# CS 230: Classify Large Corporation's Industries based on their Descriptions to Identify Critical Investment Verticals

*King Castillo Alandy Dy*

Stanford
School of Engineering

## Abstract

I have a list of companies (not classified by vertical) and their investments. I also have another list of companies with just their description and classification. I want to find out which verticals energy companies are investing in to say where corporations foresee the future of the industry. Are they investing in companies in the same verticals, complimentary verticals or completely different verticals.

## Training Data

| | description | energy |
|---|---|---|
| 4.0 | The Company is a France-based electricity prod... | 1 |
| 5.0 | The Company was created on November 27, 1962 I... | 1 |
| 6.0 | The Company is a distributor of natural gas. N... | 1 |
| 7.0 | The Company specializes in electricity and gas... | 1 |
| 8.0 | RWE is the electricity and gas companies. Thro... | 1 |

| energy | tokenized |
|---|---|
| 1 | [the, compani, france-bas, electr, producer, ,... |
| 1 | [the, compani, creat, novemb, 27, ,, 1962, it,... |
| 1 | [the, compani, distributor, natur, gas, ., net... |
| 1 | [the, compani, special, electr, gas, product, ... |
| 1 | [rwe, electr, gas, companies, ., through, expe... |

We removed stop words, stemmed the words and tokenized the data. After processing, it looks like this: "compani creat novemb 27, 1962 it oper seven divisions. the sale segment focus sale electr gas product servic end users…"

The training data for the classifier is from OrbisResearch while the unclassified data with companies' investments is by Pitchbook. (18, 677 labeled examples and 49,685 unlabeled examples)

## Experiment Setup

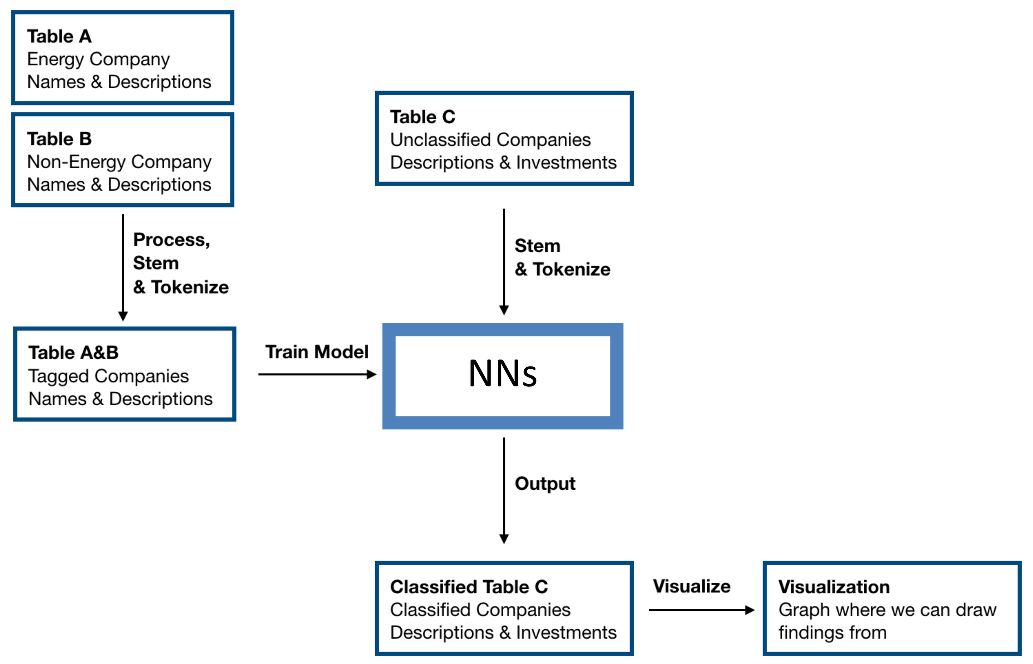


Table A - Energy companies and their descriptions
Table B - Non-energy companies and their descriptions
Table C - Unclassified companies with their descriptions and investments

Features:
We used TFIDF to get the features. The size of the vector is 300. I tried it at different levels and above 300 were marginal improvements so I decided to stick it out with 300. The vocab size is 130,977 for the entire dataset and it was 98,349 for the training set..

Explanation:
I ran both regular classifiers and also neural nets just to see how different it exactly could be. Of course, the neural nets would perform significantly better.

