Exploring the possibilities of Word Embeddings & Perceptrons

Richard Scholtens
s2956586

Abstract

This research examines the possibilities of word embeddings and neural networks. To do so it explores the possibilities of the word2vec tool and tries to combine word embeddings with a neural network classifier. The word2vec tool shows promising results when it comes to semantic and syntactic relations. By combining the power of word embeddings with a neural network one must conclude it outperformed the baseline in binary classification by 37% and multi classification by 42%.

1 Introduction

Many new technologies have been developed in the computer era. Some of these technologies are based on comprehensive programming algorithms which can be used as tools for prediction. One of the techniques used is the neural network which is inspired by the structure of the brain. Unlike machine learning, the neural network contains highly interconnected entities, called units or nodes. The neural network is one of the deep learning technologies which lies its focus on solving complex processes with the help of a group of algorithms which model the data whereas machine learning lies its focus on learning from parsed data. In short this means that traditional machine learning techniques need guidance to predict their accuracy while neural networks can define this for them self. This means the computer can make conclusions without the guidance of a human to tell what is right and wrong. Before on can fully grasp the potential of neural networks this research one must understand the basic principles. Therefore, in this report neural networks will be explained by discussing the topics of perceptrons and word embeddings. This research will also discuss some of the possibilities the Keras library provides. Keras is a high-level neural networks API, written in Python which was developed with a focus on enabling fast experimentation.

2 Data

This research will be using a set of Named Entity Disambiguation data, extracted from the OntoNotes corpus. It consists of single-word named entity (NE), labelled as belonging to one of six different categories: geo-political entity (GPE), location (LOC), person (PERSON), organisation (ORG), date (DATE) and cardinal number (CARDINAL).

NE	Category
two	CARDINAL
Norway	GPE
Scandinavia	LOC
Monday	DATE
BART	ORG
Clinton	PERSON

Table 1: Sample data

3 Method/Approach

3.1 Word Embeddings

Word embeddings is a technique which allows the mapping of a set of words or phrases in a vocabulary. These mappings are then transformed to numerical values inside a vector. This technique can help to grasp the context of a word in a document. Semantic and syntactic similarity and relations to other words can also be retrieved with the help of these vectors.

3.1.1 Similarities

To understand how word embeddings work, this research will demonstrate how these representa-

tions work with the help of the word2vec tool provided by the Rijksuniversiteit Groningen. Five words will be thrown into the word2vec tool to see if anything unexpected comes up. These words are: "Feyenoord, mouse, toothpick, furniture, and house". For the Feyenoord it is expected to find other soccer associations or Rotterdam. For the word mouse is expected to find words like cat, dog or other animals. When thinking of toothpick, one thinks of toothbrush and dental floss. Furniture can be similar to chair and couch. For house one can think about a roof, walls or door.

3.1.2 Novel analogies

It is also possible to find relations between three sets of words. For instance, when we try king, man, and woman the most likely word to retrieve is queen. In order to fully understand the outcomes of the analogies this research will examine the outcomes of the following three analogies. For this will use the following:

```
man, boy, woman
father, brother, mother
Germany, Berlin, Belgium
```

3.2 Using a perceptron for NE disambiguation

A perceptron (or multi-layer perceptron) is a neural network in which the neurons are connected to each other in different layers. A first layer consists of input neurons, where the input signals are applied. Then there are one or more hidden layers that provide more intelligence and finally there is the output layer, which displays the result of the perceptron. All neurons of a certain layer are connected to all neurons of the next layer, so that the input signal propagates through the different layers.

3.2.1 Baseline

A baseline is required to measure the performance of our classifier. The dummy classifier retrieved from Scikit Learn (Pedregosa et al., 2011) is used as a baseline. The parameter is set to the stratified strategy which generates predictions by respecting the training set's class distribution.

3.2.2 Perceptron Hyperparameters

To further understand how the perceptron classifier works one must see what the effects are when changing the hyperparameters. The classifier holds multiple hyperparameters which all can affect the outcome. This research will only discuss the epoch, batch size, learning rate and activation function.

The number of epochs stands for the number of times the classifier examines the entire data set. This means that for every time an epoch has been completed the algorithm has seen all samples in the dataset one time. The classifier needs re-examine the data set multiple times to avoid underfitting. However, a higher number of epochs might cause overfitting. To avoid this one must look closely at the validation and training error. When this number keeps on dropping it is better to up the number of epochs.

However, the classifier cannot take in the data set at once. To work around this problem, it is possible to change the batch size. The total number of samples will be divided by the number for the batch size. This means if the number of samples is ten and the batch size is two there are five batches. If the number of epochs hold three the algorithm will run five batches three times which leads to fifteen iterations. This means that five iterations of a batch are equal to one epoch.

In order to learn the classifier how to respond the estimated error the learning rate hyperparameter is used. This hyperparameter controls how much the classifier changes the weights according to the response of the estimated error. A low learning rate can cause the training process to stop or take a large amount of time. If set to high it might result into insignificant set of weights or lead to a shaky training process.

