Scientific Research Publishing

International Journal of Intelligence Science, 2023, 13, 1-21
https://www.scirp.org/journal/ijis
ISSN Online: 2163-0356
ISSN Print: 2163-0283

Word Embeddings and Semantic Spaces in Natural Language Processing

Peter J. Worth

Dept. of Computer Science and Electrical Engineering, Florida Atlantic University, Boca Raton, FL, USA
Email: pworth2022@fau.edu

How to cite this paper: Worth, P.J. (2023) Word Embeddings and Semantic Spaces in Natural Language Processing. *International Journal of Intelligence Science*, **13**, 1-21.
https://doi.org/10.4236/ijis.2023.131001

Received: December 14, 2022
Accepted: January 14, 2023
Published: January 17, 2023

Abstract

One of the critical hurdles, and breakthroughs, in the field of Natural Language Processing (NLP) in the last two decades has been the development of techniques for text representation that solves the so-called curse of dimensionality, a problem which plagues NLP in general given that the feature set for learning starts as a function of the size of the language in question, upwards of hundreds of thousands of terms typically. As such, much of the research and development in NLP in the last two decades has been in finding and optimizing solutions to this problem, to feature selection in NLP effectively. This paper looks at the development of these various techniques, leveraging a variety of statistical methods which rest on linguistic theories that were advanced in the middle of the last century, namely the *distributional hypothesis* which suggests that words that are found in similar contexts generally have similar meanings. In this survey paper we look at the development of some of the most popular of these techniques from a mathematical as well as data structure perspective, from Latent Semantic Analysis to Vector Space Models to their more modern variants which are typically referred to as *word embeddings*. In this review of algoriths such as Word2Vec, GloVe, ELMo and BERT, we explore the idea of semantic spaces more generally beyond applicability to NLP.

Keywords

Natural Language Processing, Vector Space Models, Semantic Spaces, Word Embeddings, Representation Learning, Text Vectorization, Machine Learning, Deep Learning

1. Language Processing and Machine Learning (ML)

Natural language processing (NLP) applications are ubiquitous now, perhaps the most prolific of which is Search, or *information retrieval* as it is termed in

Computer Science (CS) and Artificial Intelligence (AI) research circles. Search, arguably one of the most transformative technological advancements since the commercial adoption of the Internet in the 1990s, is an application design pattern that is rooted in NLP research and can be found in almost every application nowadays, to the point where applications are severely hindered if they have no search capability. But core NLP research and development in the last few decades has also led to technological advancements in applications such as chatbots or virtual assistants, product reviews and recommendations, document summarization and categorization, as well as spam filtering and fraud detection. All of these applications, and the underlying "intelligence" that underpins them, are based more or less upon advancements in NLP that have in many respects underpinned the technological breakthroughs that are hallmarks of the digital era. All of these applications rely on the ability of a machine to understand natural language (hence the term "natural language processing") and our ability from a research perspective to devise techniques to facilitate the development of these NLP based applications, in an unsupervised manner at massive scale, has been arguably one of the driving forces for productivity and usability, increases across the application landscape in the last decade, both for the consumer as well as the business and for the mobile as well as desktop and server sides of the application market.[1]

NLP as a discipline, from a CS or AI perspective, is defined as the tools, techniques, libraries, and algorithms that facilitate the "processing" of natural language, this is precisely where the term natural language processing comes from. But it necessary to clarify that the purpose of the vast majority of these tools and techniques are designed for *machine learning* (ML) tasks, a discipline and area of research that has transformative applicability across a wide variety of domains, not just NLP. NLP simply leverages these ML, and its successor *deep learning* (DL) capabilities in order to solve for specific NLP problems from which these core NLP applications can be developed—design patterns and algorithms that underpin NLP applications such as text summarization, keyword extraction, chatbot design and other AI applications that rest on these fundamental NLP application components.

From an ML/DL perspective, NLP is just one of many of its applications where standard input formats (*vectors*, *matrices*, *tensors*, etc.) are developed such that they can be input into advanced, highly scalable and flexible ML & DL models and frameworks, by means of which these NLP and other applications can be developed at scale. Machines of course understand numbers, or data structures of numbers, from which they can perform calculations for optimization, and in a nutshell this is what all ML and DL models expect in order for their techniques to be effective, *i.e.* for the machine to effectively *learn*, no matter what the task. NLP applications are no different from an ML and DL perspective and as such a fundamental aspect of NLP as a discipline is the collection,

[1]Perhaps the most profound example of this currently is ChatGPT by Open AI available here: https://openai.com/blog/chatgpt/.

parsing and transformation of textual (digital) input into data structures that machines can understand, a description of which is the topic of this paper (Figure 1).

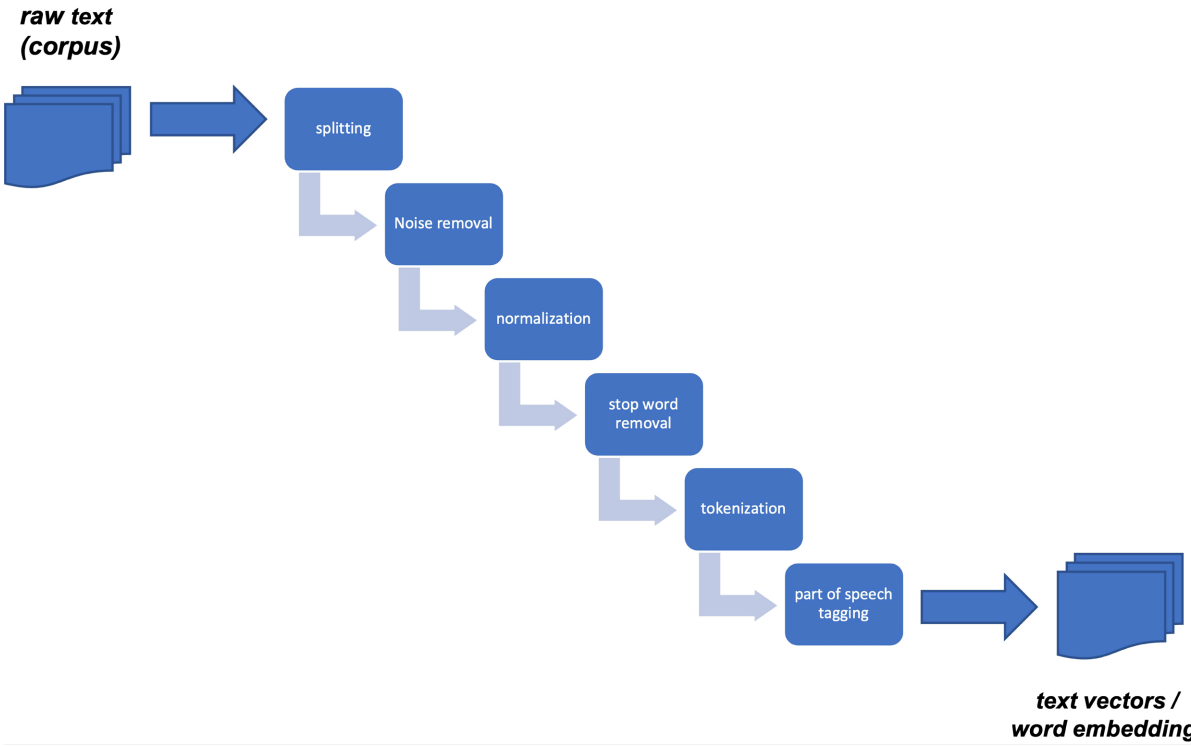

Figure 1. NLP processing workflow (aka Pipeline).

