# Learning from data: Assignment 3

**Emilio Oldenziel**
s2509679
e.oldenziel.1@student.rug.nl

## 1 Introduction

In this report we discuss two experiments, the first one is to find word similarities and analogies using word embeddings. In second experiment word embeddings are used to train a classifier which predicts a word's category out of 6 classes.

## 2 Data

In the experiments a dataset is used, this dataset contains 35.5k words (instances). Each instance has a labels which represents the word category, the category classes are; geo-political entity (GPE), location (LOC), person (PERSON), organisation (ORG), date (DATE) and cardinal number (CARDINAL). A slice of the dataset can be observed in Table 1

| Word | Label |
|---|---|
| Pele | ORG |
| year | DATE |
| Madrid | GPE |
| Barcelona | GPE |
| Revaldo | PERSON |

Table 1: A slice from the dataset

A second set of labels is introduced by reducing the 6 labels to 2 classes which are non-location and location. The labels GPE and LOC are mapped to a location label and PERSON, ORG, DATE, and CARDINAL are mapped to a non-location label. As can be observed in Table 2 and Table 3 the distribution of instances per class is not equal in any of the classes. The supplied labels even contain a maximal unbalance between GPE and LOC of factor 18, which is extremely unbalanced. The middle range containing ORG, PERSON, CARDINAL and DATA is also not balanced but only has a maximum imbalance below factor 2. The reduced labels only consist of 2 classes with an imbalance of almost to factor 2.

| Label | Count |
|---|---|
| GPE | 11392 |
| ORG | 8131 |
| PERSON | 5955 |
| CARDINAL | 5291 |
| DATE | 4213 |
| LOC | 613 |

Table 2: dataset distribution for original label

| Label | Count |
|---|---|
| NON-LOCATION | 23590 |
| LOCATION | 12005 |

Table 3: dataset distribution for reduced labels

## 3 Methods

To create a model that can classify reviews we first have to assemble a few steps.

### 3.1 Word embeddings

Instead of using a word as input for the classifier, we use the pre-computed embeddings as feature vectors, these embeddings where taken from Standfords GloVe project and contains global word vector with a dimensionality of 50.

### 3.2 Classifier

The experiments where performed using a Neural Network (Perceptron) classifier. The network consists of 1 layer with $C$ units and takes a feature vector as input.

### 3.3 Validation

A K-fold cross validation was used to validate the model. To score the predictions made by the model, the class-wise and average; recall, precision and f-1 scores were calculated after the training process in each fold. After all folds, the confusion matrix for all K folds are displayed in an image.

### 3.4 Parameters

To be able to obtain the best model, the (hyper) parameters of the pre-proccessing, classifier and validation have to be tuned.

- Optimiser: The optimiser function is used to reduce the loss in the network and calculate the approach to reach a optimal solution is version space. Stochastic gradient decent is most commonly used as optimiser which we also use in the experiments.

- Learning rate $\eta$: the learning rate $\eta$ controls how fast the network learns in the range $0 \leq \eta \leq 1$ where 0 is no learning and 1 is learn as fast as possible. In this experiment we chose learning rates $\eta \leq 0.1$ to get a non-overshooting but and fast learning process.

- Activation function: The activation function determines how the cell 'fires'. In the experiments we used a rectified linear unit (ReLU) defined as $max(0, x)$ as activation function since it learns fast by not damping positive values.

- Loss function: The loss is function is a metric to obtain the distance between the truth labels and the predicted labels. In the experiment we use the mean squared error because it mkaes the learning process faster than the other functions.

- Epochs: The amount of sweeps over the whole dataset. The network is improved by more sweeps over the dataset but too many sweeps can cause the network to go into a overfit.

- Batch size: is the amount of instances that are used for a gradient step. The default Keras value is 32, the higher the number is the less time it takes to train the network, but with a low batch size the updates are more gradient, a low batch size therefore mostly yields a better classification score.

## 4 Results

### 4.1 Similarities

| **Word** | **Distance** |
|---|---|
| shemales | 0.512174 |
| pamela_anderson | 0.508379 |
| sucking_dick | 0.499958 |
| paris_hilton | 0.499344 |

Table 4: Odd results for the word: asian

| **Word** | **Distance** |
|---|---|
| factory | 0.6708804 |

Table 5: A result for the word: plant

| **Word** | **Distance** |
|---|---|
| :-) | 0.556652 |

Table 6: A result for the word: google

| **Word** | **Distance** |
|---|---|
| brennan | 0.613305 |
| dpa_sch | 0.611703 |
| holl | 0.611221 |
| newsdesk@afxnews.com_gl | 0.608715 |
| ar_ar | 0.607523 |
| susie | 0.605760 |

Table 7: Results for the word: emilio

| **Word** | **Distance** |
|---|---|
| balmy | 0.731459 |
| sunshine | 0.725109 |
| overcast | 0.691810 |
| breezy | 0.688585 |
| gloriously_sunny | 0.677916 |
| sunny_skies | 0.668846 |
| chilly | 0.668503 |

Table 8: Results for the word: sunny

## 4.2 Word analogies

| Word | Distance |
|---|---|
| bananas | 0.587556 |
| pineapples | 0.461547 |
| peanut | 0.449559 |
| potato | 0.443853 |
| papaya | 0.440213 |
| apples | 0.433603 |
| ... | ... |

