Journal of Universal Computer Science, vol. 24, no. 10 (2018), 1378-1402
submitted: 18/4/18, accepted: 5/10/18, appeared: 28/10/18

Learning Concept Embeddings from Temporal Data

Francois Meyer
(Computer Science Division
Stellenbosch University, South Africa
francoisrmeyer@gmail.com)

Brink van der Merwe
(Computer Science Division
Stellenbosch University, South Africa
abvdm@cs.sun.ac.za)

Dirko Coetsee
(Praelexis, Stellenbosch, South Africa
dirko@praelexis.com)

Abstract: Word embedding techniques can be used to learn vector representations of concepts from temporal datasets. Previous attempts to do this amounted to applying word embedding techniques to event sequences. We propose a concept embedding model that extends existing word embedding techniques to take time into account by explicitly modelling the time between concept occurrences. The model is implemented and evaluated using medical temporal data. It is found that incorporating time into the learning algorithm can improve the quality of the resulting embeddings, as measured by an existing methodological framework for evaluating medical concept embeddings.
Key Words: Deep Learning, Natural Language Processing, Word Embeddings, Temporal Data, Skip-Gram
Category: I.2.6, I.2.7, I.5.1

1 Introduction

Recent advances in deep learning have produced compelling results in the field of artificial intelligence. Natural language processing is one of the contexts in which deep learning techniques have had success, achieving state-of-the-art results in numerous standard tasks such as part-of-speech tagging, named-entity recognition, sentiment analysis and machine translation [Young et al. 2017]. *Word embeddings* or vector representations of words have played an important role in these developments.

Word embeddings refer to the distributed representations of words in a vector space. These vectors are usually learned from a large text corpus. Word embedding techniques aim to retain the essential characteristics of words by observing how they co-occur with other words in the language. This strategy is based on the distributional hypothesis in linguistics, which states that words that occur in similar contexts tend to have similar meanings [Firth 1957]. Words with similar meanings or functions often have embeddings that are close to each other and the linguistic relationships between words are often reflected in the relative positions of word embeddings in the vector space [Mikolov et al. 2013b].

Using word embeddings to represent words improves performance in various tasks in natural language processing [Young et al. 2017]. The alternative is to use one-hot encoding to obtain vector representations for words. This entails representing each word as a vector of length V (the size of the vocabulary) with zeroes at all its indices except for a 1 at the index assigned to the word being represented. This reduces words to atomic symbols. It leads to high-dimensional representations for words (the dimensionality is equal to the size of the vocabulary) and fails to capture any useful information about the words. Word embedding techniques produce dense, low-dimensional representations (usually 50 to 300 dimensions) for words that capture relationships between words. The information encoded in these representations can be used by algorithms for downstream tasks, leading to improved performance.

In 2013 Mikolov et al. proposed `Word2Vec` [Mikolov et al. 2013a, Mikolov et al. 2013b], two architectural variants (skip-gram and continuous bag-of-words) of a model capable of efficiently learning word embeddings from much larger text corpora than previous techniques [Deerwester et al. 1990, Bengio et al. 2003, Collobert and Weston 2008, Collobert et al. 2011]. `Word2Vec` achieved remarkable results in capturing the syntactic and semantic properties of words (as measured by evaluation methods that test how well the embeddings capture specific properties of words such as relatedness and analogous relationships).

The idea of learning low-dimensional representations has been extended to concepts in other domains besides natural language processing such as medicine [Choi et al. 2016a, De Vine et al. 2014, Choi et al. 2016b], microbiology [Asgari and Mofrad 2015], and recommender systems [Barkan and Koenigstein 2016, Krishnamurthy et al. 2016]. In these settings the goal is to learn vector representations for the concepts in the domain (concept embeddings) that capture properties that are relevant and meaningful within the specific domain. Instead of learning linguistic properties by observing the co-occurrences of words in a text corpus, the idea is to learn domain-specific properties by observing the co-occurrences of concepts in some other setting. We are interested in the case of temporal sequences of events, where every event corresponds to the occurrence of a certain concept at a specific point in time. In the temporal setting the term *co-occurrence* refers to events that occur close to each other in time.

The medical domain will be used as an example throughout this paper to demonstrate the techniques related to concept embeddings. The concepts for which embeddings are learned in the medical domain are medical phenomena such as symptoms, diseases, treatments and procedures. The setting in which the concept embeddings are learned is the medical histories of patients. The medical history of a patient can be viewed as a temporal sequence of events (e.g. observations, diagnoses, prescriptions), each associated with the occurrence of a medical concept (e.g. symptoms, conditions, medications). Observing which medical phenomena often occur close to each other can reveal medically significant properties of the concepts, since the co-occurrences are often causal in nature [Fedak et al. 2015].

The idea of learning vector representations for concepts in domains other than natural language processing leads to the question of which properties should be captured by these representations. The goal with word embeddings is to capture semantic and syntactic properties, but the goal with medical concept embeddings would be to capture properties that would be meaningful in a medical context such as disease types and treatment effects.

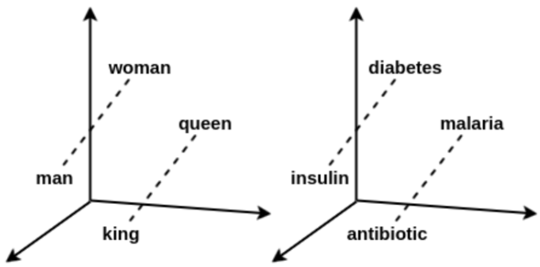

Figure 1: Word embeddings that capture semantic properties (left) and medical concept embeddings that capture causal relationships (right).

Concept embeddings have been learned by applying existing word embedding techniques, mainly `Word2Vec`, to temporal sequences of concept occurrences [Choi et al. 2016a, Choi et al. 2016b, Krishnamurthy et al. 2016]. The temporal sequences are transformed to sentences in which the words represent the concepts occurring in the sequences. This strategy, however, does not exploit any of the available temporal information except event order. Cai et al. improve on this by simultaneously learning medical concept embeddings and the temporal scopes of medical concepts [Cai et al. 2018]. They incorporate time into the model by learning weights for medical concepts for a number of discrete time periods.

The main contribution of this paper is an embedding technique designed specifically to learn concept embeddings from temporal data. By proposing a new model this paper avoids the reliance on word embedding techniques that limited previous similar efforts. We propose `Time2Vec`, a modification of the `Word2Vec` skip-gram model that incorporates the factor of time in learning concept embeddings from temporal data[1].

