Human language is excessively effective at describing how we relate to the world. With a few, short words, many ideas and actions with little ambiguity are conveyed. A word is a basic unit of language that conveys meaning of its own. It is no easy task for a computer to learn natural language, for it requires understanding how humans observe the world, if not understanding how to observe the world. With the help of words and language rules, an infinite set of concepts can be expressed. Computer programs are still lines of instructions telling a computer what to do — they often miss nuance and context.

The vast amounts of scientific knowledge are published as text, which is difficult to analyze by either traditional statistical analysis or modern machine learning methods. The ayurvedic corpora from the classical texts and various pharmacopoeias are examples of large scientific knowledge meeting these criteria. The traditional statistical analysis is mostly based on the structured data.  One method of scientifically exploring human understanding is to examine what we write. Researchers have made great advances in the area of computational methods for extracting meaning from text. Large texts combined with computational techniques for analyzing these texts allow scientists to extract meaning, patterns and knowledge from unstructured text data. The science of extracting meaning and learning from text data is called Natural Language Processing (NLP) which has its roots in computational linguistics.

NLP is carried out under the assumption that there is meaning in text and that meaning resides in the words, sentences, paragraphs, and so on. The corpus is not an ordinary bag of words, the most obvious aspect of a corpus is its distributional nature. Words in a corpus tend to concentrate in certain context and tend to co-occur with other specific words. Can the full extent of meaning be obtained exclusively from the words and their co-occurrences or is more context and information needed, such as syntax or human-derived data?

The core concept of word embeddings is that every word used in a language can be represented by a set of real numbers (a vector).Word embeddings are distributed representations of text in an n-dimensional space. There are a few key characteristics to a set of useful word embeddings:

* Every word has a unique word embedding (or “vector”), which is just a list of numbers for each word.
* The word embeddings are multidimensional; typically for a good model, embeddings are between 50 and 500 in length.
* For each word, the embedding captures the “meaning” of the word.
* Similar words end up with similar embedding values.

Types of embeddings:

One hot encoding

These are essential for solving most NLP problems. Word embeddings are computational implementation of the DistributionalHypothesis.NLP is not a single method but a collective term for any machine learning methods. The starting point is to build a text corpus, data cleaning as necessary for the algorithm. Based on the insights, the text is represented using relevant feature engineering techniques. Depending on the problem, the focus is on building predictive supervised models or unsupervised models. Finally, the model is evaluated and the overall success criteria with researchers, and deploy the final model for future usage.

The available data can be classified into 2 types: unlabeled and labelled. Usually, unlabeled data consists of samples of natural or human-created artefacts that anyone can obtain from the world. Some examples of unlabeled data are photos, audio recordings, videos, news articles, tweets, x-rays, etc. There is no “explanation” for each piece of unlabeled data -- it just contains the data, and nothing else. Labelled data normally takes a set of unlabeled data and adds each member of that unlabeled data with some sort of meaningful “tag”, “label” or “class” that is explanatory or appropriate to know. Labels for data are often obtained by asking humans or experts to make judgments about a given piece of unlabeled data and are significantly more expensive to obtain than the raw unlabeled data. After obtaining a labelled dataset, machine learning models can be applied to the data so that new unlabeled data can be presented to the model and a likely label can be guessed or predicted for that piece of unlabeled data.

Based on the 2 types of data, there are supervised and unsupervised learning algorithms. In a supervised learning model, the algorithm learns on a labelled dataset, providing an answer key that the algorithm can use to evaluate its accuracy on training data. Supervised learning problems can be grouped into Regression and Classification problems. Both problems have as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for regression and categorical for classification.

An unsupervised model, in contrast, provides unlabeled data that the algorithm tries to make sense of by extracting features and patterns on its own. As no labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning). It would help in clustering, anomaly detection, and association. As there is no “ground truth” element to the data, it’s difficult to measure the accuracy of an algorithm trained with unsupervised learning.