Table 9: pear apple banana

| Word | Distance |
|---|---|
| thursday | 0.609738 |
| friday | 0.537440 |
| saturday | 0.524219 |
| feb | 0.510410 |
| tommorow | 0.496246 |
| sunday | 0.494972 |
| july | 0.490566 |
| ... | ... |
| pompeycarpet | 0.416670 |
| oday | 0.413210 |
| carlos | 0.410290 |
| ... | ... |

Table 10: monday tuesday wednesday

| Word | Distance |
|---|---|
| dvd_burner | 0.603056 |
| thinkpad | 0.573350 |
| asus | 0.565046 |
| toshiba | 0.554476 |
| cpu | 0.545561 |
| macbook_pro | 0.539917 |
| compaq | 0.539624 |
| vaio | 0.534460 |
| ... | ... |

Table 11: apple dell lenovo

## 4.3 Classification

| Labels | Set | Accuracy |
|---|---|---|
| 6-class | Training | 0.8461 |
| 6-class | Test | 0.8445 |
| Binary | Training | 0.9414 |
| Binary | Test | 0.9411 |

Table 12: Average accuracy for with $\eta = 0.01, epochs = 50, batchsize = 32, folds = 5$

| Word | Predicted label | Similar words | Correct label |
|---|---|---|---|
| groningen | PERSON | amsterdam, berlin | LOC |
| oneplus | DATE | apple, samsung | ORG |
| merkel | LOC | obama, bush | PERSON |

Table 13: Predictions for out of vocabulary words

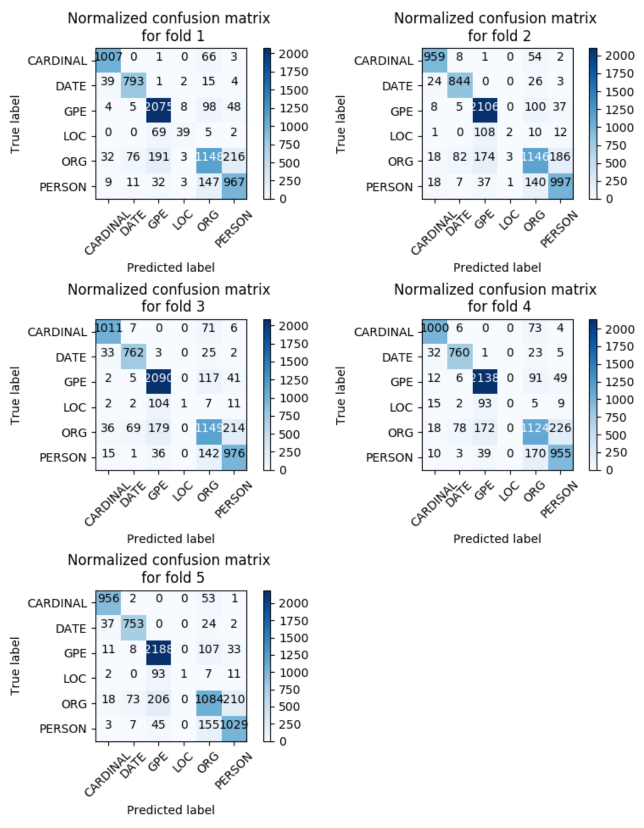

Figure 1: Confusion matrix for each fold, $\eta = 0.01, epochs = 50, batchsize = 32, folds = 5$.

## 5 Discussion

From the results of the experiments we can observe that word embeddings are a powerful method to find similar words or word analogies. In some cases the similarity results are not very accurate or evident, as we can in Table 4, Table 5 and Table 8. Besides expected similarities, like european, american, african and country names for the word asian. It also returns odd words as listed in the Table 4 that are used in the same way but are not very evident. For the word plant one would expect to find similar words like flower, tree and grass. Instead, similar words to the meaning of plant as being a working site are returned which is a homonym. For the word google we find a lot of relevant words and an unexpected smiley :-). When testing my name, I would expect a lot of other names, which is also the case but also email addresses containing names and web urls. As expected, we find weather forms for the word sunny without any unexpected behaviour. It is very likely that the accuracy for similarities is better for words that are very common such as the word sunny because the embedding is based on more evidence.
The word analogy seem to perform quite well on the chosen words, especially on very common words like fruits as can be observed in Table 9. Less common analogies like apple, dell, lenovo return a lot of other computer manufacturers but also devices and components like dvd burner and cpu.
In Table 12 we can observe the accuracy scores for the two classification problems, as expected the accuracy of the binary problem is higher than for the 6-class problem. Besides that the binary problem has more samples per class, the balance is also significantly better. The confusion matrix shown in Figure 1 shows clearly that the confusion for location is extremely high, in fold 4 even no instances with label LOC were correctly classified. The cause for this is the words are represented in a vector space that is not linearly separable by a hyper plane learned by the perceptron. Therefore, the decision boundary is optimised for the class GPE which yields less loss in the network and causes a lot of incorrect classifications for class LOC. To investigate the model's performance on words that are not in the vocabulary we chose 3 words to be predicted by the model, the results can be observed in Table 13, we can see that all words where not correctly classified. The incorrect classification is possibly caused by the words not having any part that can be associated with words of the correct class. It even possible that the word merkel is associated by the word merkelbeek which is a town in The Netherlands and therefore classified as LOC.

## 6 Conclusion

In these experiments we showed that word embedding is a useful method to be used in word classification, although it is weak against words that do not occur in the vocabulary. A percepton classifier is also not able to successfully separate the classes. A larger vocabulary and a multi-layer perceptron can improve the results.