaluations commonly rely on similarity metrics in order to compare, retrieve or coalesce word vectors. The Cosine Similarity:

$$\cos(v_{w_A}, v_{w_B}) = \frac{\langle v_{w_A}, v_{w_B}\rangle}{||v_{w_A}||\cdot||v_{w_B}||}, \quad (18)$$

is the most used metric for intrinsic evaluations. Actually, the dot product is not chosen since word vector norms capture

Table 3: Similarity datasets (Bakarov (2018)).

Name	Size	Pairwise Score based on
WordSim353 (Finkelstein et al., 2002)	353	Synonymy (common words)
MEN (Bruni et al., 2014)	3000	Synonymy (common words)
RG (Rubenstein and Goodenough, 1965)	65	Synonymy (common words)
SimLex-999 (Hill et al., 2015)	999	Synonymy (common words)
SimVerb (Gerz et al., 2016)	3500	Synonymy (essentially verbs)
RareWords (RW) (Luong et al., 2013)	2034	Synonymy (low frequency words)
HyperLex (Vulić et al., 2017)	2616	Entailment (common words)

Table 4: Analogy datasets (Bakarov (2018)).

Name	Size	Relation Types
Google Analogy Mikolov et al. (2013a)	19,000	Capital, Country, Family, Currency, Cities, Morphology
MSR Mikolov et al. (2013c)	8,000	Morphology

the frequency of words, as proven by Schakel and Wilson (2015). We consider three types of intrinsic measures, commonly used in the state of the art (as shown in Figure 11, Mikolov et al. (2013b); Schnabel et al. (2015); Pennington et al. (2014); Bojanowski et al. (2017); Tifrea et al. (2019)).

3.2.1 Similarity (Figure 11a – Table 3)

It quantifies agreement between human word similarity judgement with the word vectors of an embedding. High agreement rate means that word vectors represent well words and language aspects of the dataset. This rate is obtained using a correlation measure. Spearman is the most used correlation for this purpose. This choice is quite relevant because it summarises all kinds of correlation (linear or more complex) into a single score. Specifically, it considers a matrix of pairwise similarities $(S_{ij})_{i,j\in[\![1,N]\!]}$ obtained after compilation of several human judgements. Then, Spearman correlation is performed on the set of points, using the cosine similarity (18): $\{(x = S_{ij},\ y = \cos(v_{w_i}, v_{w_j})), i \in [\![1,N]\!], j \in [\![i+1,N]\!]\}$. The cosine similarity can be replaced by other measures capturing different kinds of relations (as entailment as done by Tifrea et al. (2019); Nickel and Kiela (2017)).

3.2.2 Analogy (Figure 11b – Table 4)

It consists of solving a relationship problem. A pair of words is given (A,B) satisfying a relation $\mathscr{R}$, i.e. A is something to B or $\mathscr{R}(A,B)$ holds. Then, a word C is given, and the objective is to find a word D in the embedding so that $\mathscr{R}(C,D)$ holds. A convenient way to write the problem is $A : B - C : ?$. For instance, we should find *Spain* as a solution of the problem *Paris* : *France* − *Madrid* : ?, because *Paris* is the capital of *France* as *Madrid* is the capital of *Spain*. D is found by solving the following problems using the cosine similarity (18) between vectors, following Levy et al. (2015):

- 3CosAdd (1): $\underset{D\in V\setminus\{A,B,C\}}{\operatorname{argmax}} \cos(v_B - v_A + v_C, v_D)$,
- 3CosAdd (2): $\underset{D\in V\setminus\{A,B,C\}}{\operatorname{argmax}} \cos(v_D, v_B) - \cos(v_D, v_A) + \cos(v_D, v_C)$,
- 3CosMul: $\underset{D\in V\setminus\{A,B,C\}}{\operatorname{argmax}} \frac{\cos(v_D v_B)\cdot\cos(v_D, v_C)}{\cos(v_D, v_A)+\varepsilon}$, $\varepsilon = 0.001$.