In this context, NLP can be understood from a workflow or data processing perspective which starts with raw text, or the *corpus* which is the term used in the literature, to which various transformation, normalization and standardization techniques are applied (such as parts of speech mapping, tokenization, and stop word removal) which culminate in the output of data structures that are optimized for various ML tasks, with the data structures themselves of course optimized for the specific type of NLP application being developed or researched. For example, if we are *translating* texts we need to work with words but if we are *summarizing* text we may want to work with sentences or a series of words, or *n-grams* as they are typically referred to, and each of these types of inputs require and necessitate different forms of output that are ultimately fed into the ML or DL model that is presumably doing something useful from a predictive or analytical perspective. These preprocessing, cleansing and normalization steps that are applied to the natural language, *i.e.* the text or corpus, before it can be worked with for learning are fundamental to NLP, without which machines would not be able to learn nearly as well as they do in practice. In almost all modern NLP applications, the output of this preprocessing or data transformation pipeline takes the form of vectors of real numbers that are mathematical computed based upon some fundamental linguistic (theoretical) properties called

word embeddings, the specific technological advancement that is the subject herein. We call these data structures *representations* of the text, and while they are typically constructed and associated with words, sometimes they are associated with sentences, groups of words or n-grams, or even documents or sets of sentences or phrases.

While the researchers and application developers in the NLP space leverage classic ML DL data structures and algorithms, they nonetheless—because of the nature of language itself—are confronted with specific unique challenges related specifically to the nature of language itself that they at least must be familiar with in order to apply ML and DL solutions to NLP problems. As such, terms that become not just relevant but elementary in understanding NLP algorithms and components are terms that come straight out of the field of linguistics, where *context*, *morphology*, *syntax*, and *semantics* have well defined meanings which correspond to certain data structures and/or processes within standard libraries that have been developed to solve for some of the very hard problems of NLP and which in turn now stand as the basis for future development – on the shoulders of giants and such (Figure 2).[2]

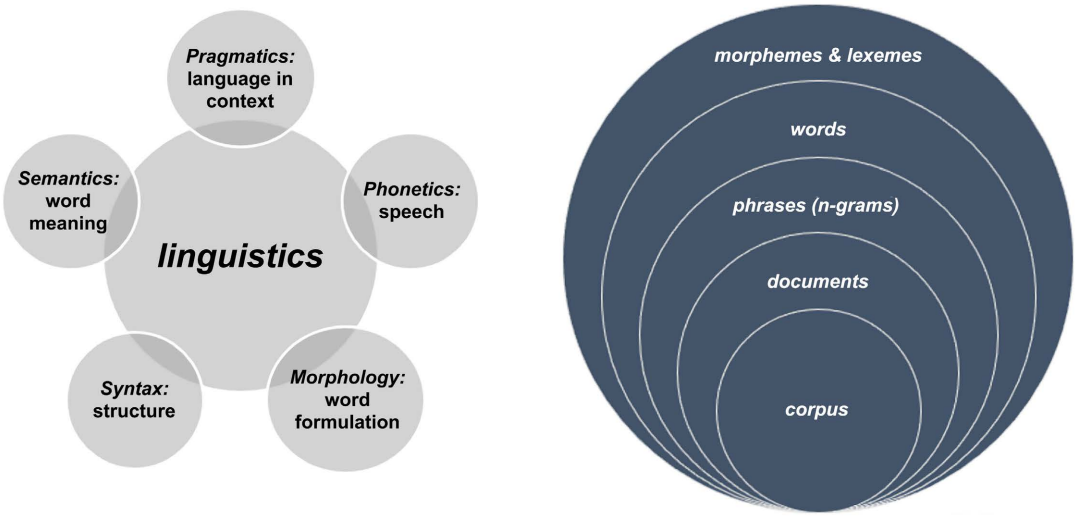

Figure 2. Linguistics and NLP.

In this world, semantics, syntax and morphology come together to determine the boundaries of various *representations* of language, language that is broken into sentences—from paragraphs and texts—which they themselves are broken into parts of speech, entity recognition and into other "parts of speech" which in turn facilitate the "understanding" of the machine to ultimately, and this is the ML part (which leverages DL design patterns), spit out optimal outcomes as defined within the context of the specific NLP application being developed—be it a support chatbot, a text summarization tool, an email spam filtering application, or one of a host of other modern day applications that are built using modern

[2]In Python for example, the most popular ML language today, we have libraries such as spaCy and NLTK which handle the bulk of these types of preprocessing and analytic tasks.

NLP technology on top of ML and/or DL models. And as it turns out, for a majority of these models, the most effective and popular format, or again representation, of language that is effective for ML and DL paradigms is *word embeddings*, a specific type of fixed dimensional data structure which encodes, via the use of sophisticated algorithms built on top of ML and DL models, semantic meaning as it related to the word, n-gram or sentence structure that it is associated with. These again mostly vectorized structures as it relates to the ML domain specifically are used not only to facilitate the "learning" process which is core to NLP within the context again of ML, but also as of course the data structures that are used to "represent" the output of the natural language processing pipeline.

2. The Curse of Dimensionality, Feature Selection, Vector Space Models and Latent Semantic Analysis

In any ML problem, one of the most critical aspects of model construction is the process of identifying the most important and salient features, or inputs, that are both necessary and sufficient for the model to be effective. This concept, referred to as *feature selection* in the AI, ML and DL literature, is true of all ML/DL based applications and NLP is most certainly no exception here. The goal of feature selection is to hone the number of features down to the most elemental level while not losing any of the granularity or color of the overall model that is reflected on the various features, or inputs, which drive the learning model in question—effectively balancing computational performance against model predictability as it were. In NLP, given that the feature set is typically the dictionary size of the vocabulary in use, this problem is very acute and as such much of the research in NLP in the last few decades has been solving for this very problem.

One of the great advancements in ML in the last two decades has been, facilitated by DL techniques underpinned by neural networks, has been the automation of the feature selection process, a process which was very manual in the early days of ML. To this end, one of the fundamental and most important parts of building NLP applications is leveraging these advancements in the feature selection process, advancements that have come mostly in the form of word embeddings which were introduced in 2013 and represent one of if not the transformative technology that has driven NLP development subsequently. Ultimately these features need to be represented as numbers, or vectors or matrices of numbers, because this is what machines can not only understand, so to speak, but also what they can compute effectively and can learn against. As such, word embeddings represent words as vectors of real numbers, where these real numbers in a very specific statistical and algorithmic sense reflect, or *represent*, the underlying text, *i.e. natural language*.[3]

This process, sometimes referred to as *text vectorization*, however can (and really should) be understood as a projection of the text (or words or *n-grams*)

[3]Python, with the *numpy* libraries in particular, is very efficient for example at working with vectors and matrices particularly when it comes to matrix math, i.e. linear algebra.

into what is referred to in the NLP (and cognitive science) literature as *semantic space*. Semantic space in this context is defined as a geometrically and algebraically defined N dimensional space where N = # of features, which again in the case of NLP ML applications is typically words, n-grams, or phrases. This space then mathematically bounds both the problem domain as well as the solution domain, and with word embeddings in particular, a modern variant of text vectorization which is the subject of this paper, the dimensions are actually learned by the model itself in order to ensure that the model exhibits certain properties, properties which are a function of what is known as the *distribution hypothesis*, more on this below (**Figure 3**).

