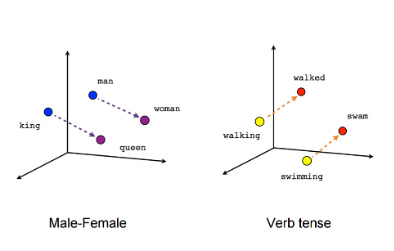
# What is a Word Embedding

Vector representations of words, or word embeddings, are a powerful tool in natural language processing. Essentially a multi-dimensional numeric representation of a word which, once generated, encapsulates a trained model’s entire understanding of a word in a machine-readable format, meaning they can easily be used as input into downstream models for more specific tasks such as classification or sentiment analysis.

## Word2Vec

In 2013 Mikolov et al. [2] proposed two model architectures for producing word embeddings in an unsupervised fashion from a corpus of text. These models were simpler than existing NN and RNN based language models and achieved a far lower computational complexity without sacrificing performance.

Interestingly, the generated word vectors also contained latent information about the relationships between words. This was recognised by the same authors in an earlier paper [8] and the Word2Vec models generate vectors which aim to maximise this property. The relationships manifest as linear regularities between word vectors and can be explored by simple vector arithmetic. The classic example being Vector(King) – Vector(male) + Vector(female) = Vector(Queen).



Vector arithmetic examples [7]

The first architecture, Continuous Bag of Words

Word2Vec is essentially a shallow feedforward model, this improves training time, but also limits the amount of information the model can capture. Another limitation is that it isn’t contextually aware. This means a word2vec model will generate the same vector for the same word, regardless of whether it has different meanings in different sentences [7].

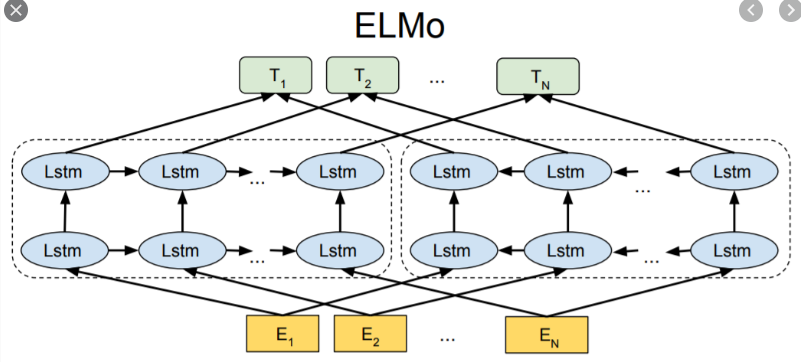
The resulting word embeddings are available to NLP practitioners as static downloadable dictionaries that can be used as input features in other models. This makes them very convenient to use.

* “Bag of words”
  + Order of context words does not influence vector prediction
* Model Architectures
  + Continuous Bag of Words (CBOW)
  + Continuous Skip-Gram (skip grams)
    - Uses current word to predict surrounding words
    - Weighs nearby words more heavily than distant words
    - Slower but better job on infrequent words

## ELMo

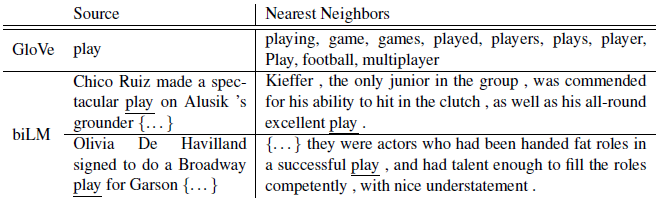
In 2018 Peters et al introduced ELMo, which stands for Embeddings from Language Models [5]. With this model, Peters et al proposed to capture not only the relationships between words (as per word2vec), but also how word uses vary across contexts. Therefore, the generated word embeddings would be functions of the entire sentence that word appears in.

They achieve this by using stacked LSTM layers. Long Short Term Memory nodes are used in neural networks to boost retention of information over longer input sequences. They achieve this by maintaining a hidden state, which is activated and added to their output by a logic gate [reference]. In the case of a language model trying to predict the next word, the model will have better knowledge about the entire sentence up to that point and will make a better prediction



<https://www.i2tutorials.com/technology/deep-dive-into-bidirectional-lstm/>

ELMo uses a pair of stacked LSTM layers going in different directions to achieve a measure of bi-directionality and capture the context of a word within a specific sentence. The table below clearly illustrates why this is an important step:



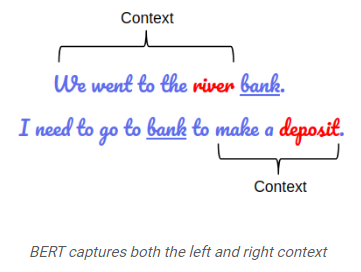
ELMo (or biLM) compared to GloVe (an earlier language model) [5]

Here we see that the previous language model GloVe had no concept of the context of a word in a sentence – the word ‘play’ had a single vector, whose nearest neighbours were words with similar vectors. ELMo (or biLM), on the other hand, must include surrounding words in the input and output as the meaning of the word ‘play’ can only really be determined by the sentence it’s used in. What’s more the two different meanings are clearly identified by the model.

Where word2vec embeddings are static and easily downloaded as a dictionary, ELMo comes as a pre-trained model, which generates embeddings based on a full sentences. Users are expected to download the weights and use the model output as a feature in their own downstream models.

## BERT

BERT (Bi-directional Encoder Representations from Transformers) improves on previous work by introducing the concept of *deep* bi-directionality. The model was introduced by Devlin et al [3] to address the fact that models up until that point where either uni-directional or only shallowly bi-directional (as per ELMo). Deep bi-directionality allows the BERT model to incorporate context from both directions in its understanding of any given word. Here’s an image from [7] showing how the meaning of the word bank can be affected by context to the left or right of the word



BERT captures both the left and right context [7]

The BERT model achieves bi-directionality by unsupervised pre-training using two different approaches.

1. It randomly masks 15% of the words and trains the model to predict the hidden words based on the other words in the sentence. This builds knowledge of word context within a sentence
2. It trains the model to predict whether sentence pairs belong together or not. This trains the model to understand context which spans sentences, which might be especially important for question answering tasks.

The concepts behind BERT are conceptually simple yet have yielded excellent empirical results [3]

Similar to ELMo, BERT can be used as-is to generate features for downstream models. However due to its more generic neural network architecture, it is also common to fine-tune BERT. This is where the entire model is retrained for a small number of iterations, thus allowing the weights to change slightly to better fit the task at hand.

## Summary table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Context Aware | Order Aware | Trained Model Required | Representations |  |  |
| Word2Vec | No | No | No | Words |  |  |
| ELMo | Yes | Uni-directional | Yes | Words |  |  |
| BERT | Yes | Bi-directional | Yes | Sub-words |  |  |

# References

1. Brief review of word embedding families, Article, 23 March 2019, <https://mc.ai/brief-review-of-word-embedding-families-2019/>
2. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. ICLR Workshop,
3. Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding, *where*
4. Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.
5. Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Settlemoyer. 2018. Deep Contextualized Word Representations, *where?*
6. Anirudh S. 2019. Word Embedding Using BERT In Python, <https://towardsdatascience.com/word-embedding-using-bert-in-python-dd5a86c00342>. TowardsDataScience.com
7. Mohid Sanad Zaki Rizvi. 2019. <https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/>. AnalyticsVidhya.com
8. T. Mikolov, W.T. Yih, G. Zweig. Linguistic Regularities in Continuous Space Word Representations. NAACL HLT 2013