The 3CosMul problem is known to be the most effective so far.

Table 5: Categorisation datasets (Bakarov (2018)).

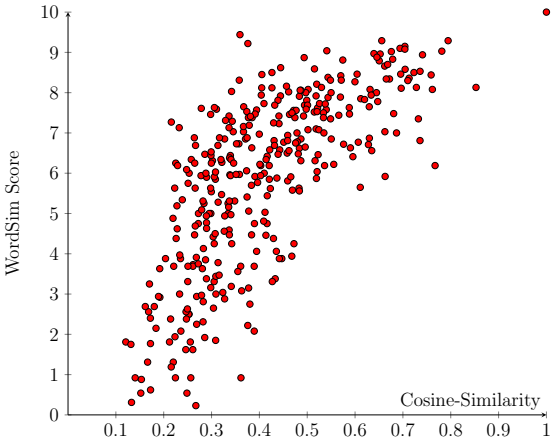
Name	Size	Number of Clusters
Battig Baroni et al. (2010)	5330	56
AP Almuhareb (2006)	402	21
BLESS Baroni and Lenci (2011)	200	17

3.2.3 Categorisation (Figure 11c – Table 5)

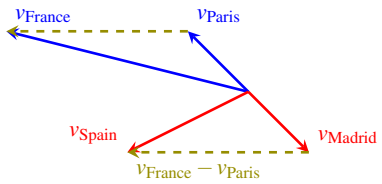
It characterises the capacity of the embedding to distinguish semantic clusters among a set of words. A categorisation dataset $\mathscr{D}$ is defined by a set of N classes $\mathscr{C} = (C_i)_{i\in[\![1,N]\!]}$, and a set of K words $W = (w_j)_{j\in[\![1,K]\!]}$. Each word belongs to a specific class, thus $\mathscr{D} = \{(w, C(w)), w \in W\}$ indicates the class $C(w) \in \mathscr{C}$ of a word $w \in W$. Then, the goal is to create N reconstructed clusters $\mathscr{C}^r = (C_i^r)_{i\in[\![1,N]\!]}$ with the K word vectors. CLUTO toolkit from Karypis (2003), with default parameters, is generally (Baroni et al. (2014); Schnabel et al. (2015)) used to complete this operation. At the end, comparison between $\mathscr{C}$ and $\mathscr{C}^r$ is performed with the purity measure as explained by Manning et al. (2008):

$$\text{Purity}(\mathscr{C},\mathscr{C}^r) = \frac{1}{K}\sum_{i=1}^{N} \max_{j\in[\![1,N]\!]} \#(C_i \cap C_j^r). \tag{19}$$

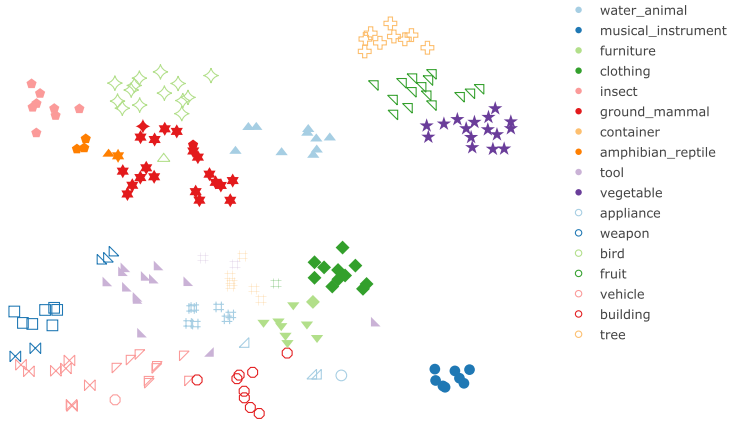
A thorough presentation of notorious datasets for intrinsic tasks is given by Bakarov (2018). We stored commonly used datasets in Tables 3 (similarity), 4 (analogy) and 5 (categorisation). Figure 11 pictures the three tasks. The major issue concerning the intrinsic evaluations is that they are adapted to static word embeddings and do not deal with contextualised ones. In addition, they evaluate on an artificial pre-defined task not necessarily in relation to a real need.