[1] The implementation of `Time2Vec` that accompanies this paper has been made avail-

The rest of the paper is structured as follows. Section 2 discusses the algorithms underlying existing word embedding techniques, how they have been used to learn concept embeddings and their shortcomings in this regard. Section 3 describes the method proposed by this paper to learn concept embeddings from temporal data and details the implementation of it. Section 4 presents the results obtained by applying this method to a medical data set and how it compares to existing approaches. Lastly, Section 5 discusses the conclusions reached by this investigation and proposes a few possible directions for future research.

2 Background

Word embeddings were brought to the forefront of natural language processing research by Mikolov et al. [Mikolov et al. 2013a]. They presented `Word2Vec`, two methods for learning word embeddings: skip-gram and continuous bag-of-words (CBOW). We will focus on the skip-gram model, since it performs better on most tasks [Mikolov et al. 2013a] and is the method usually used to learn concept embeddings [Choi et al. 2016a, Barkan and Koenigstein 2016, Asgari and Mofrad 2015].

2.1 Skip-gram

The objective of the skip-gram training algorithm is to learn word embeddings that can be used to predict a word's surrounding words in a sentence. Two vector representations, an input embedding and an output embedding, are learned for each word. The input embedding represents a word when it is used to predict surrounding words, while the output embedding represents a word when it is the target word being predicted. The final embeddings produced by the model use the input embeddings to represent the words. The algorithm moves through the entire training corpus, using the centre word of each training window (a specified number of consecutive words) to predict the other words in the window. The word embeddings are learned to improve the word prediction accuracy, maximising the log-likelihood

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-k\leq j\leq k, j\neq 0}\log p(w_{t+j}|w_t;\theta), \tag{1}$$

where $w_1, w_2, ...$ is the sequence of words in the training corpus, T is the length of the sequence, k specifies the training window size, and $p(w_{t+j}|w_t;\theta)$ is the probability of the word w_{t+j} occurring in the training window of the centre word w_t as predicted by the model. The log-likelihood is a function of the model

able at `https://github.com/francois-meyer/time2vec`.

parameters θ (the set of word embeddings) given the training corpus. Maximising it is equivalent to minimising the cost function

$$C(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{-k \leq j \leq k, j \neq 0} L(w_t, w_{t+j}; \theta), \tag{2}$$

where $L(w_t, w_{t+j})$ is the loss resulting from trying to predict the surrounding word w_{t+j} from the centre word w_t.

The skip-gram model trades complexity for computational efficiency, abandoning the non-linear hidden layers present in other neural language models. The model consists of an input layer encoding the input word, a hidden layer selecting the word's embedding, and a softmax classifier as the output layer:

$$p(w_{t+j}|w_t) = \frac{\exp\left({v'_{w_{t+j}}}^\top v_{w_t}\right)}{\sum_{i=1}^{V} \exp\left({v'_i}^\top v_{w_t}\right)} \tag{3}$$

where v_w and v'_w are the input and output vector representations for w, $v^\top$ is the transpose of vector v, and V is the number of words in the vocabulary.

The cost of computing the softmax function is proportional to V, which makes it impractical for language modelling, since the vocabulary being modelled consists of hundreds of thousands of words. Mikolov et al. proposed a number of strategies that reduced the computational complexity of the training algorithm, allowing the model to be trained on text corpora consisting of billions of words in less than a day [Mikolov et al. 2013b]. The most important of these strategies are computationally efficient alternatives to the expensive softmax function, many of which are sampling-based approaches that approximate the softmax function. Noise contrastive estimation (NCE) is one such strategy that reduces the training algorithm to a binary classification task [Dyer 2014]. The model is trained to distinguish a positive sample from a certain number of negative samples. The training complexity of the model is proportional to

$$O = E \times T \times C \times D \times N, \tag{4}$$

where E is the number of training epochs (iterations of the training corpus), T is the number of words in the training corpus, C is the size of the training window, D is the dimensionality of the learned embeddings and N is the number of negative samples drawn per positive training sample.

2.2 Concept embeddings

Vector space representations for concepts have been learned from temporal data sets [Choi et al. 2016a, Choi et al. 2016b, Krishnamurthy et al. 2016]. Such efforts have mainly consisted of the following strategy:

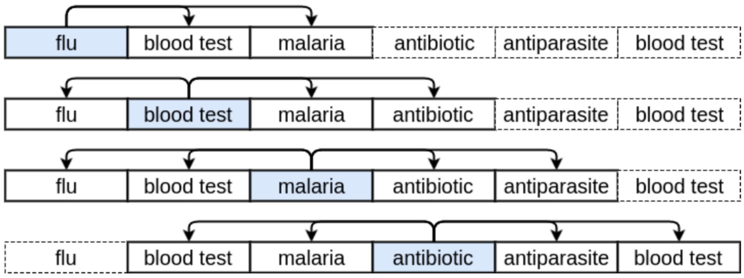

Figure 2: The skip-gram training algorithm applied to medical data. The history of a patient is treated as a sentence in which the words are the medical concepts that occur in the patient's records.

1. Transforming the data set to a text corpus in which the sentences correspond to temporal sequences and the words in a sentence represent the concepts occurring in a specific temporal sequence.
2. Applying existing word embedding techniques (sometimes with slight modifications) to the transformed data to obtain embeddings for the concepts.

Although these two steps broadly describes the strategy to learn concept embeddings, the details vary from case to case. In many cases the order of the words in the transformed data is determined by the temporal order of the concept occurrences that they represent.

Choi et al. applied the latter strategy to medical data [Choi et al. 2016a]. They transformed the medical records of a patient to sentences in a text corpus. All the medical concepts occurring within a specific time interval were transformed to "sentences" in which the words represented the medical concepts that occurred in that time interval. Then the skip-gram model was applied.

While this strategy does learn the concept embeddings from the occurrences of the concepts in a temporal sequence, it does so by treating the temporal sequences as a text corpus. This disregards the amount of time between events.

2.3 Temporal data

The idea of order exists in both text and temporal data (the order of words in a sentence and the order of events in a temporal sequence), but there is a difference between the meaning of order in the two types of data. In text data the order of words is determined by the syntax of the language and the notion of time does not exist. In temporal data each concept occurrence is associated with a timestamp and the order of concepts is determined by how they occur relative to each other in time. Successive events in a temporal sequence are not necessarily spaced equally in time and there is information about the concepts in the lengths of time intervals between their occurrences.