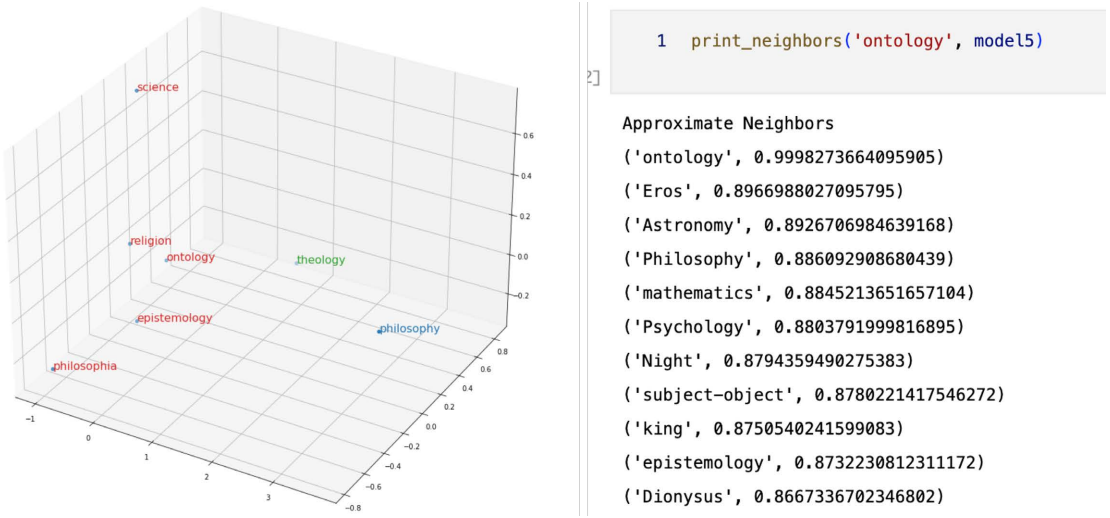

Figure 3. Semantic space and word similarities.

In classical physics we deal (mostly) with four dimensions—three in Euclidean (or Cartesian) space and a fourth dimension of time—while with ML problems we are operating in dimensional space which is defined by the set of features in our model (usually denoted by *X*), where dimension sizes of 100 or even sometimes 1000 are quite normal given how ubiquitous data is in the modern, Internet era. With the increase in dimensional complexity though comes a geometrically (if not exponentially) increase in computational resource requirements, a problem called the *curse of dimensionality* in ML and AI circles. In the NLP domain this problem is exacerbated by the fact that the feature set for some of the problem domains (like for example predictive text, chatbots, virtual assistants or language translation) can be almost the size of the language dictionary itself.[4]

This curse of dimensionality, which the NLP domain is in particular is challenged with, has driven a significant amount of research into this particular aspect of ML, *i.e. feature selection*, which is basically the process of identifying the most important, or most relevant, set of features in a given model or problem domain such that the applications that are developed can effectively run in a

[4]For a sense of scale the English language has almost 200,000 words and Chinese has almost 500,000.

reasonable time with a reasonable amount of computing resources—however reasonable might be defined for a given problem in a given industry. [What's reasonable for Google for example is not the same thing as what's reasonable for an Internet startup building a virtual assistant for a specific industry.] As such, much of the research in NLP in the last few decades, and many of the breakthroughs in terms of application performance and effectiveness (accuracy), have been in the area of encoding, or compressing, information related to various forms of language—words, phrases, n-grams, sentences or documents—to condense information, ultimately semantic information, into data structures (vectors, matrices) that machines can process efficiently, *i.e. learn* efficiently from—aka NLP feature selection. This part of NLP application development can be understood as a projection of the natural language itself into feature space, a process that is both necessary and fundamental to the solving of any and all machine learning problems and is especially significant in NLP (**Figure 4**).

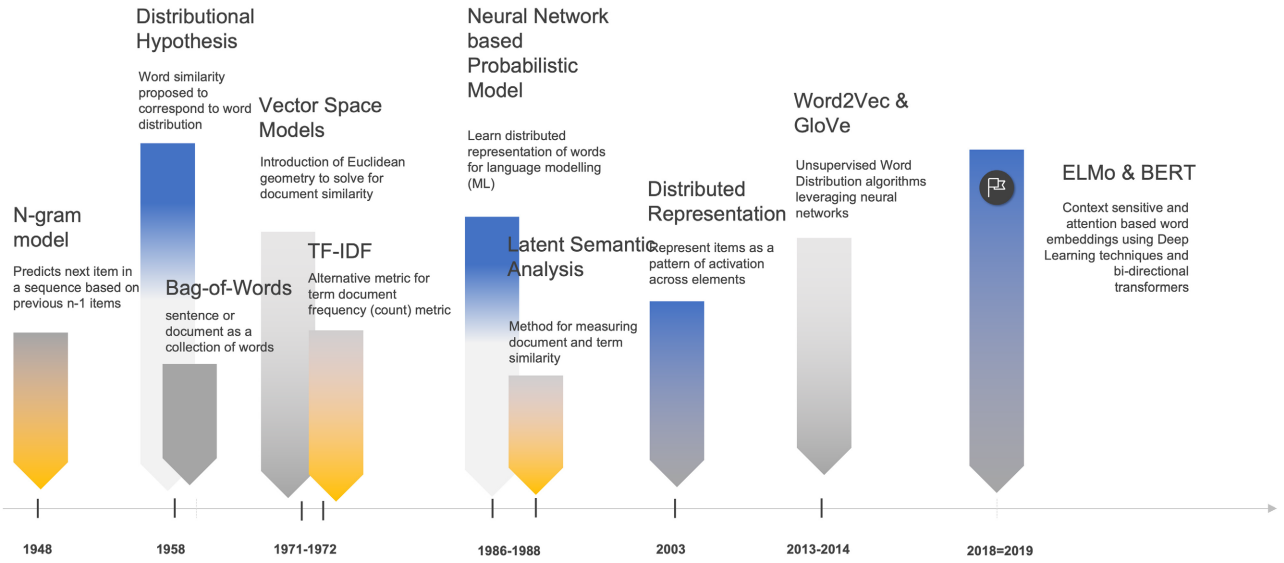

Figure 4. NLP research and development timeline.

This challenge of dealing with the sparsity of language data in vectorized form, which again is fundamental to the problem of machine learning and AI more generally, goes back to the 1980s with the advancement of techniques such as Vector Space Models (VSMs) and Latent Semantic Analysis (LSA) as well as random indexing, all techniques that were designed to convert sparse data and compress it for more efficient computation within the context of machine learning applications, *i.e.* dimension reduction or feature selection techniques. In 2000, neural networks were introduced into the problem domain to solve for some the same problems and eventually these techniques evolved into word embeddings which have the unique quality of being able to be self-generated, *i.e.* automatically generated using unsupervised learning methods [1], and which are a way of representing semantic information, or meaning, in a non-sparse way.