(a) Similarity task on WordSim (Finkelstein et al., 2002). Red dots represent word pairs, x-value being the cosine similarity for this pair in a word embedding and the y-value being the labelled value in the dataset. The spearman correlation is computed on this chart, giving 0.75 in this case.



(b) Analogy task, the relation IsCapitalOf is assumed to be linear and is extracted from the example (*France*, *Paris*). On the figure, v_{Spain} is perfectly fitting $v_{\text{Madrid}} + v_{\text{France}} - v_{\text{Paris}}$. In reality, this is different, therefore the nearest neighbour of $v_{\text{Madrid}} + v_{\text{France}} - v_{\text{Paris}}$ is found in the embedded vocabulary. In practice, cosine similarities between vectors is used.



(c) Categorisation task on Bless dataset (Baroni and Lenci, 2011) dataset (following Bakarov (2018)) with a 300d embeddings projected on a 2d-plane with t-SNE: each color represents a golden category, each mark represents a predicted category. The purity measures the mixture in the predicted categories w.r.t. the golden categories. For instance, predicted bowtie category in the bottom-left corner are dispatched in two golden categories (open-blue: weapon, and open-pink: vehicle).

Fig. 11: Intrinsic task samples.

3.3 Extrinsic

Extrinsic evaluations (Schnabel et al., 2015) need higher level and more complex representations than word embeddings to be solved. Therefore, external information is added via training and additional modelling in order to solve these complex tasks when word vectors may not be enough to solve the extrinsic problem. For instance, static embeddings are fed into a RNN that will name entity over a sentence. The RNN will comprehend sequential information. Another training phase is needed in order to optimise the RNN for the task. Therefore, external information is added to the word vectors through the RNN. Similarly, contextualised word embedding has to be fed into task-specific layers and fine-tuned to be compatible with the task. In both static and contextualised models, the task-specific layers are used to extract the rele-

Table 6: Extrinsic Evaluation

Embedding Type	NER			SA			TC		
	P	R	F1	P	R	F1	P	R	F1
GloVe (768d)	0.833	0.809	0.820	0.381	0.358	0.346	0.908	0.908	0.908
Word2Vec SG (768d)	0.882	0.837	0.858	0.447	0.436	0.415	0.920	0.92	0.919
FastText (768d)	0.881	0.852	0.866	0.440	0.434	0.391	0.918	0.918	0.917
BERT	0.921	0.928	0.925	0.508	0.500	0.471	0.937	0.938	0.937

vant features prior learnt by the embedding model on natural language text.

A large number of other extrinsic tasks are detailed in the survey of Bakarov (2018). As stated by the authors, it is arguable that extrinsic evaluations provide good assessment of word embeddings quality, due to the fact that there are correlations across a wide range of extrinsic tasks. However, this assumption is controversial among the NLP community, since some embeddings can be constructed in order to achieve specific tasks while neglecting others, as shown by Claveau and Kijak (2016). Actually, the global quality of an embedding does not necessarily exist, instead, maximising the quality of the representation with regard to a specific task has more sense, as proven by the same authors and Schnabel et al. (2015). Overall, evaluation of embeddings remains significantly complex as interaction mechanisms across different kinds of tasks and different methods are not well understood.

To overcome the need of proper and high-level evaluation, the famous benchmark GLUE Wang et al. (2018) was proposed to evaluate word embedding models and is now widely used. This benchmark provides nine different natural language understanding tasks aiming at enhancing the simple input-output correspondence evaluation. These nine tasks are supposed to give a wide description of the inner understanding of the word embedding model.