We expect that events occurring close to each other provide stronger evidence for causality than events occurring far apart. The nature of this temporal relationship is defined differently in different domains. In some domains, concepts that occur within an hour of each other might be the only relevant co-occurrences, while in others co-occurrences that span years might still be important to consider. The way temporal relationship strength changes over time, weakening as events are further apart, also varies from domain to domain. Although we do not investigate it in this paper, we expect that in some domains the weakening would be gradual, while in others it would be rapid.

3 METHODS

The technique proposed by this paper is based on the skip-gram model of `Word2Vec`, incorporating the factor of time into the learning algorithm. Section 3.1 discusses the theory underlying the proposed concept embedding technique and Section 3.2 describes the implementation that accompanies this paper.

3.1 Time2Vec

The strategies to learn concept embeddings from temporal data described so far incorporate the factor of time at a rudimentary level. They treat all concept co-occurrences that are close in time the same, while ignoring concept co-occurrences that are far apart. These strategies rely largely on word embedding techniques and make use of existing implementations [Mikolov et al. 2013b, Pennington et al. 2014].

This paper proposes `Time2Vec`, a time-based concept embedding technique. We propose the following modifications to standard word embedding models:

- The model defines the sliding training window to include all concepts occurring within a specific time interval, instead of a certain number of concepts.
- The importance assigned to concept co-occurrences in the learning algorithm depends on the length of the time interval between the occurrences, decreasing as more time passes.

Our method is an attempt to reformulate skip-gram, a natural language processing model, as a temporal sequence model. `Time2Vec` retains many of the characteristics of skip-gram that make it effective and efficient.

3.1.1 Temporal training window

The objective of the `Time2Vec` training algorithm is to learn concept embeddings that can be used to predict the concepts that occur close to a concept in time. The algorithm moves through each temporal sequence in a data set, using each

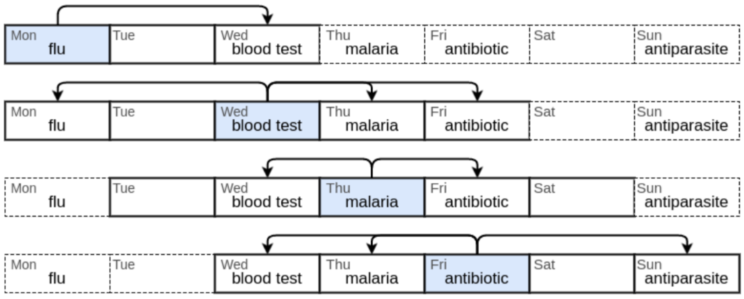

Figure 3: Time2Vec slides a temporal training window through a patient's medical history, using each concept to predict the concepts occurring around it within a specified time interval. In this example the time interval is two days before to two days after.

concept occurrence to predict the concepts that occur close to it in the past and in the future. The training window around the concept occurrence c_t is defined as the set of concept occurrences S_t that occur before and after it within a specified time interval. So

$$S_t = \{c_j \,|\, d_{t,j} < l, j \neq t\}, \tag{5}$$

where $d_{t,j}$ is the length of time between concept occurrences c_t and c_j, regardless of order, and l is the length of the time interval within which concept occurrences are observed. The total time interval of the training window is therefore of length $2l$, since it considers concept occurrences before and after c_t.

The concept embeddings are learned to improve the probabilities of the concepts occurring close to concept c_t given the occurrence of c_t, maximising the log-likelihood of the data,

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{c_j \in S_t} \log p(c_j | c_t), \tag{6}$$

where $c_1, c_2, ..., c_T$ is the temporal sequence of concept occurrences. This definition of the training window make sense: the proximity of concept occurrences should be viewed with regard to their positions in time, instead of their relative positions in a sequence.

3.1.2 Temporal decay

The co-occurrence of two concepts in a temporal training window is assigned an importance based on how far apart the concepts occur in time. The co-occurrence of two concepts that occur close to each other is assigned a higher importance than the co-occurrence of two concepts that occur further apart in time.

We then weight the cost function of the learning algorithm with these importance scores. This ensures that losses from more important co-occurrences contribute more to the resulting cost function than losses from less important co-occurrences (concepts that occur further apart in time).

Weighting the cost function is a strategy in machine learning that is employed to handle the problem of unbalanced classes (one class of training samples being much more prevalent than others). The losses from less common classes are weighted more heavily to penalise misclassifications more severely in the training algorithm. However, in `Time2Vec` the goal of weighting the cost function is to penalise the learning algorithm more for incorrectly predicting important co-occurrences and less for incorrectly predicting unimportant co-occurrences.

Maximising the objective function of `Time2Vec` is equivalent to minimising the cost function

$$C(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{c_j \in S_t} \alpha(c_t, c_j) L(c_t, c_j), \tag{7}$$

where $L(c_t, c_j)$ is the loss resulting from trying to predict c_j from c_t and $\alpha(c_t, c_j)$ is a numerical measure of the importance assigned to the co-occurrence (c_t, c_j). The importance assigned to co-occurrences is controlled by α, which is a function of the length of time between c_t and c_j. This is a modification of the cost function of the skip-gram model (Equation 2) that incorporates the weights assigned to the co-occurrences. The main intention of the modification is to assign more importance to concepts that occur close to each other and less importance to concepts that occur further apart. Therefore we choose α to be a function that decreases as the amount of time between two concept occurrences increases. We evaluate the following two decay functions:

$$\alpha(c_t, c_j) = \begin{cases} r - \lambda d_{t,j} & d_{t,j} \leq l \\ 0 & d_{t,j} > l, \end{cases} \tag{8}$$

$$\alpha(c_t, c_j) = \begin{cases} r e^{-\lambda d_{t,j}} & d_{t,j} \leq l \\ 0 & d_{t,j} > l, \end{cases} \tag{9}$$

where r is a constant, λ is a positive rate of decay and l is a hard limit after which concept occurrences are disregarded, as defined in the formulation of the temporal training window. These values are hyperparameters of the model that determine the nature of the temporal decay used by the learning algorithm. It should be noted that the decay functions above can never take on negative values. The value of l can be specified to force the decay function to zero for all values of $d_{t,j}$ greater than l or it is naturally the value of $d_{t,j}$ for which the decay function is equal to zero.

The type of decay function and the value of its parameters should depend on the domain being modelled, since the strength of temporal connections depends on the domain. In the linear decay of Equation 8 the importance of co-occurrences decreases gradually as concepts occur further apart, while in the exponential decay of Equation 9 the importance of co-occurrences decreases rapidly over time. These decay functions ensure that losses corresponding to concepts with strong temporal connections contribute more to the cost function than the losses corresponding to concepts with weaker temporal connections. The intended effect of this is that the extent to which the embeddings are adjusted to improve predictions depends on how close the predicted concepts occur to each other.