One of the fundamental tenets of linguistics that has been well established

since the middle of the last century is the idea that the meaning of a word can be understood by not just the context within which it is found (or heard as the case may be) but that generally, over a large corpus of text, or a large sampling of language as the case may be, that words found within a similar context should in most cases have similar meanings, leading to the now oft-used and famed expression in both linguistics and NLP circles that a word can be characterized, or understood in some sense, by the company it keeps. This is known in linguistics as well as NLP circles as the *distributional hypothesis* and it is this tenet that underpins many of the techniques described here which fall under the heading of word embeddings. It's upon this notion, the distributional hypothesis, that much of the semantic work in NLP is based, in particular as it relates to text vectorization and word embeddings, the topic of this paper. The idea being that if we can associate with a given work certain metrics that correlate to the word's distribution throughout a given text or corpus, words that have similar meanings should in turn have similar distributional properties throughout that given textual corpus (Figure 5).

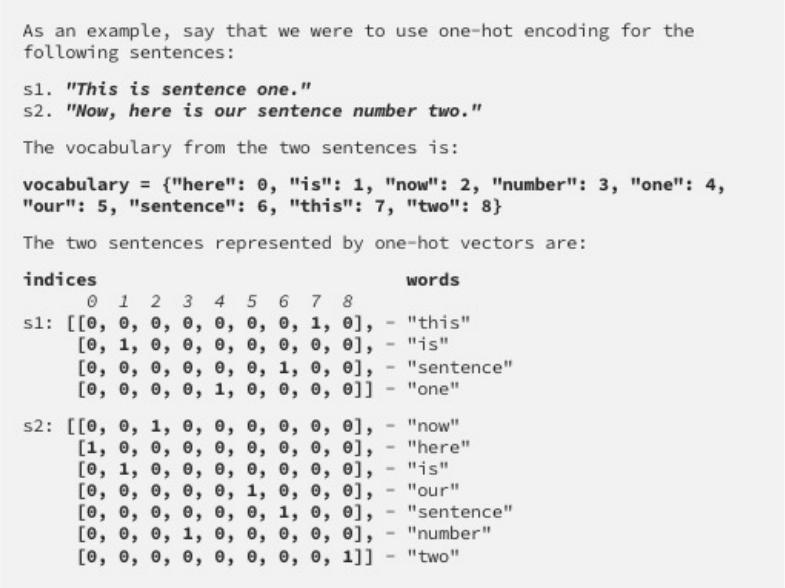

Figure 5. One hot encoding example[5]. [2]

While one-hot encoding and count vector (Bag of Words) representations of text allow for the comparison of words in sentences, documents and sets of documents, they don't store information related to the context of a given word, or sentence, within a given document, or corpus—such context is necessary for example when developing (ML/NLP) applications that solve for sentiment analysis, (machine) translation, or question-answering (chatbots or IVR) problems. These techniques also suffer from the fact that their number of dimensions (cf. *curse of dimensionality*), the semantic spatial boundaries you could call it, where

[5]Image from Hu 2020.

the number of features or dimensions of the model is a function of the overall dictionary for the corpus at hand being analyzed. This becomes very cumbersome for ML generally given that the performance of a given model is a function of the feature set that is being analyzed and in this case the feature set is the scope of words being used, with each word in the set being associated with a vector that is sized based upon the total word count (Figure 6).[6]

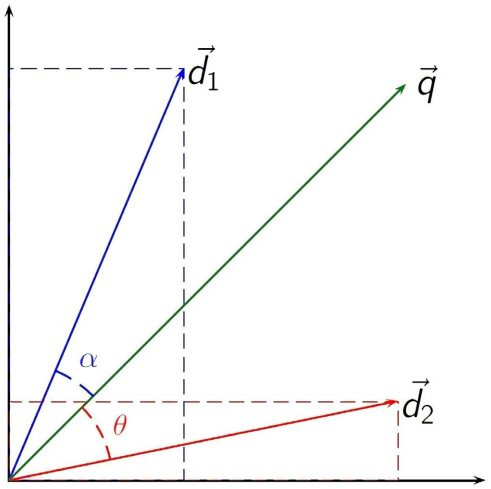

Figure 6. Document query example on vector space model[7].

An alternate method of storing word, sentence or document level information was developed in the 1970s by Gerard Smart—the father of Information Retrieval as he is sometimes called for a system called SMART [3], which was the first time that Vector Space Models (VSMs) were used to represent text such that the similarity between documents as well as queries, in a search request for example, could be computed using the cosine between the term vector and the document vector. [4] With SMART, both text documents as well as search queries were stored as vectors of term counts (TF-IDF introduced by Sparck Jones [5]) thereby allowing for the similarity of searches and documents to be both accurately and easily computed at scale.[8] A related method of using Euclidean geometrical mathematical formulations to solve for document similarity, as opposed to work similarity, was developed in the late 1980s called Latent Semantic Indexing (LSI), or Latent Semantic Analysis (LSA) within the context of words which was a later development [6]. LSI was patented in 1988 and introduced in a research paper in 1990 [7] and is also predicated on the distributional hypothesis, using linear al-

[6]While there are methods for reducing this "feature size", an elemental task in all machine learning problems (e.g., simply limiting the word count to the most used, or frequently used, top N words, or more advanced methods such as Latent Semantic Analysis), such methods are beyond the scope of this paper.

[7]By Riclas - Own work, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=9076846

[8]Latent Semantic Analysis (LSA), or Latent Semantic Indexing (LSI) as it is sometimes called in information retrieval research circles, was patented in 1988 and represents a similar statistical method to VSMs, also based upon the distributional hypothesis, which facilitates the ability to find similar documents and terms across documents. (Deerwester *et al.* 1990, Turney *et al.* 2010)

gebra techniques, specifically an operation called Singular Value Decomposition (SVD), to operate on the term-document matrix to identify similarities between documents and their associated terms [8].

3. The Distributional Hypothesis, Word Embeddings, Word2Vec and GloVe

Approaches such as VSMs or LSI/LSA are sometimes as *distributional semantics* and they cross a variety of fields and disciplines from computer science, to artificial intelligence, certainly to NLP, but also to cognitive science and even psychology. The methods, which are rooted in linguistic theory, use mathematical techniques to identify and compute similarities between linguistic terms based upon their distributional properties, with again TF-IDF as an example metric that can be leveraged for this purpose. In this context, these methods which began as VSMs combined with or supplemented by LSI or LSA, have evolved into what we call in NLP research circles today as *word embeddings*, a semantically encoded representation of words, documents or terms based upon the premise that a given word's meaning can be understood by its context, or in more vernacular terms, the company it keeps, *i.e.* what is known as the *distributional hypothesis*, an idea popularized by John Firth in the 1950s who coined the phrase "*you shall know a word by the company it keeps*" [9].

While these internal (text) representations come in a variety of forms, each form being optimized for a specific task or specific type of ML model for a given (NLP) application (and sometimes are used together to form a multi-dimensional view of the "language" that is being "processed"), generally however, given the propensity of machines to be efficient at computing numbers (scalers), vectors and matrices (tensors), they involve a conversion of text data into some vector of scalers (or reals), where in a mathematical sense the "feature space" is bound by a relevant and impactful (predictive) number of dimensions that can then be used for efficient computation. There are various methods for doing this, the most popular of which are covered in this paper—one-hot encoding, Bag of Words or Count Vectors, TF-IDF metrics, and the more modern variants developed by the big tech companies such as Word2Vec, GloVe, ELMo and BERT. Regardless of vectorization technique used, in other words which word embedding algorithm is selected, these pre-computed text vectors which encode a specific kind of semantic information can then be fed into various ML or DL models for actual NLP application development—such as again performing some sort of sentiment analysis or keyword generation, or in more complex examples like chatbots or language translation.