As an example we provide a small benchmark gathering some of the famous extrinsic task in NLP. We consider the three following tasks:

- **Named Entity Recognition (NER)** is a sequence labelling task, where a sentence is given and the model has to assign each word of the sentence to a particular entity class (a person, an organisation or other). CoNLL2003 (Sang and Meulder, 2003) is the dataset we use for the experiments, composed of news thread extracts.
- **Sentiment Analysis (SA)** is a sentence-level classification task where the model is asked to give a sentimental class for each sentence. We use the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013), which is a set of movie reviews and sentiment are either very positive, positive, neutral, negative or very negative (SST-1 setup presented in Zhou et al. (2016)).
- **Text Classification (TC)** is a sentence-level / document-level classification task where the model has to classify text into different categories. AGNews dataset[3] is an online resource composed of news articles falling into 4 categories (World, Sports, Business and Sci/Tech) used here.

As mentioned earlier, we have to train new models when solving extrinsic tasks:

- For NER, we feed the sequence of static vectors or contextualised vectors into a one-layer bidirectional GRU followed by a CRF layer that predict the sequence of labels. Fine-tuning of some of the weights is allowed for contextualised models.
- For TC and SA, we used the sequence of static vectors or contextualised vectors in combination with a one-layer bidirectional GRU followed by a linear mapping fitting the number of classes.

The implementation and pre-trained models are from the Python library *transformers*, presented by Wolf et al. (2019). Table 6 reports the results we obtained using these three tasks and 4 word embeddings (1 is contextualised and 3 are static).

GloVe, Word2Vec Skip-Gram and FastText Skip-Gram are trained on a Wikipedia Dump following Bojanowski et al. (2017). All methods are used with default parameters. The dimension is set to 768 to match the dimensionality of BERT and perform a fair comparison. For each task, we train the bidirectional GRU and CRF model for 5 epochs and allow fine-tuning for BERT embeddings only. We noticed that fine-tuning static embeddings reduces performance gap between them.

BERT is the best of all tested embeddings, and this is explained by the fact that it uses contextual information and therefore can create a word representation adapted to its usage in a sentence. We notice that FastText is the best among static word embedding and this is mainly due to the way it handles OOV. These are simple observations on these tasks and different could be observed on different datasets.

4 Discussion

In this part we discuss properties characterising methods of Section 2.

[3] http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

4.1 Presentation of properties

Table 7 describes different embedding methods using five properties:

- **OOV (Out Of Vocabulary)**: a word embedding will handle OOV if it is able to give representations for unseen words. Basically, every embedding technique is able to do so by introducing a <UNK> token. However, this is not a proper solution for the OOV issue since each unknown word shares an identical vector. In the table, we indicate with a tick mark (✓) embeddings able to generalise the representations to unknown words using, for instance, subword information. The — symbol highlights methods that do not handle OOV completely. In fact, ELMo, BERT and XLNET work at the token level, and the tokeniser might incorporate subword information. Therefore, they handle OOV only if context words around the unknown word are well known.
- **Contextualised**: when the representation of a word is calculated online using its context, i.e. when the word embedding is a contextualised word embedding.
- **Self-supervised** (Zhai et al., 2019): commonly the task used to train an embedding is self-supervised. This means that a surrogate supervised task is automatically extracted from an unlabelled dataset. We do not consider WordNet (Miller, 1995) as an unlabelled dataset, because high-level human knowledge was required to build this graph. Hence techniques that involve WordNet do not satisfy this property.
- **Non-Euclidean**: when the representation is using a non-Euclidean geometry.
- **Multisources**: embeddings use a main optimisation criterion. However, it is possible to enrich word vectors using various kinds of datasets and thus different criteria. For instance in Sun et al. (2018), authors use the KL-divergence and the Wasserstein distance to embed simultaneously a plain text dataset and a knowledge base. Contextualised embeddings are particularly multitask because they often require a final fine-tuning step.