3.2 Implementation

We avoid a reliance on existing implementations of word embedding techniques. We therefore implemented the technique in Python in the form of a package named `time2vec`. Since the model is based on the skip-gram model of `Word2Vec` the implementation is structured similarly. It also incorporates many of the strategies proposed by Mikolov et al. [Mikolov et al. 2013b] to reduce computational complexity and improve the quality of embeddings, such as approximating the softmax with noise contrastive estimation and subsampling frequent concepts. The new computational challenges that do not arise in the implementation of word embedding techniques are discussed in Section 3.2.3.

The system is organised in a pipeline architecture with the following component functionalities: initialising a model, building the vocabulary, generating training data, learning embeddings, and exploring embeddings. These steps have to be executed in the order in which they are presented here, because each step relies on the steps before it. However, since some of the steps can take considerable time to execute (generating training data and learning embeddings can take several hours if the data set contains millions of records), the implementation makes it possible to keep the progress of previous steps if some of the later steps are executed again, but with different hyperparameters. This ensures that the effect of different model hyperparameters can be tested efficiently at different stages of the modelling process.

The model is implemented as a Python class with methods that execute the different stages of the system pipeline. The parameters and functions mentioned in the following discussions are of the `Time2Vec` class, unless indicated otherwise.

A constraint in implementing embedding models is that the large data sets used to train the models do not fit into memory. The implementation is designed to handle data sets larger than the memory capabilities of the computer on which it is executed. This is achieved by operating on smaller, manageable chunks of the data set during all stages of the training process. The number of records read into memory at any time is specified by the `chunk_size` variable. All the operations on the data set are performed in this way to ensure memory efficiency.

id	datetime	concept
patient1	2017-06-01	flu
patient1	2017-06-03	blood test
patient1	2017-06-04	antibiotics
patient2	2015-01-01	lung cancer
patient2	2016-02-03	chemotherapy
patient3	2015-12-04	blood test
patient3	2015-12-07	malaria
patient3	2015-12-07	antiparasite

Table 1: An example of training data for `Time2Vec`.

3.2.1 Model initialisation

The model is initialised by creating an instance of the `Time2Vec` class. This requires specifying a data set from which to learn the embeddings with the `data_file_name` parameter. The data set must be stored in a CSV file with the following fields:

– `id`: identifies a specific temporal sequence (e.g. the medical records for a specific patient).
– `datetime`: time stamp of the concept occurrence (in the Python date format `%Y-%m-%d %H-%M-%S`).
– `concept`: name of the concept that occurred.

The data set is also required to be grouped by `id` and ordered temporally within these groups (see Table 1 for an example). This requirement on the format of the data is convenient for computational reasons. The `Time2Vec` class also requires a number of parameters during initialisation that specify the hyperparameters of the model. The roles of these hyperparameters are detailed during the discussions of how they are incorporated into the implementation. The hyperparameters are:

Temporal parameters

The modifications we propose rely on a number of parameters. The type of decay (none, linear or exponential) and the unit of time used to calculate the decay (days, months or years) have to be specified, as well as the numerical parameters of Equations 8 and 9.

Embedding parameters

The dimensionality of the embeddings have to be specified, as well as the strategies from word embedding techniques (e.g. noise contrastive estimation) that are employed to speed up training and improve the embeddings.

```
Data: Temporal data set (see Table 1)
Result: Model vocabulary
for each concept c_t in the data set do
    if c_t in raw_vocab then
        count[c_t]++
    else
        add c_t to raw_vocab
        count[c_t] = 1
    end
end
for each concept c_j in raw_vocab do
    if count[c_j] ≥ min_count then
        add c_j to vocab
    end
end
```

Algorithm 1: The procedure implemented in `build_vocab` constructs the vocabulary of the model.

Training parameters

Training is subject to standard hyperparameters in machine learning models. They are the learning rate, batch size, the number of training epochs, and the percentage of training data set aside for validation.

3.2.2 Building the vocabulary (see Algorithm 1)

The *vocabulary* is the set of concepts for which embeddings are learned. Creating the vocabulary requires gathering and storing information about the concepts that occur in the data set. This functionality is implemented in the `build_vocab` method that iterates through the entire data set, accumulating the following information:

- The set of distinct concepts that occur in the data set.
- The number of times each concept occurs in the data set.
- The total size of the data set (number of records).

This information enables the creation of a model vocabulary in the form of a lookup table that maps all the concepts in the vocabulary to unique indices. These indices are used to encode the concepts numerically during training (using a 1-of-V encoding, where V is the size of the model vocabulary).

In addition to creating the vocabulary, `build_vocab` also includes implementations of minimum count omissions and subsampling, strategies developed for speeding up training and improving the quality of the resulting embeddings.

Minimum count

During construction of the lookup table, concepts that occur less than `min_count` times (typically around 5 in `Word2Vec`) are omitted from the vocabulary. This ensures that they are ignored during training and that no embeddings are learned for them. The motivation for this is that certain concepts could occur so few times that the model would not be able to capture the properties of the concepts in their embeddings. The only concepts considered by the model are those that occur often enough that high quality embeddings can be learned for them.

Subsampling

The problem of some concepts occurring too often is handled by subsampling. The `subsampling` parameter specifies the relative frequency threshold for determining how certain concept occurrences are randomly removed. The probability of keeping a concept occurrence is determined by:

$$p = \left(\sqrt{\frac{f}{t}} + 1\right) \times \frac{t}{f}, \tag{10}$$

where t is the value of the `subsampling` parameter and f is the relative frequency of the concept in the data set (the proportion of records in the data set corresponding to occurrences of the concept). Concepts that occur more frequently than others are more likely to be removed from the data set. This ensures that the disproportionate influence of very frequent concepts is diminished by diluting their presence in the data set. Smaller values of t correspond to the removal of more concept occurrences.

3.2.3 Generating training data (see Algorithm 2)

The modifications proposed by this paper introduce new computational challenges that do not appear in the context of word embedding techniques. `Word2Vec` learns from a text corpus, moving through it word by word and predicting the words before and after a word. The algorithm keeps track of its position in the training corpus and finds the words that it has to predict by looking at a specific number of words before and after of a word.