Some of the simplest forms of text vectorization include *one-hot encoding* and *count vectors* (or *bag of words*), techniques. These techniques simply encode a given word against a backdrop of dictionary set of words, typically using a simple count metric (number of times a word shows up in a given document for example). More advanced frequency metrics are also sometimes used however,

such that the given "relevance" for a term or word is not simply a reflection of its frequency, but its relative frequency across a corpus of documents. TF-IFD, or *term frequency-inverse document frequency*, whose mathematical formulation is provided below, is one of the most common metrics used in this capacity, with the basic count divided over the number of documents the word or phrase shows up in, scaled logarithmically.

Word embeddings evolved against the backdrop of these simpler mechanisms for *representing* terms, words or n-grams, designed specifically to answer questions about semantic similarity which is a key requirement in search applications, or Information Retrieval as it is called in research circles. For example, in a given Google search, one is trying to determine as accurately or 'closely' as possible, what documents are as closely related to the given search query that is input by a given user. This application requirement can be conceived of, from a software or NLP perspective, as solving for word, phrase or sentence *similarity*, for if you can identify which documents are most similar to your search input, you now have an effective way to rank order search results. Sound familiar? (**Figure 7**)

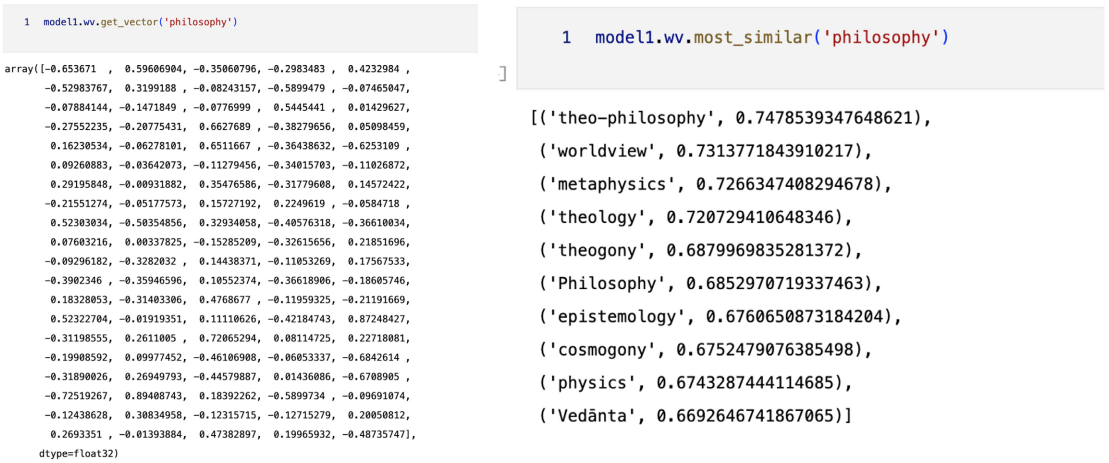

Figure 7. Word embeddings and similarity vectors.

This approach however relies on a certain (set of) linguistic theories which then in turn are formulated mathematically, or more precisely statistically, in an area of research and development that is sometimes referred to as *distributional semantics*, where similarities between linguistic terms are derived from the distributional properties of the underlying terms. In this context, word embeddings can be understood as *semantic representations* of a given word or term in a given textual corpus. At their core then, word embeddings can be understood as a set of mathematically (statistically) generated (real) numbers, attributes or features within the context of ML, that are created from a body of text which map a given word into a semantic space for a given domain which follows from the *distribution hypothesis*, namely that words that are found in similar contexts have similar meanings, such that word, phrase or even document similarity can

be efficiently computed. *Semantic spaces* are the geometric structures within which these problems can be efficiently solved for.

One of the major breakthroughs in word embedding technology came in 2013 when a team at Google led by Tomas Mikolov created Word2Vec [10] [11], a word embedding solution that uses (unsupervised) deep learning techniques to capture word associations from a large corpus of text leveraging both (continuous) bag of words as well as skip gram approaches to defining word context (semantics). When setting up the word embedding structure, we tell the model only how big we want the vector to be for each word (maybe between 50, 100, 200 or 300 for example), and let the model learn all the weights as well as the associated terms for said work, or token, again given context. As a result of such modeling, we can identify for example, words of similar meaning by looking at how close they are in the underlying vector space of words that is generated from the given text.

There are two ways which the algorithm can be coded to learn the vector feature, or parameter values, one based on what is called the continuous bag of words model and another based on a continuous skip gram model, with the former method learning the embedding by predicting the current word based on context (or place) within the text, and the latter model predicting surrounding words based on a given current word. Both methods contextualize a given word that is being analyzed by using this notion of a sliding window, which is a fancy term that specifies the number of words to look at when performing a calculation basically. The size of the window however, has a significant effect on the overall model as measured in which words are deemed most "similar", *i.e.* closer in the defined vector space. Larger sliding windows produce more topical, or subject based, contextual spaces whereas smaller windows produce more functional, or syntactical word similarities—as one might expect (**Figure 8**).

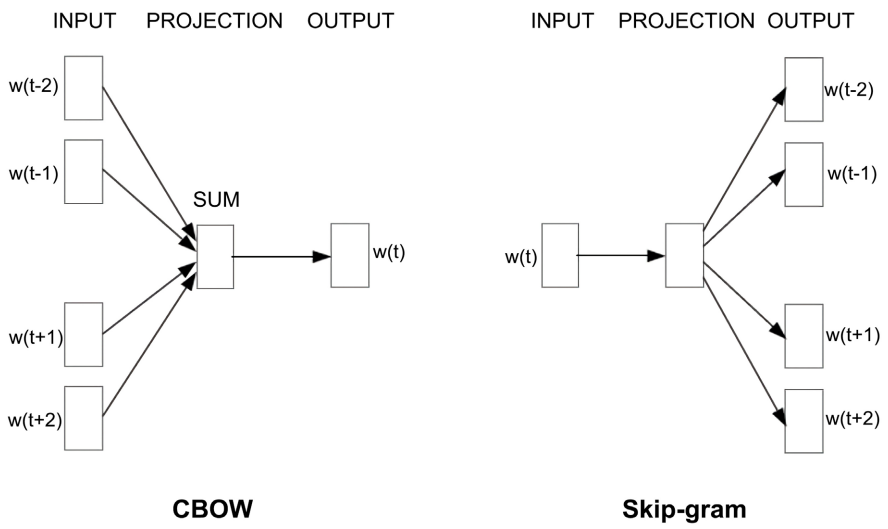

Figure 8. Word2Vec design architecture[9].

[9]Image from Mikolov *et al.* 2013.1, still the best description of the way Word2Vec constructs its word embeddings using one input layer, one hidden layer and one output layer.

Word2Vec is trained on the Google News Dataset on about 100 billion words and supports both word similarity as well as word prediction capabilities and as such has applicability in a variety of NLP applications such as Recommendation Engines, Knowledge Discovery (Search), as well as Text Classification problems. Perhaps the most revolutionary advancement that Word2Vec supports however, as do other word embedding libraries today, is the ability to pre-compute, store and re-use word embeddings offline, an engineering design technique that has come to be known as *pre-trained word embeddings*, which irrespective of the underlying algorithm that is used to generate the word embeddings themselves, has more so than anything else perhaps accelerated the pace of NLP innovation.