4.2 Analysis

Training data sources are a major leverage for learning effective word embeddings. Plain text data is the most common sort of datasource considered by techniques listed in Table 7. Word embedding is performed with a **self-supervised training** task based on the *«distributional hypothesis»*. Word2Vec, GloVe and FastText have shown to be helpful representations for a wide range of NLP tasks. Indeed, they are often chosen as input prior knowledge of extrinsic problems (Lample et al., 2016; Zhou et al., 2016).

Sometimes, this hypothesis is locally unsatisfied (Ling et al., 2015). Indeed, those models are particularly convenient for building unigram word vectors but not for n-gram ones. **Contextualised** propositions, such as FLAIR or BERT, are word embedding alternatives and solutions to these issues. Their surrogate **self-supervised training** task is based on a language model and therefore try to increase the theoretical reliability of the whole model. In the end, those new architectures globally improved the performance on extrinsic tasks. However, this gain is not free, and this because of the cost of the training phase. Today, on standard computers, FastText tool[4] complete a training on Wikipedia (2.5B tokens) in about a day. Conversely, according to Devlin et al. (2019), on the same corpus, BERT requires not less than 4 days on 16 TPUv3 chips, a massive and expensive physical architecture. A solution to time-consuming step is to download pre-trained multilingual or monolingual BERT models (when available), and perform a fine-tuning step to suit the downstream task. Another one is to do large-batch optimisation as done by You et al. (2019).

Instead of changing the hypothesis, it is possible to work with **non-Euclidean** methods. While Euclidean embeddings produce particularly efficient representations on semantic or analogy tasks, they still seem unable to extract logical thinking or entailment scheme from natural sentences. Hyperbolic or probabilistic embeddings try to reconfigure the representation (introducing asymmetric metrics) in order to integrate these different aspects of language. Hyperbolic spaces are known to encode more information with a lower dimensionality and show performance challenging high-dimensional Euclidean embeddings with low dimension (Leimeister and Wilson, 2018). On entailment task such as HyperLex (Tifrea et al., 2019), hyperbolic word embeddings trained on wikipedia outperform their euclidean counterparts. However, on paradigmatic and syntagmatic tasks, Euclidean word embeddings remain the state of the art. Concerning probabilistic word embeddings, they also introduce uncertainty encoded in the variance of the distributions. Thanks to that, they significantly improved the performance on intrinsic tasks (Sun et al., 2018).

Using multiple sources (**multisources**) of data also helps enhancing word vectors. So far we only evoked plain text data source, but there are many structured sources on the web such as WordNet Miller (1995). Those could be embedded into vectors along with plain text data source as done in Sun et al. (2018).

[4] https://github.com/facebookresearch/fastText

Table 7: High-level properties of various embeddings techniques.

Embedding technique	OOV	Context-tualised	Self-supervised training	Non-Euclidean	Multi-sources
GloVe (Pennington et al., 2014)			✓		
Word2Vec (Mikolov et al., 2013b)			✓		
FastText (Bojanowski et al., 2017)	✓		✓		
SensEmbed (Iacobacci et al., 2015)			✓		✓
CoupledCR (Lu et al., 2020)			✓		✓
Word2Gauss EL (Vilnis and McCallum, 2015)			✓	✓	
Word2Gauss Wasserstein (Sun et al., 2018)			✓	✓	✓
Poincaré GloVe (Tifrea et al., 2019)			✓	✓	
Lorentz Skip-Gram (Leimeister and Wilson, 2018)			✓	✓	
ELMo (Peters et al., 2018)	—	✓	✓		✓
FLAIR (Akbik et al., 2018)	✓	✓	✓		✓
BERT (Devlin et al., 2019)	—	✓	✓		✓
GPT-3 (Brown et al., 2020)	—	✓	✓		✓
XLNET (Yang et al., 2019)	—	✓	✓		✓

5 Conclusion

In this survey, we explained the main categories of widely used word embeddings. Terminologies and definitions are given to properly segment each technique as well as their advantages. Similarly, major metric tools have been introduced. Intrinsic metrics were thoroughly presented in order to give a clear procedure to compute them. Finally, high-level properties are proposed to qualitatively assess word embeddings and highlight uninvestigated approaches.