`Time2Vec` cannot learn as directly from temporal data, since it requires additional computation to generate training data. For each concept occurrence the training window has to be determined. This entails finding all the concepts that occur within the specified time interval around the concept occurrence. Only then are the concepts that have to be predicted known. Secondly, the addition of temporal decay requires the loss from each prediction within the training window to be weighted in order to specify the importance assigned to it. This

```
Data: Temporal data set (see Table 1)
Result: Training data for Time2Vec
for each id do
    for each record c_t of id do
        for each record c_j of id after c_t do
            if c_j is within time interval l from c_t then
                calculate weight = α(c_t, c_j)
                add c_t → c_j with weight to training data
                add c_j → c_t with weight to training data
            else
                break
            end
        end
    end
end
```

Algorithm 2: The training data generation takes advantage of the requirements on the format of the data set and the symmetry of the samples in the eventual training data. Instead of looking forward and backward in time from each concept occurrence, it only looks forward and adds the corresponding backward prediction when adding a forward prediction, only calculating the common weight once, since $\alpha(c_t, c_j) = \alpha(c_j, c_t)$.

requires calculating a weight for each of the co-occurrences within a training window using a decay function, as described in Section 3.1.2. This leads to expensive additional computation, since there are as many training windows as there are concept occurrences, each requiring a number of weights to be calculated. The training complexity of the model is proportional to:

$$O = E \times T \times Q \times C \times D \times N \tag{11}$$

where E is the number of training epochs (iterations of the temporal data set), T is the size of the data set, Q is the computational complexity of determining the training window for a concept occurrence and computing the weights of all the samples in the training window, C is the average number of concepts in a training window, D is the dimensionality of the learned embeddings and N is the number of negative samples drawn per true training sample.

The Q term in the expression above is what leads to the additional complexity of `Time2Vec` compared to `Word2Vec`. For each concept occurrence the concepts that occur in its temporal training window have to be determined and a weight has to be calculated for each of these concepts. It means that besides training the model, generating training data for the model proves a challenge. We therefore structure generating training data as a separate step in the training process, unlike with word embedding techniques.

The method `gen_train_data` generates the training data by transforming the temporal data set to the format required by the learning algorithm (a set of weighted training samples). The algorithm that is used to perform this transformation is described in Algorithm 2. The training data is then stored in the CSV file specified by the `train_file_name` parameter. This allows other models to use the training data by loading it with the function `load_train_data`, instead of having to generate their own training data.

Parallelisation

The training data generation is parallelised in the implementation of `gen_train_data` by concurrently applying Algorithm 1 to separate chunks of the data set in order to speed up the execution. This is implemented with the `multiprocessing` module in Python. The `processes` parameter determines how many processes the training data is distributed to. The generated training data is safely written to the training data file from the separate processes by protecting write access to the file with a lock.

3.2.4 Learning embeddings

Training is implemented in the method `learn_embeddings`. The method constructs the model proposed by this paper, trains it on the generated training data, and produces concept embeddings for the concepts in the vocabulary. The machine learning aspects of the model were implemented with TensorFlow[2], an open-source software library for machine learning. The choice to use TensorFlow was motivated by the fact that it supports the operations required by word embedding techniques as part of its API. TensorFlow programs consist of a construction phase, in which the model is specified, and the execution phase, in which the model is trained.

Construction phase

The structure of the model resembles that of the skip-gram model of `Word2Vec`, but the losses are weighted as described in Section 3.1. The softmax is approximated with noise contrastive estimation (NCE), using the `nce_loss` function provided by TensorFlow. The `num_samples` parameter specifies the number of negative samples used per training sample for the objective function.

Execution phase

The optimisation of the objective function is done through mini-batch gradient descent, the standard optimisation technique for deep learning, using the

[2] https://www.tensorflow.org/

`GradientDescentOptimizer` class provided by TensorFlow. The learning rate of the optimiser (how quickly the model learns) can be specified by the `lr` parameter. If the value of the `min_lr` parameter is less than the `lr` parameter, the learning rate decays linearly over the training epochs, starting at `lr` and ending at `min_lr`. The size of training batches (the number of training samples given to the learning algorithm at a time) is specified by the `batch_size` parameter. The learning algorithm iterates the entire temporal data set as many times as specified by the `epochs` parameter. The `valid` parameter specifies the proportion of the training data to be removed during training and used to obtain a validation loss that is reported after each epoch. The validation loss is the loss resulting from applying the model to the previously unseen data. It provides a measure of the current performance of the model. The hyperparameters of the model can be tuned by observing the behaviour of the validation loss.

3.2.5 Exploring embeddings

The embeddings learned by the model can be viewed and stored for later use by calling the `save` method. The vector representations of specific concepts can be obtained with the `get_vector` function. The concepts that are most similar to a specific concept in terms of the cosine similarities of their embeddings (a measure of similarity used in the context of word embeddings) can be found using the `most_similar` function. This can be used to investigate how the concept embeddings cluster together.

4 Results

The embeddings presented here were obtained by applying the implementation discussed in Section 3.2 to a medical data set. The medical domain is well suited for `time2vec` and provides an existing evaluation framework with which to evaluate medical concept embeddings. The results obtained with the strategies proposed by this paper are compared to the results obtained by applying the approach proposed by Choi et al. to the medical data set [Choi et al. 2016a].

The evaluation and comparison of medical concept embeddings provide different challenges to that of word embeddings. However, some of the ideas underlying the standard evaluation methods for word embeddings [Mikolov et al. 2013b] can be extended to evaluation methods for medical concept embeddings. Evaluation methods for word embeddings measure to what extent embeddings capture the syntactic and semantic properties of words. Evaluation methods for medical concept embeddings should measure to what extent embeddings capture the medical properties of the concepts. This can be done by comparing the structure of a set of medical concept embeddings to existing medical ontologies. This approach tests whether the embeddings capture the information represented by specific medical ontologies, such as the categories of medical concepts, their properties, and the relationships between them.

The evaluation framework used to evaluate embeddings in this section is based on the methodological framework proposed by Choi et al. [Choi et al. 2016a]. It provides a quantitative methodology that can be used to investigate the characteristics of medical concept embeddings. The framework consists of two approaches to evaluation (described in Sections 4.3 and 4.4), each testing a different property of the medical concept embeddings.

4.1 Data set

The type of medical data required by `Time2Vec` is referred to as patient-level data. A data set of this type consists of patient records that provide the medical history of each patient as a sequence of medical events. Each event should consist of the time-stamped occurrence of a medical concept associated with a specific patient. However, patient-level data such as this is not publicly available. This is because of restrictions on the distribution of such data intended to protect the privacy and confidentiality of patients.