An alternative, unsupervised learning algorithm for constructing word embeddings was introduced in 2014 out of Stanford's Computer Science department [12] called GloVe, or *Global Vectors for Word Representation.* While GloVe uses the same idea of compressing and encoding semantic information into a fixed dimensional (text) vector, *i.e.* word embeddings as we define them here, it uses a very different algorithm and training method than Word2Vec to compute the embeddings themselves. The GloVe model is predicated on the idea that word co-occurrence represents semantic information just as word distribution does and as such it creates a massive word context co-occurrence matrix that it uses as the basis for its training, looking to optimize its factorization against both a word-feature matrix and a feature-context matrix, optimizing via an objective function (loss function) that aims to learn word embeddings such that their dot product equals the logarithm of the word's probability of co-occurrence (**Figure 9**).

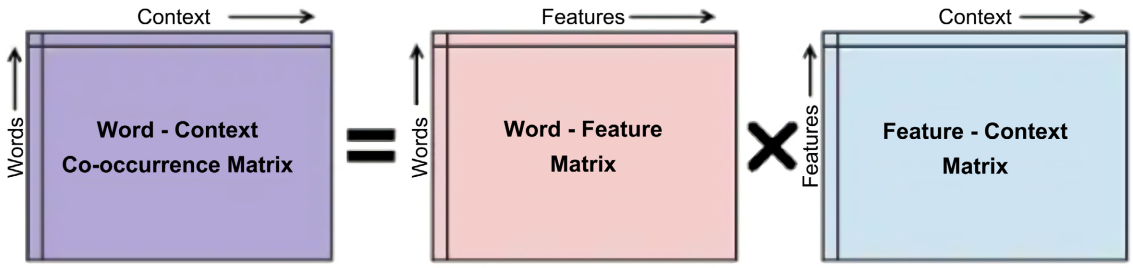

Figure 9. GloVe embeddings statistical co-occurrence design[10]. [13]

From the 2014 GloVe paper itself, the algorithm is described as "…*essentially a log-bilinear model with a weighted least-squares objective. The main intuition underlying the model is the simple observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning*." Given that the GloVe algorithm is trained using global word to word co-occurrence matrices, it encodes more specificity regarding word association than Word2Vec because it uses word co-occurrence probabilities across an entire corpus to compute the embeddings, thereby encoding an extra layer of se-

[10]Image from Sarkar 2018, probably the best visual description of the GloVe algorithm to date.

mantic information (meaning), allowing for greater performance and efficacy across a range of NLP tasks.

Having said that, it's important to understand that both Word2Vec and GloVe construct a semantic space that leverages the position of a given word within the context of the textual corpus itself as the basis for its semantic spatial geometry so to speak, both relying on the distributional hypothesis but using different statistical techniques to encode the notion of "distribution" as it were—Word2Vec being a more localized version, parametrized by its window size, and GloVe looking across the entire corpus at co-occurrence metrics. Furthermore both Word2Vec and GloVe word embeddings, despite their differences, nonetheless share the basic characteristics of modern word embeddings that make them powerful in an NLP context, namely that they 1) solve for the curse of dimensionality problem by minimizing the number of dimensions, or features, associated with a given NLP term, word or n-gram, and 2) can be pre-trained for use in a wide variety of applications without having to incur the cost of training for each application, the latter being a radical transformative step for the rate of NLP application development overall.

4. Neural Networks, Deep Learning, the Attention Mechanism and Transformers (ELMo and BERT)

One of the linguistic aspects, or levels of granularity, missing from the simple count vectorization as well as even the more advanced Word2Vec and GloVe word embedding techniques however is that despite their sophistication and ease of use, each word in these models has one and only one representation, a limitation given that one of the distinctive characteristics of language is that words in different contexts can have widely varying meanings, a property called *polysemy* in linguistics. To address this, more advanced, bi-directional Deep Learning techniques have been developed that allow both the local and global context of a given word (or term) to be taken into account when generating embeddings, thereby addressing some of the shortcomings of the Word2Vec and GloVe frameworks.

The most popular of these types of approaches that have been recently developed are ELMo, short for Embeddings from Language Models [14], and BERT, or Bidirectional Encoder Representations from Transformers [15]. These word embedding techniques are more advanced than their predecessors in terms of encoding more information and as such have for the most part supplanted Word2Vec and GloVe in more advanced NLP applications—given the fact that they are also pre-trained on massive textual data sets, with no increase in overhead to the applications themselves and can in most cases, because it generates the same output data structure as their predecessors, *i.e.* word embeddings, can be basically swapped into existing applications with very little development cost.

ELMo was released by researchers from the Allen Institute for AI (now AllenNLP) and the University of Washington in 2018 [14]. ELMo uses character level encoding and a bi-directional LSTM (long short-term memory) a type of

recurrent neural network (RNN) which produces both local and global context aware word embeddings. While forward language models compute the probability of a word given previous words and a backward language model does the opposite, ELMo leverages the bi-directional LSTM architecture to jointly maximize the likelihood of both the forward and backward words (characters actually in the case of ELMo), yielding better *sense disambiguation* support, addressing the *polysemy* issue where word meanings are contextualized and not fixed across the entire corpus (**Figure 10**) [16].

$$L = \sum_{t=1}^{N} (\log P(w_t | w_1, w_2, ..., w_{t-1}) + \log(P(w_t | w_{t+1}, w_{t+2}, ..., w_N)).$$

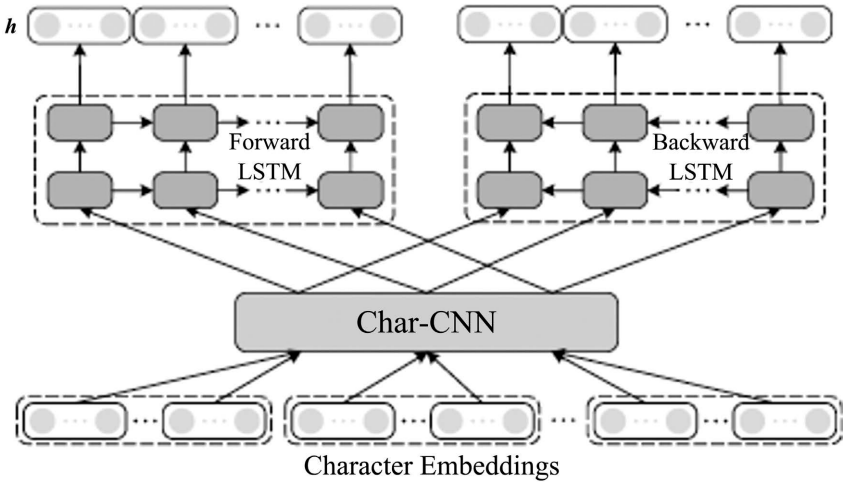

Figure 10. ELMo objective function neural network (bidirectional LSTM) architecture[11].

ELMo therefore represents a breakthrough in word embedding architecture given the additional semantic information it encodes as a result of its more sophisticated use of DL paradigms to effectively encode semantic information from two directions simultaneously, resulting in better performance versus the older word embedding techniques across a wide array of NLP problems such as question answering, named entity extraction, and sentiment analysis to name a few [14] [16]. ELMo also has the unique characteristic that, given that it uses character-based tokens rather than word or phrase based, it can also even recognize new words from text which the older models could not, solving what is known as the *out of vocabulary problem* (OOV).