To conclude, we propose three main research paths.

First, contextualised word embeddings can only be assessed using extrinsic tasks. Therefore it is hard to understand what kind of semantics or syntactics they intrinsically capture. It would be beneficial to see whether the promise of context-dependent representation is fulfilled, and assess the extent of this potential accomplishment.

Hyperbolic deep neural network (Ganea et al., 2018) is a recent kind of neural network able to comprehend more naturally entailment and logical thinking. Many libraries as *geoopt* (Kochurov et al., 2019) offer the possibility to build hyperbolic networks. Therefore, it is possible to build contextualised word embeddings using the hyperbolic geometry and compare it with their Euclidean nemesis whether they are better able to capture grammar or thinking.

Finally, correlations between different kinds of measures (global, intrinsic, extrinsic) is not clear. In other word, it is not obvious to tell the quality of an embedding on a specific extrinsic task regarding its global performance on several intrinsic or global tasks. Investigations on this path would leverage two points. On the one hand, it would give new optimisation criterion for the training phase. On the other hand, it would propose a quick way to select an embedding for an extrinsic task.

Conflict of interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. ACL.

Almuhareb, A. (2006). Attributes in lexical acquisition, Thesis from the University of Essex.

Amsaleg, L., Chelly, O., Furon, T., Girard, S., Houle, M. E., Kawarabayashi, K.-i., and Nett, M. (2015). Estimating local intrinsic dimensionality. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 29–38, New York, NY, USA. ACM.

Bakarov, A. (2018). A survey of word embeddings evaluation methods. *CoRR*, abs/1801.09536.

Baroni, M., Dinu, G., and Kruszewski, G. (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland. ACL.

Baroni, M. and Lenci, A. (2011). How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10, Edinburgh, UK. Association for Computational Linguistics.

Baroni, M., Murphy, B., Barbu, E., and Poesio, M. (2010). Strudel: A corpus-based semantic model based on properties and types. *Cognitive science*, 34:222–54.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Proceedings of NeurIPS, Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krüger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. *OpenAI publications*.

Bruni, E., Tran, N. K., and Baroni, M. (2014). Multimodal distributional semantics. *J. Artif. Int. Res.*, 49(1):1–47.

Claveau, V. and Kijak, E. (2016). Direct vs. indirect evaluation of distributional thesauri. In *International Conference on Computational Linguistics, COLING*, Osaka, Japan.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 999888:2493–2537.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186. arXiv preprint (2018):1810.04805.

E. Houle, M., Kashima, H., and Nett, M. (2012). Generalized expansion dimension. In *Proceedings - 12th IEEE International Conference on Data Mining Workshops, ICDMW 2012*, pages 587–594.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. (2002). Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Ganea, O., Bécigneul, G., and Hofmann, T. (2018). Hyperbolic neural networks. *CoRR*, abs/1805.09112.

Gerz, D., Vulic, I., Hill, F., Reichart, R., and Korhonen, A. (2016). Simverb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, volume abs/1608.00869.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. (2018). Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*.

Gu, J., Bradbury, J., Xiong, C., Li, V. O. K., and Socher, R. (2018). Non-autoregressive neural machine translation. In *International Conference on Learning Representations, ICLR*, volume abs/1711.02281.

Harris, Z. (1954). Distributional structure. *Word*, 10:146–162.

Hill, F., Reichart, R., and Korhonen, A. (2015). SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *American Journal of Computational Linguistics*, 41(4):665–695.

Iacobacci, I., Pilehvar, M. T., and Navigli, R. (2015). SensEmbed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 95–105, Beijing, China. Association for Computational Linguistics.

Józefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Karypis, G. (2003). Cluto: A clustering toolkit. *Technical Report 02-017, University of Minnesota (Department of Computer Science)*.