## My work

1st model:

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 128)               38528
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 129
=================================================================
```

I experimented with different dropout rates here. Even at 0.5, there was a significant difference between training accuracy and testing accuracy so I was overfitting to the data. When I increased the dropout rate all the way to 0.8, then the difference became more reasonable.
I used sigmoid since it was a binary classification problem.

I performed mini-batch gradient descent with a batch_size of 40 and at 30 epochs. The optimal accuracy was hit by the 3rd and 12th epoch so it wasn't necessary but I let it run since it was going pretty quickly anyways.

With all of this we were able to achieve:
Training accuracy: 0.9788
Testing accuracy: 0.9711

| Classifier | Acc: Train Set | Acc: Test Set |
|---|---|---|
| Logistic Regression | 97% | 95.1% |
| SVM (Linear SVC) | 96% | 95.7% |
| Ridge Classifier | 96% | 95.7% |
| Passive-Aggressive | 98% | 96.2% |
| **1st Model** | **97.9%** | **97.1%** |
| **CNN** | **99.7%** | **97.3%** |

**Results:**
As seen in the table to the right, we can see that most of these classifiers operated with more or less the same accuracies and F-Scores. We used a training set sized at 95% (17,743) and a test set sized at 5% (934).

## Machine Learning

2nd model:

```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_4 (InputLayer)            (None, 53)           0
__________________________________________________________________________________________________
embedding_4 (Embedding)         (None, 53, 300)      29504700    input_4[0][0]
__________________________________________________________________________________________________
conv1d_13 (Conv1D)              (None, 51, 128)      115328      embedding_4[0][0]
__________________________________________________________________________________________________
conv1d_14 (Conv1D)              (None, 50, 128)      153728      embedding_4[0][0]
__________________________________________________________________________________________________
conv1d_15 (Conv1D)              (None, 49, 128)      192128      embedding_4[0][0]
__________________________________________________________________________________________________
max_pooling1d_13 (MaxPooling1D) (None, 17, 128)      0           conv1d_13[0][0]
__________________________________________________________________________________________________
max_pooling1d_14 (MaxPooling1D) (None, 16, 128)      0           conv1d_14[0][0]
__________________________________________________________________________________________________
max_pooling1d_15 (MaxPooling1D) (None, 16, 128)      0           conv1d_15[0][0]
__________________________________________________________________________________________________
concatenate_4 (Concatenate)     (None, 49, 128)      0           max_pooling1d_13[0][0]
                                                                 max_pooling1d_14[0][0]
                                                                 max_pooling1d_15[0][0]
__________________________________________________________________________________________________
dropout_6 (Dropout)             (None, 49, 128)      0           concatenate_4[0][0]
__________________________________________________________________________________________________
flatten_4 (Flatten)             (None, 6272)         0           dropout_6[0][0]
__________________________________________________________________________________________________
dense_11 (Dense)                (None, 128)          802944      flatten_4[0][0]
__________________________________________________________________________________________________
dense_12 (Dense)                (None, 1)            129         dense_11[0][0]
==================================================================================================
Total params: 30,768,957
Trainable params: 1,264,257
Non-trainable params: 29,504,700
__________________________________________________________________________________________________
```

People often assume that deeper neural networks will generally perform better. In this case, it is definitely true. It was much slower but also quite significantly more accurate. I mimicked https://arxiv.org/pdf/1408.5882.pdf Yoon Kim's CNN but I used sigmoid for the last sense layer because it is a binary classifier and I had more than 1 dense layer.

These changes were made through trial and error and thankfully brought us to an even higher accuracy than I thought was possible. At the moment, I am still overfitting a bit but I increased the dropout to 0.6. This allowed us to achieve an accuracy of 0.9732.

Discussion:
To the right is the visualization of our results, energy companies invest the most in productivity software, energy production and alternative energy equipment verticals the most. It makes sense that energy incumbents would primarily invest in energy production and alternative energy equipment but it was surprising to see many of them making investments in the productivity software space. Our results were really good as you can see with the accuracy of the various classifiers which we used.

Related Work:
There is a lot of work related to the space of classifying words based on just text. For example, a paper that really influenced my approach was Convolutional Neural Networks for Sentence Classification by Yoon Kim. In fact, the CNN I used in my second attempt is pretty much the same but with an extra layer added on given the complexity inherent in the problem I am working on. For this problem specifically as I detail later, there is a lot of patterns that need to be caught between words far away from each other and there are multiple ways in which every word may be used so seeing broader patterns is important. Other works that are relevant are included as references.

Future:
If I had way more time, I would use something similar and just have multiple NN like the first model feed into a multi-class softmax layer. That way I'll be able to classify all types of businesses and see where investment is going. Then I'll start a VC fund based off of this.