A further advancement in word embedding technology was made in 2018 when Google labs developed what is today arguably the most advanced word embedding language model/embedding framework based upon a new DL architectural framework called *transformers* [17], a technology that had been developed a year prior that was designed and optimized specifically for language

[11]Objective function and CNN architecture diagram combined here are from Wang *et al.* [16].

translation and that further extended the capabilities of prior DL constructed word embedding frameworks such as ELMo. Transformers are based upon encoder and decoder based neural network architecture that are specific designed to solve sequence-to-sequence based tasks, as are prevalent in NLP applications and design patterns like language translation for example, while supporting long range dependencies, the latter requirement representing one of the main breakthroughs of the technology as prior technologies, such as RNNs or CNNs, had limitations in terms of context size given the underlying neural network architecture.

As the 2017 paper title which introduced transformers indicates—*Attention is All You Need*—transformer architecture is predicated on the notion of *self-attention*, or simply the *attention mechanism* [18], which is the term used to describe the way in which transformers encode and decode sequences with no constraints on the size of the given context which is used to facilitate the transformation from inputs to outputs. In other words, with transformer encode-decode architecture, while the model focuses on a specific part of a given sequence, like a word or n-gram for example in an NLP context, the model still nonetheless has access to the entire textual input if necessary as context in order to facilitate the next prediction, or output, in the sequence, removing constraints on sequence context effectively (**Figure 11**).

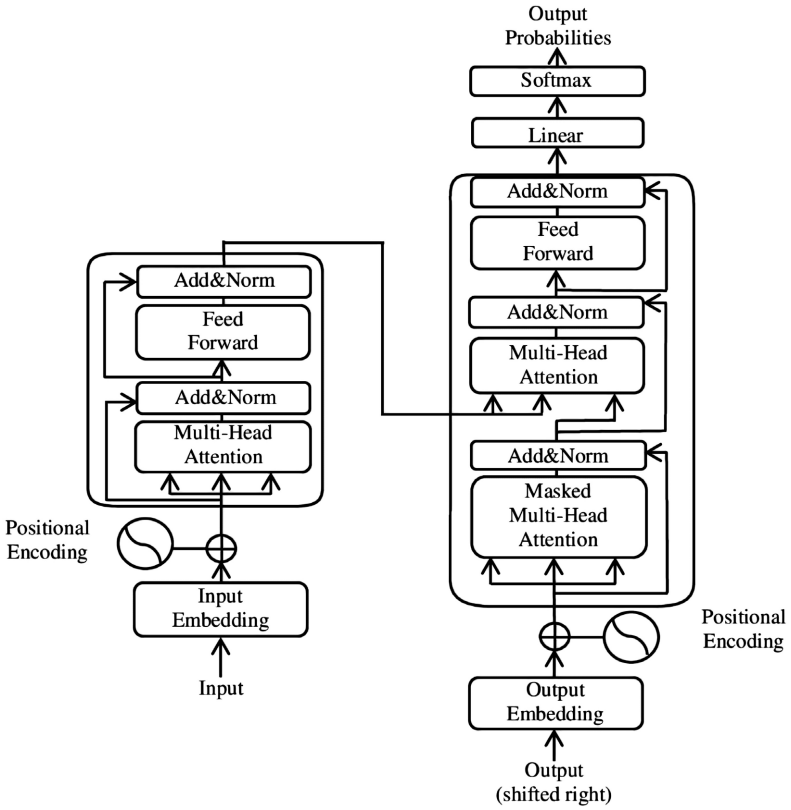

Figure 11. Transformer model architecture[12].

[12]By Yuening Jia - DOI:10.1088/1742-6596/1314/1/012186, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=121340680

BERT was developed at Google and was released in 2019 in a paper entitled *BERT*: *Pre-training of Deep Bidirectional Transformers for Language Understanding*, and, as the title refers to, leverages a transformer-based architecture, and the attention mechanism to achieve state of the art performance on a wide range of NLP tasks such as chatbots, question and answering capabilities, natural language inference capability, classification and named entity recognition tasks and more, leveraging stronger contextualized textual representations than any of its predecessors. BERT leverages this transformer based attention mechanism which emphasizes or enhances some parts of the text while de-emphasizing others using weighting techniques which are common in deep learning applications and also has the unique property of *bidirectionality*, which means that it can process, *i.e.* read text, in both directions (front to back and back to front or right to left and left to right as the case may be) at the same time, greatly increasing training performance compute times and resource requirements.

The model itself uses two novel training techniques together to develop a robust set of embeddings associated with words that goes well beyond simple text based vectorization techniques that we have discussed prior—the first is *masked language model* (MLM), which is designed to mask words and have the model predict them randomly throughout the sequence, and the other is *next sequence prediction* (NSP) which predicts whether or not a given sentence A will follow a given sentence B, the two methods being optimized for jointly working in parallel to provide a sort of multi-dimensional semantic view of the text being processed. [15]. It also is designed to come in pre-trained formats, **BERT_large** which comes with over 345 million parameters or **BERT_base** which has just over 110 million, both of which are designed to be integrated into NLP applications as is (pre-trained) and yet still allow for further customization of the embeddings to support a given task at hand (**Figure 12**).

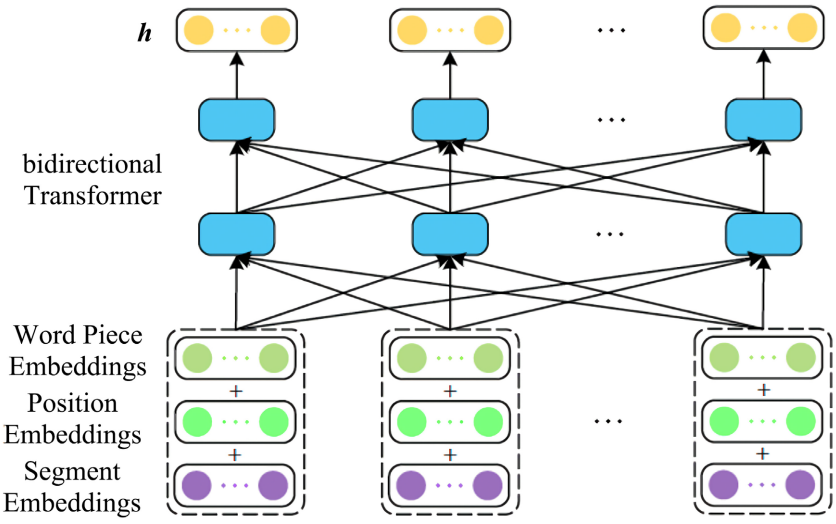

Figure 12. Bidirectional encoder representation from transformers architecture (BERT)[13].

[13]Image from Wang *et al.* 2020.

So with both ELMo and BERT computed word (token) embeddings then, each embedding contains information not only about the specific word itself, but also the sentence within which it is found as well as context related to the corpus (language) as a whole. As such, with these advanced forms of word embeddings, we can solve the problem of *polysemy* as well as provide more context-based information for a given word which is very useful for semantic analysis and has a wide variety of applications in NLP. These methods of word embedding creation take full advantage of modern, DL architectures and techniques to encode both local as well as global contexts for words. It is in fact these more modern DL architectures which have facilitated the development of these more effective and performant word embedding architectures, paving the way for the success of modern language models such as BERT (designed for language modeling and next word or sentence prediction by Google and used in their search engine) as well as GPT, another deep learning, pre-trained language model used for text and language prediction (predictive text).