Kochurov, M., Kozlukov, S., Karimov, R., and Yanush, V. (2019). Geoopt: Adaptive riemannian optimization in pytorch. `https://github.com/geoopt/geoopt`.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Laub, J. and Müller, K.-R. (2004). Feature discovery in non-metric pairwise data. *J. Mach. Learn. Res.*, 5:801–818.

Leimeister, M. and Wilson, B. J. (2018). Skip-gram word embeddings in hyperbolic space. *CoRR*, abs/1809.01498.

Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc.

Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Ling, W., Tsvetkov, Y., Amir, S., Fermandez, R., Dyer, C., Black, A. W., Trancoso, I., and Lin, C.-C. (2015). Not all contexts are created equal: Better word representations with variable attention. In *Proceedings of the 2015 Con-*

ference on Empirical Methods in Natural Language Processing, pages 1367–1372, Lisbon, Portugal. Association for Computational Linguistics.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Lu, W., Zhang, Y., Wang, S., Huang, H., Liu, Q., and Luo, S. (2020). Concept representation by learning explicit and implicit concept couplings. *IEEE Intelligent Systems*.

Luong, T., Socher, R., and Manning, C. (2013). Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, Sofia, Bulgaria. Association for Computational Linguistics.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

McCann, B., Bradbury, J., Xiong, C., and Socher, R. (2017). Learned in translation: Contextualized word vectors. volume abs/1708.00107.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Nickel, M. and Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6338–6347. Curran Associates, Inc.

Nickel, M. and Kiela, D. (2018). Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry. In *Proceedings of the International Conference on Machine Learning, ICML*.

Niven, T. and Kao, H. (2019). Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proc. of NAACL*.

Poling, B. and Lerman, G. (2014). A new approach to two-view motion segmentation using global dimension minimization. *International Journal of Computer Vision*, 108/3.

Radford, A. (2018). Improving language understanding by generative pre-training. *OpenAI publications*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI publications*.

Roy, O. and Vetterli, M. (2007). The effective rank: A measure of effective dimensionality. In *2007 15th European Signal Processing Conference*, pages 606–610.

Rubenstein, H. and Goodenough, J. B. (1965). Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633.

Sang, E. F. T. K. and Meulder, F. D. (2003). Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*.

Schakel, A. M. J. and Wilson, B. J. (2015). Measuring word significance using distributed representations of words. *CoRR*, abs/1508.02297.

Schnabel, T., Labutov, I., Mimno, D., and Joachims, T. (2015). Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 298–307, Lisbon, Portugal. Association for Computational Linguistics.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Sun, C., Yan, H., Qiu, X., and Huang, X. (2018). Gaussian Word Embedding with a Wasserstein Distance Loss. *CoRR*, abs/1808.07016.

Sun, K., Wang, J., Kalousis, A., and Marchand-Maillet, S. (2015). Space-time local embeddings. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 100–108. Curran Associates, Inc.

Tifrea, A., Becigneul, G., and Ganea, O.-E. (2019). Poincaré glove: Hyperbolic word embeddings. In *International Conference on Learning Representations (ICLR 2019)*.

Torregrossa, F., Claveau, V., Kooli, N., Gravier, G., and Allesiardo, R. (2020). On the correlation of word embedding evaluation metrics. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 4789–4797, Marseille, France. European Language Resources Association.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Vilnis, L. and McCallum, A. (2015). Word representations via gaussian embedding. In *International Conference on Learning Representations, ICLR 2015*.

Vulić, I., Gerz, D., Kiela, D., Hill, F., and Korhonen, A. (2017). HyperLex: A large-scale evaluation of graded lexical entailment. *American Journal of Computational Linguistics*, 43(4):781–835.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2019). Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. (2019). XLNet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

You, Y., Li, J., Hseu, J., Song, X., Demmel, J., and Hsieh, C. (2019). Reducing BERT pre-training time from 3 days to 76 minutes. *CoRR*, abs/1904.00962.

Zhai, X., Oliver, A., Kolesnikov, A., and Beyer, L. (2019). S^4l: Self-supervised semi-supervised learning. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., and Xu, B. (2016). Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*.