How to represent words?

To start off, we need to be able to represent words as input to our Machine Learning models. One mathematical way of representing words is as vectors. There are an estimated 13 million words in English language. But many of these are related. Spouse to partner, hotel to motel. So do we want separate vectors for all 13 million words? No. We must search for a N-dimensional vector space (where N << 13 million) that is sufficient to encode all semantics in our language. We need to have a sense of similarity and difference between words. We can exploit concept of vectors and distances between them (Cosine, Euclidean etc. ) to find similarities and differences between words.

How do we represent meaning of words?

If we use separate vectors for all 13 million words (or maybe more) in English vocabulary, we’ll be facing several problems. Firstly, we’ll have large vectors with a lot of ‘zeroes’ and one ‘one’ (in different position representing a different word). This is also known as one-hot encoding. Secondly, when we search for phrases such as “hotels in New Jersey” in Google, we want results pertaining to “motel”, “lodging”, “accommodation” in New Jersey returned as well. And if we are using one-hot encoding, these words have no natural notion of similarity. Ideally, we would want dot products (since we are dealing with vectors) of synonym / similar words to be close to one.

Distributional similarity based representations:

You shall know a word by the company it keeps — J. R. Firth

In very simple layman terms, let’s take a word ‘bank’. One of many meanings of this word is a financial institution and another one is land alongside a body of water. If in a sentence, bank occurs with neighboring words as money, government, treasury, interest rates etc. we can understand it’s the former meaning. Contrarily, if neighboring words are water, shore, river, land etc. the case is latter. We can exploit this concept to deal with polysemy and synonyms and make our model learn.

Words are simply discrete states like any other data e.g. genes, social media, etc. and we are simply looking for the transitional probabilities between those states: the likelihood that they will co-occur. The purpose and usefulness of word embedding is to group the vectors of similar words together in vector space, i.e. it detects similarities mathematically. Embeddings create vectors that are distributed numerical representations of word features, features such as the context of individual words. It does so without human intervention. Given enough data, usage and contexts, NLP algorithms can make highly accurate guesses about a word’s meaning based on past appearances. Those guesses can be used to establish a word’s association with other words (e.g. “man” is to “boy” what “woman” is to “girl”), or cluster documents and classify them by topic. Those clusters can form the basis of search, sentiment analysis and recommendations in such diverse fields as scientific research, legal discovery, e-commerce and customer relationship management.

<https://towardsdatascience.com/word-to-vectors-natural-language-processing-b253dd0b0817>

<https://arxiv.org/pdf/1902.06006.pdf>

Dimensionality refers to how many attributes a dataset has. E.g. if the input text has 20,000 unique words, then this corpus of text would have dimension of 20,000, each word = 1 dimension. The dimension of the data is the number of variables that are measured on each observation. High-dimensional datasets present many mathematical challenges. One of the problems with high-dimensional datasets is that, in many cases, not all the measured variables are important for understanding the underlying phenomena of interest.

There’s no obvious way to suitably compare two words unless we already know what they mean. The goal of word-embedding algorithms is to embed words with meaning based on their relationship with other words. Words are embedded into a real vector space, which comes with notions of distance and angle. We hope that these notions extend to the embedded words in meaningful ways, quantifying relations between different words. And as further evidence that a word’s meaning can be implied from its relationships with other words, they actually found that the learned structure for one language often correlated to that of another language, perhaps suggesting the possibility for machine translation through word embeddings (Mikolov 2013c)

Word embeddings transform human language meaningfully into a form conducive to numerical analysis. In doing so, they allow computers to explore the wealth of knowledge encoded implicitly into our own ways of speaking.

And let’s not get into any philosophical considerations of whether the computer really understands the word. Come to think of it, how do I even know you understand a word of what I'm saying? Maybe it's just a matter of serendipity that the strings of words I write make sense to you. But here I am really talking about how to oil paint clouds and you think that I'm talking about machine learning.