An alternative that has emerged out of the need for patient-level data for research is simulated medical data. This refers to data containing virtual patients with simulated medical histories. The Observational Medical Outcomes Partnership (OMOP) designed and developed the Observational Medical Dataset Simulator (OSIM) that simulates patient-level medical data [Murray et al. 2011]. The second version of this simulator (OSIM2) generates simulated medical data with a model that is based on the characteristics of real healthcare data. The model generates data sets that contain hypothetical patients with simulated medical histories. The medical histories consist of records of diseases and treatments that represent the relationships that exist in real medical data.

Observational Health Data Sciences and Informatics (OHDSI) have made medical data sets generated with OSIM2 avaible online[3]. One of their data sets is used to evaluate `Time2Vec` in this paper. It is a data set that contains 1 million simulated patients. The data set required some preprocessing in order to get it to the format required by `Time2Vec`. The original data set stores the patient records in two separate tables. The first table contains records related to patient conditions and the second table contains records related to prescribed drugs. Both of these tables contain columns not used by `Time2Vec`. The only columns that are required are the ones that store the patient ID, the date of the occurrence and concept that occurred. The concepts are represented by concept IDs, which are unique identifiers for all the concepts modelled by OSIM2. The concept IDs are mapped to the concepts that they represent by a set of vocabularies made

[3] ftp://ftp.ohdsi.org/osim2/

available online by OHDSI[4]. These vocabularies contain descriptive information about the medical concepts such as their names and their types. The following steps of preprocessing were performed:

1. The unnecessary columns were removed from the original two tables.
2. The two tables were combined into a single table with all patient records.
3. The concept IDs were replaced with concept names by mapping the IDs to names with the vocabularies.

The original OSIM2 data set contains 42,197,301 patient records, each describing the occurrence of a medical concept for one of the 1,216,554 patients. The vocabularies did not provide a mapping from concept ID to concept name for all the concepts in the data set. The reason for the presence of these unmappable concept IDs could not be determined. The patient records containing concepts that could not be mapped with the vocabularies were removed from the data set. The result is a data set in the format required by `Time2Vec` that contains 1,211,863 patients with 38,199,509 medical records.

4.2 Embeddings

Choi et al. learned medical concept embeddings by applying the skip-gram model of `Word2Vec` to the medical insurance claims data of 4 million patients [Choi et al. 2016a]. They proposed the following strategy:

1. Partition the medical history of a patient into time intervals of a certain size.
2. Remove duplicate concept occurrences within each partition and randomly shuffle the remaining concepts.
3. Treat the concept occurrences of each partition as a sentence and apply the skip-gram model to these sentences.

This method was applied to the OSIM2 data set by setting up a `Time2Vec` model that partitions the medical records of patients into months. Since the temporal training window is effectively removed by this setup, it is equivalent to applying the skip-gram model to the partitioned data. The second step of their approach was omitted, since it is aimed at removing the many duplicate codes that appear in medical claims data and is not necessary for the OSIM2 data.

In addition to the concept embeddings obtained by applying the above method, two groups of `Time2Vec` medical concept embeddings were learned from the OSIM2 data set. The first group defines the temporal training window as 30 days. Three sets of such embeddings were obtained, each using a different type of temporal decay (no decay, linear and exponential) within the 30-day training window. The second group of embeddings define the temporal training window as 14 days. This group also contains three sets of embeddings, each using a different type of temporal decay. These embeddings are compared to the embeddings

[4] http://ftp.ohdsi.org/

obtained with the baseline approach and to each other. This makes it possible to observe the effect of different temporal training windows and different temporal decay strategies on the resulting embeddings.

The parameters of the linear decay functions were chosen such that the functions become zero at the endpoints of the training window. The parameters of the exponential decay functions were chosen to produce relatively high rates of decay, truncating to zero at the endpoints of the temporal training windows, in order to contrast it against the gradual decay of the linear decay functions.

The training hyperparameters of all the models were tuned by observing the behaviour of the validation loss for different values of the hyperparameters. The learning rates that produced smoothly decreasing validation losses for each of the models were used to obtain the final embeddings. The following hyperparameters were used for all of the models:

- The embeddings are 100-dimensional (`dimen` = 100).
- A minimum count of 5 was required (`min_count` = 5).
- No subsampling was employed (`subsampling` = 0).
- The models were trained for 20 epochs (`epochs` = 20).
- The batch size during training was 32 (`batch_size` = 32).

Generating training data for the models was the most computationally demanding step, taking between 5 and 10 hours when distributed over 10 2GHz Intel Xeon E5-2640 v2 processors. Learning the embeddings from the training data took between 2 and 5 hours when automatically distributed by TensorFlow over the machine's available CPU cores.

4.3 Conceptual similarity

The first evaluation strategy tests whether the property of conceptual similarity is captured by the medical concept embeddings. The approach determines whether the embeddings of conceptually similar concepts (i.e. concepts of the same type) are grouped together. The conceptual type of each medical concept in the data set is obtained with the OHDSI vocabularies, which maps each medical concept to a clinical category known as its domain. The data set contains concepts from four domains: conditions, drugs, procedures and observations.

Choi et al. proposed the Medical Concept Similarity Measure (MCSM) to quantitatively evaluate this characteristic for a set of medical concept embeddings [Choi et al. 2016a]. The MCSM of a set of medical concepts V with respect to a medical concept type T (e.g. conditions), parameterised by a neighbourhood size k is defined as

$$\mathrm{MCSM}(V,T,k) = \frac{1}{|V(T)|} \sum_{v \in V(T)} \sum_{i=1}^{k} \frac{1_T(v(i))}{\log_2(i+1)}, \tag{12}$$

Closest neighbours of Diabetes mellitus screening	(Procedure)
Thyroid disorder screening (Procedure)	**0.995805**
Viral screening (Procedure)	**0.992821**
Endocrine/metabolic screening (Procedure)	**0.991196**
Hyperlipidemia screening (Procedure)	**0.989067**
History of clinical finding in subject (Observation)	0.988393
High risk sexual behavior (Condition)	0.987506
Genitourinary disease screening (Procedure)	**0.987456**
Nephropathy screening (Procedure)	**0.986403**
Screening for disorder (Procedure)	**0.985877**
Contact - infectious disease (Condition)	0.984437

Table 2: The medical concept types and cosine similarities of the 10 closest neighbours of *Diabetes mellitus screening* in the embedding space.

where $V(T) \subset V$ is the set of medical concepts of type T, $v(i)$ denotes the ith closest neighbour of the medical concept v in the embeddings and 1_T is an indicator function which is 1 if the concept $v(i)$ is of type T and 0 otherwise. The MCSM tests whether the embeddings close to the embedding for a concept tend to be for concepts of the same type. The MCSM of a set of medical concept embeddings with respect to a medical concept type is defined as the average of this measure over all the concepts of the specific type. A higher MCSM for a medical concept type indicates that embeddings for concepts of the type tend to cluster together and that conceptual similarity is learned by the embeddings.