5. Summary: The Distributional Hypothesis and Semantic Spaces, the Theoretical Foundations of NLP

One of the fundamental theoretical underpinnings that has driven research and development in NLP since the middle of the last century has been the *distributional hypothesis*, the idea that words that are found in similar contexts are roughly similar from a semantic (meaning) perspective. It's upon this theoretical foundation that first Vector Space Models were developed in the 1970s, Latent Semantic Indexing/Analysis was invented in the 1980s and 1990s, and represents the theoretical foundations of more modern advanced text vectorization techniques which leverage neural networks in order to encode semantic information with lexical structures such as characters, words, n-grams, or even sentences documents such as Word2Vec, GloVe, ELMo and BERT amongst others.

The effectiveness of these developments, such as Word2Vec or GloVe embeddings, or their more modern and sophisticated counterparts ELMo or BERT, is predicated on the notion that a word's (or character, phrase, word, n-gram or sentence as the case may be) *distribution properties* throughout a given textual corpus, in a very utilitarian and scientific sense, semantically define the word, or term, itself. In other words (pun intended), a statistically driven distributionally semantic based approach to text modeling is a scientifically effective strategy for efficiently solving some of the most intractable problems that underpin application development in the NLP domain today—with applicability from information retrieval or search, to document classification and sentiment analysis, to text summarization and annotation problems, and even text generation applications like language translation or predictive text [19]. Each of these applications in one way or another needs to understand the semantic spatial relationship between and among all of the different component parts of a given textual corpus in order to be effective, and as such word embeddings, and the semantic space to

which they belong, become an integral part of the NLP/ML pipeline. As such, even the more classic versions of word embeddings such as Word2Vec and GloVe provide a way to automatically generate and encode semantic information into a fixed, N dimensional space by means of being trained across a massive digital textual corpus (corpi) just once, this notion of *pre-training* perhaps being one of the most revolutionary developments in NLP in the last two to three decades.

While word embeddings are simply vector representations of text represented by a fixed set of real numbers (dimensions or features), in modern variants the numerical attributes contained within these word vectors contain highly sophisticated encoded information about the context, the meaning, of a given word in said linguistic context, a context bounded by the dimensional attributes of the semantic space which is encoded into each and every word embedding. From this semantic location as it were, we can both evaluate how similar said word or term is to any other word located in the semantic space, as well as identify similar words in said space, by simply measuring the distance between the two words in the space itself using the angle difference between the two words as a function of the origin of the space. Furthermore, once calculated, these (pre-computed) word embeddings can be re-used by other applications, greatly improving the innovation and accuracy, effectiveness, of NLP models across the application landscape.

These *word embedding* algorithms as they are now called, all rest on sophisticated mathematical and statistical formulations which take advantage of the state of the art methods in Deep Learning such as Long Short Term Memory architectures and Transformers which leverage this notion of *attention* along with local as well as global context dependencies, the bulk of which are open-sourced and available in pre-trained formats so as to facilitate their integration into a wide range of NLP tasks and applications, represent the most advanced method we know today to leverage semantic information to perform learning on natural language, one of the most fundamental technologies that has driven technology innovation in the Digital Era. And while these word embeddings are critical to the effectiveness of a broad range of NLP applications, leveraging this notion of Semantic Spaces which was developed almost 50 years ago, nonetheless do not have much applicability beyond the NLP and ML/DL fields as their meaning (pun intended) is really only derived in the sense of their utility in identifying word and term (and document) similarity rather than denoting something intelligible so to speak by anyone, or anything, other than a machine.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Hinton, G. and Roweis, S. (2002) Stochastic Neighbor Embedding. In: Becker, S.,

Thrun, S. and Obermayer, K., Eds., *Advances in Neural Information Processing Systems* 15 (*NIPS* 2002), The MIT Press, Cambridge.
https://cs.nyu.edu/~roweis/papers/sne_final.pdf

[2] Hu, J. (2020) An Overview of Text Representations in NLP. Towards Data Science.
https://towardsdatascience.com/an-overview-for-text-representations-in-nlp-311253730af1

[3] Salton, G. (1971) The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice-Hall, Hoboken.

[4] Salton, G., Wong, A. and Yang, C.S. (1975) A Vector Space Model for Automatic Indexing. *Communications of the ACM*, **18**, 613-620.
https://doi.org/10.1145/361219.361220

[5] Sparck Jones, K. (1972) A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation*, **28**, 11-21.
https://doi.org/10.1108/eb026526

[6] Landauer, T.K. and Dumais, S.T. (1997). A Solution to Plato's Problem: The Latent Semantic Analysis Theory of the Acquisition, Induction, and Representation of Knowledge. *Psychological Review*, **104**, 211-240.
https://doi.org/10.1037/0033-295X.104.2.211

[7] Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W. and Harshman, R.A. (1990) Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science* (*JASIS*), **41**, 391-407.
https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9

[8] Turney, P. and Pantel, P. (2010) From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, **37**, 141-188.
https://doi.org/10.1613/jair.2934

[9] Firth, J.R. (1957) Studies in Linguistic Analysis. Blackwell, Oxford.

[10] Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013) Efficient Estimation of Word Representations in Vector Space. ArXiv: 1301.3781.

[11] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J. (2013) Distributed Representations of Words and Phrases and Their Compositionality. In: Burges, C.J., Bottou, L., Welling, M., Ghahramani, Z. and Weinberger, K.Q., Eds., *Advances in Neural Information Processing Systems* 26, Curran Associates, Inc., Red Hook, 3111-3119.

[12] Pennington, J., Socher, R. and Manning C. (2014) GloVe: Global Vectors for Word Representation. *Proceedings of the* 2014 *Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), Doha, 25-29 October 2014, 1532-1543.

[13] Sarkar, D. (2018) A Hands-on Intuitive Approach to Deep Learning Methods for Text Data—Word2Vec, GloVe and FastText. Towards Data Science.
https://towardsdatascience.com/understanding-feature-engineering-part-4-deep-learning-methods-for-text-data-96c44370bbfa

[14] Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L. (2018) Deep Contextualized Word Representations. In: Walker, M., Ji, H. and Stent, A., Eds., *Proceedings of the* 2018 *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, *Volume* 1 (*Long Papers*), Association for Computational Linguistics, New Orleans, 2227-2237. https://doi.org/10.18653/v1/N18-1202

[15] Devlin, J, Chang, M.-W., Lee, K. and Toutanova, K. (2019) BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In: Burstein, J., Doran, C. and Solorio, T., Eds., *Proceedings of the* 2019 *Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, *Volume* 1 (*Long and Short Papers*), Association for Computational Linguistics, Minneapolis, 4171-4186.

[16] Wang, Y., Hou, Y., Che, W. and Liu, T. (2020) From Static to Dynamic Word Representations: A Survey. *International Journal of Machine Learning and Cybernetics*, **11**, 1611-1630. https://doi.org/10.1007/s13042-020-01069-8

[17] Vaswani, A., Shazeer, N., Parmer, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L. and Polosukhin, I. (2017) Attention Is All You Need. ArXiv: 1706.03762.

[18] Galassi, A., Lippi, M. and Torroni, P. (2021) Attention in Natural Language Processing. *IEEE Transactions on Neural Networks and Learning Systems*, **32**, 4291-4308. https://doi.org/10.1109/TNNLS.2020.3019893

[19] Koroteev, M.V. (2021) BERT: A Review of Applications in Natural Language Processing and Understanding. ArXiv: 2103.11943.