Consider a 20-word context. If we assume that the average English speaker's vocabulary is 25,000 words, then the increase of 1 word corresponds to an increase of about 7.2×1084 contexts, which is actually more than the number of atoms in the universe. Of course, most of those contexts wouldn't make any sense.

The algorithm used by the Google researchers mentioned above assumes 300 features.

The term distributed representation of words comes from this: we can now represent words by their features, which are shared (i.e. distributed), across all words. We can imagine the representation as a feature vector. For example, it might have a 'noun bit' that would be set to 1 for nouns and 0 for everything else. This is, however, a bit simplified. Features can take on a spectrum of values, in particular, any real value. So, feature vectors are actually vectors in a real vector space.

The distributed representations of words “allows each training sentence to inform the model about an exponential number of semantically neighboring sentences”, (Bengio 2003).

The softmax function is often chosen as the ideal probability distribution. One can control the algorithm by specifying different hyper parameters: do we care about order of words? How many surrounding words do we consider?

Word embeddings are a recent addition to an NLP researcher’s toolkit. They are dense, real-valued vector representations of words that capture interesting properties among them. Word embeddings are learned from raw corpora. Usually, the larger the corpora, the better are the quality of the embeddings learned. However, the larger the corpora, the larger are the amount of resources and time needed for their training. Thus, different groups release their learned embeddings publicly. Such pre-trained embeddings is a primary reason for the inclusion of word embeddings in mainstream NLP. However, such pre-trained embeddings are usually learned on generic corpora. Using such embeddings in a particular domain such as medical domain leads to following problems:

* No embeddings for domain-specific words. For example, phenacetin is not present in pretrained vectors released by Google.
* Even those words that do have embeddings may have a poor quality of the embedding, due to different senses of the words, some of which belonging to different domains.

It is difficult to obtain large amounts of domain specific data. However, many NLP applications have benefited from the addition of information from small domain-specific corpus to that obtained from a large generic corpus (Ito et al., 1997).

* You open Google and search for a news article about the ongoing women’s T20 world cup and get a lot of search results about it.
* You type a sentence in Google translate in English and get an Equivalent translation in many major languages.
* You type about the corona virus scare which originated at the end of 2019 and get onslaught of reliable as well as fake news.

 All of this has been made possible by massive Google search engine at the back end, crunching complicated text data every second. The answer to the above questions lie in creating a representation for words that capture their meanings, semantic relationships and the different types of contexts they are used in. And all of these are implemented by using Word Embeddings or numerical representations of texts so that computers may handle them.

<https://graphics.cs.wisc.edu/Papers/2018/HG18/embeddings_preprint.pdf>

1. Word embedding in Natural Language Processing:

Natural Language Processing (NLP) refers to computer systems designed to understand human language. NLP attempts to extract information from sentences [NLP 1]. The data contained in EMR for diseases and medicines for each patient is converted to a sentence. The collective sentences are presented to the NLP algorithm as a source text. This data is not smoothly and uniformly distributed. E.g., many medicines are prescribed very few times; many diseases are rarely reported. Such textual data may be very sparse and may have high dimensions. High Dimensional means that the numbers of dimensions are so high that calculations become extremely difficult. With high dimensional data, the number of features can exceed the number of observations [High dimension 1]. The more dimensions are present in a data the more difficult it becomes to predict certain quantities. Number of diseases and number of medicines identified through EMRs created dimensions for us in our ongoing analysis. Visual exploration of such data becomes impossible. However, it is essential to understand the data that can be achieved using dimensionality reduction techniques.

Word embedding techniques and visual exploration techniques were employed.

Word embeddings is one the many approaches used to extract information from a large corpus. The term word embeddings was originally coined by Bengio et al. in 2003 who trained them in a neural language model together with the model’s parameters. Their architecture forms very much the prototype upon which current approaches have gradually improved. It is one of the most popular representations of document vocabulary. They are distributed representations of text in an n-dimensional space. They have proven beneficial for a wide variety of language processing tasks [word embedding 1]. They are a form of word representation that bridges the human understanding of language to that of a machine [word embedding 2].

