MACHINE LEARNING ENABLED VENTURE CAPITALISM

Ian Cheung[1] , Yigit Ihlamu[2]

Abstract

We used a BERT-based [1] embedder to embed a dataset of companies ("company universe") into a 768 dimensional vector ("representation") space. We then constructed a search engine within this space that queries the universe for best matches. We found that this performs at accuracy with reasonable speed. Additionally, we propose a training protocol that allows for model fine-tuning, which increases both query speed and accuracy for specific tasks.

Contents

1. INTRODUCTION

When there is a window of opportunity, entrepreneurs around the world seek the same opportunities–there is almost never only one company tackling a problem. For that reason, we built a software suite ("VelaTwins") that allows us to search a database for companies that are alike. This is particularly necessary for VC firms, who are always looking for (1) substitute investments and (2) opportunities to diversify within a particular market.

1.1. Data

Our dataset contains 655,000 companies and descriptions that are scraped from websites like *crunchbase*. These companies are split into around 55,000 unique categories, with a median of 10 companies per category. An example dataset is presented in figure (1).

[1] Keble College, Oxford
[2] Vela Partners

name	short_description	category_list	category_groups_list	description
Toolex Machine	Manufacturing & supplying Lathe, Boring,	Information Technology,Manufacturing	Information Technology,Manufacturing	Toolex Machine Tools Pvt Ltd Commenced in the ...
Perish the Thought Golf	Perish The Thought Golf is a company that deve...	Sports	Sports	Perish The Thought Golf is a company that deve...
Sell One Thing	A One Page Product Checkout Page	E-Commerce	Commerce and Shopping	Sell One Thing helps people sell stuff online....
Citeulike	Citeulike is a free online service to organize...	Education,Internet	Education,Internet Services	Citeulike is a free online service to organize...
Juick	Juick is a microblogging website dedicated to ...	Blogging Platforms,Information Technology,Mess...	Content and Publishing,Information Technology,...	IM-based social network and microblogging serv...

Figure 1: *Dataset*

1.2. Aims and approach

Given a particular company's description, we want to return a set of n other companies that are in the same business. Fundamentally, this problem consists of two different parts: an embedding process (section 2.2) to represent words as computer-readable vectors, and a search process (section 2.3). Fortunately, both of these problems have well-defined solutions. Since the introduction of word2vec [2] in 2013, word-embedding algorithms have been much improved. Currently, the most effective algorithm is widely recognised to be Google's BERT, which is a bi-directional transformer. However, BERT is extremely computationally expensive–for example, running BERT on our dataset (500 MB) on the author's computer would take up most of his productive years. Previous attempts at reducing BERT's runtime (e.g. reducing dataset size via heuristics) came at the cost of accuracy, as well as introducting human biases. For this project, we used the SBERT [3] variation. SBERT is a siamese network built on top of BERT that reduces runtime by nearly four orders of magnitude while presesrving BERT-level accuracy. For the search process, we use the industry standard cosine similarity to search the representation space for the n most similar companies to the input.

2. MODEL OVERVIEW

Figure 2 is an overview of the model architecture. The model is built around the SBERT embedder. For the purposes of demonstration we used the stock (pretrained) SBERT embedder, although `VelaTwins` is also capable of being re-trained on custom data (detailed in section 4), which would greatly increase the overall model accuracy. `VelaTwins` is also built with the settings configured as a shared-state, which allows for the model parameters to be individually and continuously reconfigured in one step. This allows for users to fine-tune the model, or indeed, to employ a reinforcement learn-

ing algorithm. This paper is also accompanied by a Jupyter notebook that demonstrates the capabilities of the model in detail on propietary data.