Table 2 demonstrates how the measure is computed. The medical concept *Diabetes mellitus screening* is of the type *Procedure*. The closest neighbours of the concept in the embedding space are used to compute the MCSM, with the neighbours of the same type contributing to the measure. The MCSMs of different sets of embeddings are compared in Table 3.

Table 3 compares embeddings obtained with the strategies proposed by this paper to embeddings obtained with the baseline approach described in Section 4.2 (using monthly partitions). The models with a 30-day temporal training window achieve the highest MCSMs for all of the concept types. The model with linear decay in the 30-day temporal training window achieves the highest MCSM for three of the four concept types. The model with exponential decay in the 30-day temporal training window achieves the highest MCSM for one of the concept types. Somewhat surprisingly, the models with no temporal decay do not perform better than the baseline approach with regard to the MCSM. This shows that the temporal training window alone does not improve the model's ability to capture the medical similarity property. However, the addition of temporal decay within the temporal training window leads to improved performance with regard to the

Decay	Conditions	Drugs	Procedures	Observations
Baseline	14.00	9.48	1.66	1.35
30-day temporal training window				
None	13.97	9.43	1.72	1.33
Linear	**14.41**	**9.59**	3.17	**3.14**
Exp	13.88	7.57	**3.22**	3.13
14-day temporal training window				
None	13.47	7.31	1.9	1.34
Linear	14.12	8.61	2.91	2.33
Exp	13.85	7.05	2.91	2.32

Table 3: Comparison of the MCSMs of embeddings learned with the baseline approach to those of embeddings learned with with `Time2Vec` using different temporal training windows and temporal decay strategies.

MCSM. The models with linear temporal decay achieve higher MCSMs than the models with the same temporal training window without temporal decay. In particular, the MCSMs of the linear decay models for the concept types *Procedures* and *Observations* are higher than those of models with the same temporal training window without decay. These concept types are the rarest in the OSIM2 data set and therefore the most challenging to cluster together.

Concepts of more prevalent types are more likely to seem to cluster together, since they occur in the neighbourhood of concepts simply because they make up a high proportion of the concepts. The models with exponential temporal decay achieve lower MCSMs than any of the other models for the concept types *Conditions* and *Drugs*, but achieve results close to those of the linear models for the two less prevalent concept types. The models with temporal decay are able to capture the conceptual similarity of rare concept types particularly well.

4.4 Medical relatedness

The second evaluation strategy tests whether the property of medical relatedness is captured by the medical concept embeddings. The approach determines whether the embeddings of medically related concepts (i.e. concepts that belong to the same branch or discipline of medicine) are grouped together. The diseases in the data set are mapped to their equivalent codes in the SNOMED-

Closest neighbours of Asthma	(Asthma)
IgE-mediated allergic asthma (Lung disease due to external agents)	0.96091
Exacerbation of asthma (Asthma)	**0.959216**
Cough (Miscellaneous mental health disorders)	0.955143
Asthma with status asthmaticus (Asthma)	**0.95311**
Montelukast (None)	0.950272
Bronchospasm (Other upper respiratory disease)	0.949745
Disorder of respiratory system (Other lower respiratory disease)	0.948517
Intrinsic asthma without status asthmaticus (None)	0.948444
Budesonide (None)	0.947301
Extrinsic asthma with status asthmaticus (None)	0.947129

Table 4: The single-level CCS categories and cosine similarities of the 10 closest neighbours of *Asthma* in the embedding space.

CT[5] terminology with the OHDSI vocabularies. The SMONED-CT codes are then mapped to their equivalent ICD-9-CM codes using the mapping provided by the U.S. National Library of Medicine[6]. The ICD-9-CM codes are grouped into categories by the CCS[7] classification system, which provides single-level and multi-level groupings of all the ICD-9-CM diseases. The medical relatedness property is evaluated by taking advantage of three of these groupings: the single-level grouping and the first two levels of the multi-level grouping. The single-level grouping classifies the ICD-9-CM codes of diseases into 285 mutually exclusive categories, while the multi-level grouping classifies the codes into a hierarchical classification system in which each level further subdivides the preceding levels into more specific categories of diseases.

Choi et al. proposed the Medical Relatedness Measure (MRM) to quantitatively evaluate this characteristic for a set of medical concept embeddings [Choi et al. 2016a]. The structure of the MRM is the same as that of the MCSM, but the MRM uses a classification of medical relatedness instead of a classification of medical concept types. The MRM of a set of medical concepts V with respect to a disease grouping G, parameterised by a neighbourhood size k is defined as

$$\mathrm{MRM}(V, G, k) = \frac{1}{|V(G)|} \sum_{v \in V(G)} \sum_{i=1}^{k} \frac{1_G(v(i))}{\log_2(i+1)}, \tag{13}$$

where $V(G) \subset V$ is the set of medical concepts which can be mapped to ICD-9-CM codes, $v(i)$ denotes the ith closest neighbour of the medical concept v in

[5] http://www.snomed.org/snomed-ct

[6] https://www.nlm.nih.gov/research/umls/mapping_projects/snomedct_to_icd9cm_reimburse.html

[7] https://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccsfactsheet.jsp

Decay	Single-level	Multi-level 1	Multi-level 2
Baseline	0.88	2.78	1.58
30-day temporal training window			
None	0.90	2.82	1.62
Linear	**1.78**	**3.91**	**2.82**
Exponential	1.63	3.79	2.7
14-day temporal training window			
None	0.85	2.76	1.56
Linear	1.48	3.69	2.53
Exponential	1.4	3.55	2.42

Table 5: Comparison of the MRMs of embeddings learned with the baseline approach to those of embeddings learned with with `Time2Vec` using different temporal training windows and temporal decay strategies.

the embeddings and 1_T is an indicator function which is 1 if the concept $v(i)$ is in the same group as v according to G and 0 otherwise.

The MRM tests whether the embeddings close to the embedding for a disease tend to be for diseases that are medically related. The MRM of a set of medical concept embeddings with respect to a disease grouping G is defined as the average of this measure over all the diseases. A higher MRM for a grouping indicates that embeddings for medically related diseases tend to cluster together and that medical relatedness is learned by the embeddings.