Word Embedding is used in Predictive modelling based on natural language processing. The working of word embedding relies on converting space vector representation into a dense continuous vector space which enables to find out contextual similarity between phrases and words in a given document. Word embedding is a dense feature in a low dimensional vector. Each word is represented in form of vector that represents some features [Word embedding 3].

Word2vec, FastText, Elmo, GloVe, etc. are a few of the unsupervised algorithms based on word embeddings developed in 2010s. These do not require any annotated corpora. Routine steps followed are: (1) Preparation of data, (2) define hyper parameters like window size, embedding size, epochs, learning rate, etc., (3) generate training data and train the model, (4) user the trained model for prediction. Predictions tasks cover: “Similarity”, “dosen’t\_match”, “most\_similar”, “score” are a few functions to analyze the developed model (figure 5).

1. Visual exploration techniques in Natural Language Processing

Principal Component Analysis (PCA) is a technique for reducing the number of dimensions in a dataset while keeping most information. It uses the correlation between some dimensions and tries to provide a minimum number of variables that keeps the maximum amount of variation or information about how the original data is distributed. Since it is easy to plot and read a 2-dimensional or a 3-dimensional plot, first 2 or 3 principal components are plotted, the more the variation covered the better. [High dimension 2]

t-distributed-SNE technique is a variation of Stochastic Neighbour Embedding (Hinton and Roweis, 2002) that is easier to optimize, and produces 2 or 3-dimensional visualizations by reducing the tendency to crowd points together at the centre of the map. t-SNE technique creates a single map that reveals structure at many different scales. This is particularly important for high-dimensional data that lie on several different, but related, low-dimensional manifolds, such as images of objects from multiple classes seen from multiple viewpoints.

t-SNE has three potential weaknesses: (1) it is unclear how t-SNE performs on general dimensionality reduction tasks, (2) the relatively local nature of t-SNE makes it sensitive to the curse of the intrinsic dimensionality of the data, and (3) t-SNE is not guaranteed to converge to a global optimum of its cost function. [t-sne 1].

Another algorithm, called Uniform Manifold Approximation and Projection (UMAP) has been recently published by McInnes and Healy [UMAP 1]. They claim that compared to t-SNE it preserves as much of the local and more of the global data structure, with a shorter runtime. UMAP outputs are faster to compute and more reproducible than those from t-SNE [UMAP 2].

[Link to multiple tensor Projectors at one place for Diseases And Medicine](https://projector.tensorflow.org/?config=https://raw.githubusercontent.com/coursephd/PostgreSQL/master/DeepLearning/002-dismed-subgrp-w2v/099-multiple-config.json)

<https://perssongroup.lbl.gov/papers/dagdelen-2019-word-embeddings.pdf>

<https://onlinelibrary.wiley.com/doi/epdf/10.1111/j.1756-8765.2010.01117.x>

<https://medium.com/@b.terryjack/nlp-everything-about-word-embeddings-9ea21f51ccfe>

<https://stackoverflow.com/questions/19170603/what-is-the-difference-between-labeled-and-unlabeled-data>

file:///C:/Users/mahajvi1/Downloads/14724-66671-1-PB.pdf

<https://perssongroup.lbl.gov/papers/dagdelen-2019-word-embeddings.pdf>

<https://www.gavagai.io/text-analytics/a-brief-history-of-word-embeddings/>

<https://corpling.hypotheses.org/495>

<https://annabellelukin.edublogs.org/files/2013/08/Firth-JR-1962-A-Synopsis-of-Linguistic-Theory-wfihi5.pdf>