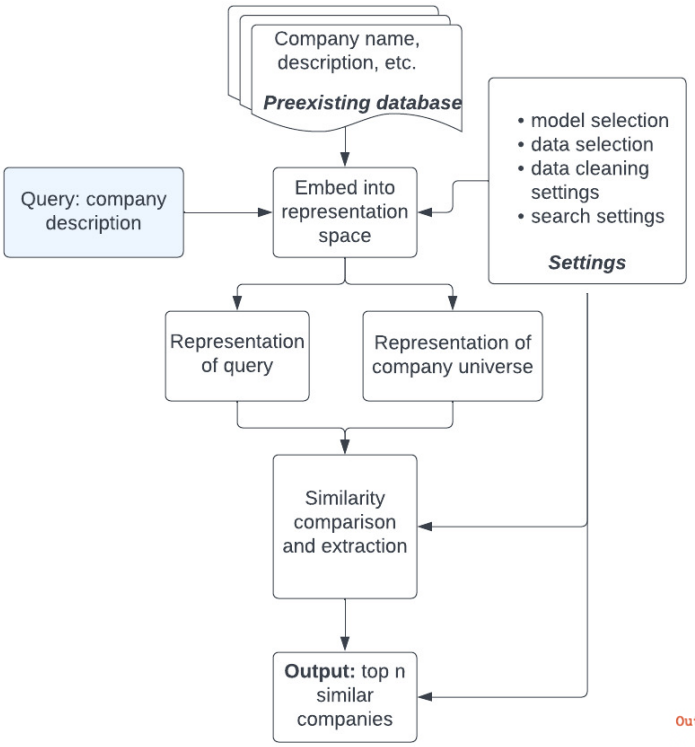

Figure 2: *Coarse model architecture*

2.1. Data Preparation

To prepare the dataset for our model, we use the module in `Data.config_data`. This takes as an input the raw data (in csv format) and returns a list of companies along with a description. Descriptions are cut off after a certain number of words, and "meaningless" words are also removed (these can be configured in `Settings.custom`). The processed data can then be fed into the embedding process (section 2.2).

2.2. Embedding Process

The embedding process in `Model.model` (the vertical flow chart in figure 2) should be run only once during set up. By default, this module uses the pre-trained SBERT embedder. However, custom embedders can also be used by passing them as keyword arguments (demonstrated in section 4). Stock SBERT embeds the list of companies and descriptions into a representation space of 768 dimensions (each company-description pair is a vector in this space). With the fine-tuned model in section 4, this is then passed through a fully-connected network and condensed down to 256 dimensions. This module outputs a $d \times n$ matrix where d is the dimension of the representation space and n is the number of companies in the company universe. This matrix is then saved for future use.

2.3. Search Engine

The search process is in `Query.query`. It takes a `string` input and embeds it into the same representation space as section 2.2 using `Model.model`. A cosine similarity matrix is then produced between the input and every element of the universe, and the top n scores are saved and the corresponding companies returned. `evaluate = True` can be passed in as a keyword argument, which tells the query model to log results that are above a certain cosine similarity threshold. Additionally, `evaluate` also returns the SBERT evaluation score, which takes in two "gold standard" calibration sentences and evaluates the model on whether the output is close enough to the calibration. The threshold and calibration sentences are configured in `Settings.custom`).

3. RESULTS

We ran the model on our propietary dataset of 655,000 companies. For demonstration purposes, we configured the model to return pairs of companies that are deemed to be correlated. Figure 3 shows one of these pairs. Note that these two companies are indeed very similar, and provide services that are almost identical.

```
Out[83]: array(['Removals Perth',
                'Friendly courteous furniture removalists in Perth WA, providing c
       areful time low cost removals so your move totally stress free. We expert
       removals packing professionals. Having moved mansions in Dalkeith units',
                'B Moved Removals',
                'Our specialty areas include furniture removals from ADELAIDE, MEL
       BOURNE, SYDNEY, BRISBANE, BUNDABERG, ROCKHAMPTON, MACKAY, TOWNSVILLE, CAI
       RNS. We offer professional prompt service, as well as excellent rates mov
       e office equipment furniture,'],
              dtype='<U339')
```

Figure 3: *Similar sentences*

For a randomly selected set of 50,000 companies in the company universe, we found 1010 company pairs that are "highly correlated", which we defined to be when the cosine similarity is over 0.65. This, combined with the two metrics in section 2.3, can be used for model evaluation. However, due to time restriction, we were not able to systematically investigate the effects of different model settings.

4. MODEL FINE TUNING

Users of `VelaTwins` would presumably like to use the model on their own datasets. However, datasets are sometimes dramatically different. As seen in section 3, stock SBERT misses out on many results. To address this, we developed a more customisable embedder built on top of SBERT. The embedder architecture is shown in figure 4 below. The first two modules within the model, BERT and the pooling algorithm, are effectively the same as stock SBERT. However, we attach a fully-connected neural network to the pooling module, which condenses the 768 dimensional vector space into 256 dimensions. This neural network can then be trained on a specific dataset using `Model.train_model` to form connections that were not immediately obvious

to the pre-trained SBERT. The addition of this dense layer seems to do well in adapting SBERT to specific use cases, although the user must be careful to avoid overfitting. The use of this function is strongly recommended if the user has enough training data (see section 4.1).