Table 4 demonstrates how the measure is computed. The medical concept *Asthma* is in the more general *Asthma* category according to the CCS single-level disease grouping. The closest neighbours of the concept in the embedding space are used to compute the MRM, with the neighbours in the same category contributing to the measure. The MRMs of different sets of embeddings are compared in Table 5.

Table 5 compares the same embeddings as Table 3. The model with linear decay in the 30-day temporal training window achieves the highest MRM for all three of the groupings under consideration. Interestingly, the results almost mirror the conceptual similarity results. The models with no temporal decay do not perform better than the baseline approach with regard to the MRM, but the addition of temporal decay does lead to increasing MRMs. The models with temporal decay, both linear and exponential, achieve significantly higher MRMs than the models with no temporal decay for all of the groupings. The linear decay models narrowly outperform the exponential decay models with the same temporal training windows for each of the groupings. The models with temporal decay are better able to capture the medical relatedness of concepts.

5 Conclusion and Future Work

This paper investigated the use of word embedding techniques to learn concept embeddings in domains other than natural language processing. We proposed `Time2Vec`, a model designed specifically to learn concept embeddings from temporal data. The model modifies the skip-gram architecture of `Word2Vec` to incorporate the factor of time by defining the training window on the time between events and having the importance assigned to co-occurrences decay according to the time between events.

The model was applied to a synthetic medical data set to obtain medical concept embeddings. The learned embeddings were evaluated using an existing methodological framework that evaluates how effectively a set of embeddings capture the medical properties of concepts. The results showed that just introducing a temporal training window did not improve the quality of the medical concept embeddings. However, the addition of temporal decay within the temporal training window, particularly linear decay, improved the model's ability to capture the medical properties in the embeddings. The results reveal the underlying temporal characteristics of the data.

The models using a 30-day temporal training window with temporal decay achieve the best performance among all the sets of embeddings. This shows that the events occurring within 15 days of each other are relevant, but that that the importance of these co-occurrences decrease as the events occur further apart. This is a property of the medical data and the way that the strength of the temporal relationship between medical events decays over time.

In the future, the proposed model should be applied to real medical data and other types of temporal data to further evaluate the modifications proposed by this paper. Other domains in which `Time2Vec` can be tested include entertainment media (using viewing histories) and personal finance (using payment histories). The ideal temporal training window size and temporal decay type should also be investigated for new domains. There are also other ways to incorporate the factor of time into the learning algorithm that might improve performance: The training window can be further modified (e.g. predicting only in one direction) and the temporal decay can be incorporated into other word embedding architectures (e.g. the continuous bag-of-words model of `Word2Vec`). Finally, the implementation of `Time2Vec` revealed new computational challenges and the usability of the package would benefit from improvements to the runtime complexity of the feature extraction algorithm.

References

[Asgari and Mofrad 2015] Asgari, E., Mofrad, M.: "Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics"; PLoS ONE 10(11): e0141287, 2015.

[Barkan and Koenigstein 2016] Barkan, O., Koenigstein, N.: "Item2Vec: Neural Item Embedding for Collaborative Filtering"; arXiv preprint arXiv:1603.04259, 2016.

[Bengio et al. 2003] Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: "A Neural Probabilistic Language Model"; The Journal of Machine Learning Research, 3, pp. 1137-1155, 2003.

[Choi et al. 2016a] Choi, Y., Chiu, C., Sontag, D.: "Learning low-dimensional representations of medical concepts"; AMIA, pp. 41-50, 2016.

[Choi et al. 2016b] Choi, E., Bahadori, M., Searles, E., Coffey, C., Sun, J.: "Multilayer Representation Learning for Medical Concepts"; Proceedings of the 22nd ACM SIGKDD International Conference on KDD, pp. 1495-1504, 2016.

[Collobert and Weston 2008] Collobert, R., Weston, J.: "A unified architecture for natural language processing"; Proceedings of the 25th International Conference on Machine Learning 08, 20(1), pp. 160-167, 2008.

[Collobert et al. 2011] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: "Natural Language Processing (Almost) from Scratch"; Journal of Machine Learning Research, 2011.

[Deerwester et al. 1990] Deerwester, S., Dumais, S., Landauer, T., Furnas, G., Harshman, R.: "Indexing by latent semantic analysis"; Journal of the American Society for Information Science, pp. 391-407, 1990.

[De Vine et al. 2014] De Vine, L., Zuccon, G., Koopman, B., Sitbon, L., Bruza, P.: "Medical Semantic Similarity with a Neural Language Model"; Proceedings of the 23rd ACM International CIKM, pp. 1819-1822, 2014.

[Dyer 2014] Dyer, C.: "Notes on Noise Contrastive Estimation and Negative Sampling"; arXiv preprint arXiv:1410.8251, 2014.

[Fedak et al. 2015] Fedak, K., Bernal, A., Capshaw, Z., Gross, S.: "Applying the Bradford Hill criteria in the 21st century: how data integration has changed causal inference in molecular epidemiology"; DOI: 10.1186/s12982-015-0037-4, 2015.

[Firth 1957] Firth, J.: "A synopsis of linguistic theory 1930-1955"; Studies in linguistic analysis, pp. 1-32. Oxford: Blackwell, 1957.

[Krishnamurthy et al. 2016] Krishnamurthy, B., Puri, N., Goel, R.: "Learning Vector-Space Representations of Items for Recommendations using Word Embedding Models"; Procedia Computer Science Volume 80 Issue C, pp. 2205-2210, 2016.

[Cai et al. 2018] Cai, X., Gao, J., Ngiam, K. Y., Ooi, B. C., Zhang, Y., Yuan, X.: "Medical Concept Embedding with Time-Aware Attention"; Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, 2018.

[Mikolov et al. 2013a] Mikolov, T., Chen, K., Corrado, G., Dean, J.: "Efficient estimation of word representations in vector space"; ICLR Workshop, 2013.

[Mikolov et al. 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: "Distributed representations of words and phrases and their compositionality"; NIPS, pp. 3111-3119, 2013.

[Murray et al. 2011] Murray, R., Ryan, P., Reisinger, S.: "Design and Validation of a Data Simulation Model for Longitudinal Healthcare Data"; AMIA Annual Symposium Proceedings, pp. 1176-1185, 2011.

[Pennington et al. 2014] Pennington, J., Socher, R., Manning, C.: "GloVe: Global vectors for word representation"; EMNLP, pp. 1532-1543, 2014.

[Young et al. 2017] Young, T., Hazarika, D., Poria, S., Cambria, E.: "Recent Trends in Deep Learning Based Natural Language Processing"; arXiv preprint arXiv:1708.02709v4, 2017.