<https://www.springboard.com/blog/introduction-word-embeddings/>

<https://arxiv.org/ftp/arxiv/papers/1902/1902.00551.pdf>

<https://www.cse.iitb.ac.in/~pb/papers/bionlp-acl17-medical-coding.pdf>

Conversion of EMR data to a series of words:

This refers to converting an EMR into a sentence, which is essentially a series of words. Recall that an EMR is a series of time-stamped visit episodes. Each episode may contain many parts of information, but we focus mainly on diagnoses and treatments. The episode is then sequenced into a phrase. The order of the element in the phrase may follow the pre-defined ordering by the EMR system, for example, primary diagnosis is placed first, followed by secondary diagnoses, followed by procedures. Within an episode, occasionally, there are one or more transfers between care providers, for example, separate departments from the same hospital, or between hospitals. Between two consecutive episodes, there is a time gap, whose duration is generally randomly distributed. We discretize the time gap into five intervals, measured in months: (0-1], (1-3], (3-6], (6-12], 12+. Each interval is assigned a unique identifier, which is treated as a word. For example, 0-1m is a word for the (0-1] interval gap. With these treatments, an EMR is a sentence of phrases separated by words for time gaps. The phrases are ordered by their natural time-stamps. For robustness, infrequent words are coded as RAREWORD. The following is an example of a sentence.

Context of vector space, dimensionality and contexts

<https://gist.github.com/aparrish/2f562e3737544cf29aaf1af30362f469>

<https://github.com/aparrish/rwet/blob/master/understanding-word-vectors.ipynb>

How to generate embeddings:

<https://devopedia.org/word-embedding>

Very interesting way of explaining the word embedding:

<http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/#Word%20Embeddings>

The application of any statistical model requires choices; embeddings are no exception. For political scientists downloading code (or pre-fit embeddings), at a minimum, they need to decide:

1. how large a window size they want the model to use;

Window-size determines the number of words, on either side of the focus word to be included in its context. The semantic relationship appropriately modelled by embeddings varies with window-size, with larger sizes (> 2) capturing more topical relations (e.g. Obama - President) and smaller ones (< 2) capturing syntactic relations (e.g. dance - dancing). For topical relationships, larger windows (usually 5 or above) tend to produce better quality embeddings although with decreasing returns (Mikolov et al., 2013)—a result we corroborate below. Intuitively, larger contexts provide more information to discriminate between different words.

1. how large an embedding they wish to use to represent their words;

The dimensions of embedding vectors typically range between 50−450. Dimensions capture different aspects of “meaning” or semantics—hidden to the researcher—that can be used to organize words. Too few dimensions—imagine the extreme of 1—and we miss potentially meaningful relationships between words; too many—imagine the extreme of a full co-occurrence vector with every word in the vocabulary—and some dimensions are likely to be redundant (add no information). Empirically, more dimensions generally improve performance across a wide variety of tasks but with diminishing returns. Interestingly, extant literature suggests that the point at which improvements become marginal differs depending on the problem (see Melamud et al., 2016, for discussion).

1. Whether to fit the embedding models locally, or to use pre-trained embeddings fit to some other (ideally related) corpus.

Embedding models can be data hungry, meaning they need a lot of text to produce ‘useful’ results. Consequently, researchers with small corpora often use generic pre-trained embeddings trained on much larger document numbers. Pre-trained embeddings also help avoid the overhead cost associated with estimating and tuning new embeddings for each task. But there are trade-offs. The training corpus used to estimate these embeddings need not accurately capture the semantics of domain-specific texts. Intuitively, we would want to use pre-trained embeddings trained on a corpus generated by a similar “language model” —a population of speakers— to that which generated our corpus of interest. The more similar the two language models, the more similar the underlying semantics.

<https://www.nyu.edu/projects/spirling/documents/embed.pdf>

<https://tel.archives-ouvertes.fr/tel-02366013/document>

What are word embeddings and dimension?

<https://www.freecodecamp.org/news/demystify-state-of-the-art-text-classification-word-embeddings/>