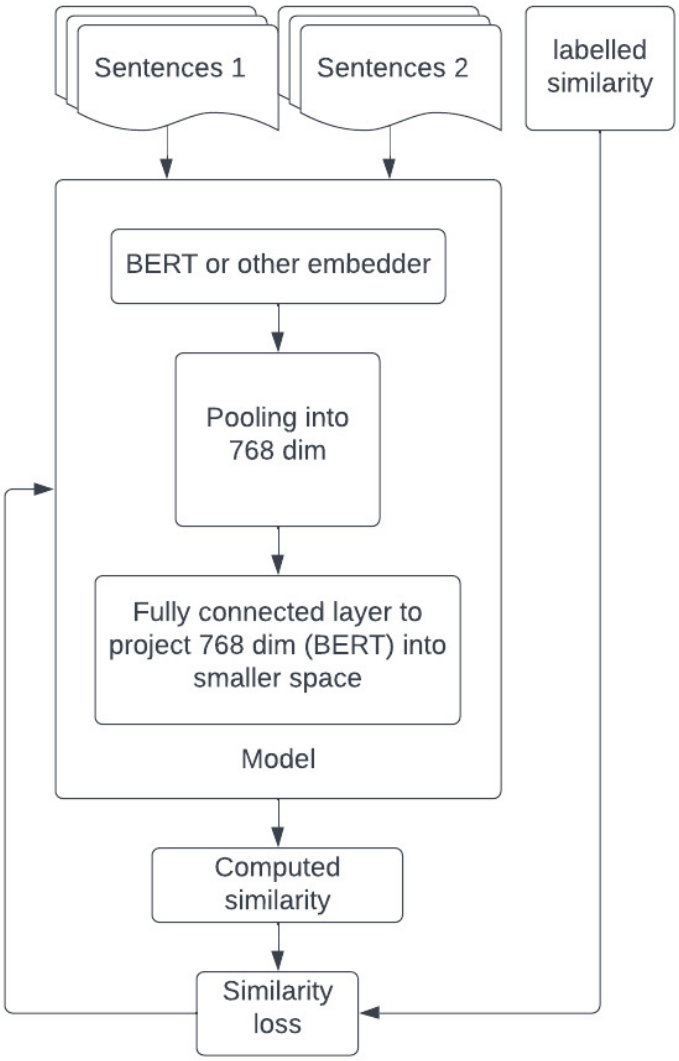

Figure 4: *Model training*

4.1. Training data

As with all simple neural networks, the fully-connected layer needs to be trained on labelled data. The specific dense layer in `VelaTwins` trains on pairs of companies (in the format of figure 3), along with a labelled cosine similarity. The network then compares company pairs and computes a similarity score, which is passed through to a similarity loss function along with the labelled similarity. This loss function is minimised during the training process.

After training, the model can then be saved and passed through `Model.model` to be used as before.

5. NEXT STEPS

Having built a proof-of-concept, we now look forward to the potential deployment of the model. However, before that can be done, there are a few issues we need to address.

- Data Processing: it is not yet obvious what the optimal length of the description is, what words should be allowed and what words shouldn't, and whether it is wise to pre-filter based on company categories.
- Fine-tuning: we currently don't have enough labelled data to properly train our fully-connected layer. We are currently in the process of devising a method to create a labelled dataset systematically.
- BERT selection: there are different versions of BERT that are trained on different datasets (e.g., wikipedia, encyclopedia, etc), and it is not yet clear which is optimal.
- Automatic data entry: as there are more companies being founded every day, we are looking to connect `VelaTwins` to automatic data collection software, so the company universe can continue expanding.

It is clear from this project, though, that the model architecture proposed in this paper can be used at scale.

6. REFERENCES

[1] Devlin et al., *BERT: Pre-training of Deep Bidirectional Transformers*, Google AI, 2019

[2] Mikolov et al., *Efficient Estimation of Word Representations in Vector Space*, Google, 2013

[3] Reimers and Gurevych, *Sentence Embeddings using Siamese BERT-Networks*, Technische Universitat Darmstadt, 2019