It is possible to change the output of nodes from the classifier. For this the activation function can be used. The activation function is either linear or non-linear. This research will only discuss the Linear, Sigmoid and Tanh function. The Sigmoid and Tanh function are both logistic. The linear function is not limited to any range while the Sigmoid function maps the resulting values zero to one. However, the Than function maps the values between minus one to one. When looking at the linear function it performs less than the other functions because it is less compatible to the complex hyperparameters of the classifier. The Sigmoid function tries to predict the outcome based on probabilities. Because the values are mapped between zero to one it is easier to see where the slope starts and therefor easier to see what the best settings should be. The Than function is compa-

rable to the Sigmoid function with the advantage that is the values can also be mapped as negatives whereas the Sigmoid function would map these as zero. This means more information can be derived from the Than function.

3.3 Word Embeddings and Generalization

To get a better understanding of word embeddings this research will examine how generalizations work. By having the classifier classify a word that does not occur in the training data but is rather similar it is possible to see how the generalization process. For this examination the research will limit itself to multi classification and the following words, namely Antwerpen, KFC and thirteen. These words do not appear in the training data but are related to similar words in the data set. These are for Antwerpen: "Amsterdam (GPE), Paris (GPE), and Madrid (GPE)". For KFC: "McDonalds (ORG), Dominos (ORG), and Starbucks (ORG)". For thirteen the following words are similar: "ten (CARDINAL), eleven (CARDINAL), and twelve (CARDINAL)". By comparing the output label of the classified word with that of the similar words it is possible to see how well the system performs. The perceptron classifier will be used for the labelling. Epoch will be set to ten, batch size to 32 and the activation function to Tanh. The most common label for a word will be assigned.

3.4 Error analysis

To see where confusion within the algorithm lies it is possible to create a confusion matrix. This will allow us to see if the labels are predicted correct or where it is predicted wrong. For this the confusion matrix from SciKit Learn (Pedregosa et al., 2011) will be used.

4 Results

4.1 Word2Vec similarities

The following words were used in the word2vec tool: "Feyenoord, mouse, toothpick, furniture, and house". As was expected other soccer associations like NEC Nijmegen, Sparta Rotterdam and Ajax Amsterdam were found. For mouse however the results were very different. It was expected to be considered as animal but due to technological advances like the computer mouse words like keyboard and computer brands were mentioned. The toothpick also did not give the expected outcomes. Instead it defined other types of picks. However, furniture did give the expected outcomes. House gave more synonyms like bungalow or apartment.

4.2 Analysing analogies

This section will examine the outcomes of certain analogies. The earlier mentioned analogies will be examined. These are the results:

```
man, boy, woman = girl
father, brother, mother = son
Germany, Berlin, Belgium = Austria
```

As can be seen only the first analogy has an outcome one can agree with.

4.3 Baseline comparisation

To see how good the perceptron classifier performs its performance is compared to the score of the baseline. The perceptron classifier has a F1-score of 92% in the binary classification performs 37% better than the baseline. However, when applied on the multi classification the perceptron classifier holds an F1-score of 64% which is 42% better than the baseline. Comparing the F1-scores of the baseline and perceptron classifier leads to the conclusion that the perceptron classifier obtains the best results on multi classification. However, comparing the F1-scores of the perceptron classifier for binary and multi classification it performs 28% better on binary classification. When considering these results one must conclude that the perceptron classifier does perform rather well.

Classifier	Precision	Recall	F1	Acc
Baseline	0.55	0.55	0.55	0.33
Classifier	0.92	0.91	0.92	0.91

Table 2: Results of binary classification

Classifier	Precision	Recall	F1	Acc
Baseline	0.22	0.22	0.22	0.10
Classifier	0.64	0.65	0.64	0.65

Table 3: Results of multi classification

4.4 Changing perceptron hyperparameters

The figures 1 and 2 shows how accuracy and loss respond when changing the number of epochs. As you can see, when the number of epochs increases the loss drops and the accuracy increases until a certain number of epochs. One can see that setting the epoch hyperparameter to three will suffice for the binary classification. The multi classification is best set to five. By increasing the number of epochs one is sure to be overfitting the classifier.

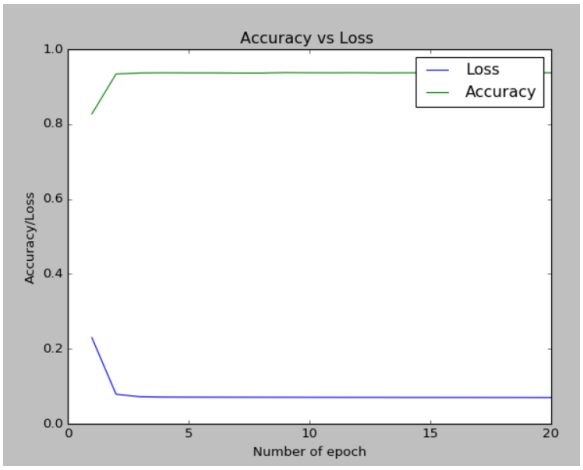

Figure 1: Binary classification increase of epochs.

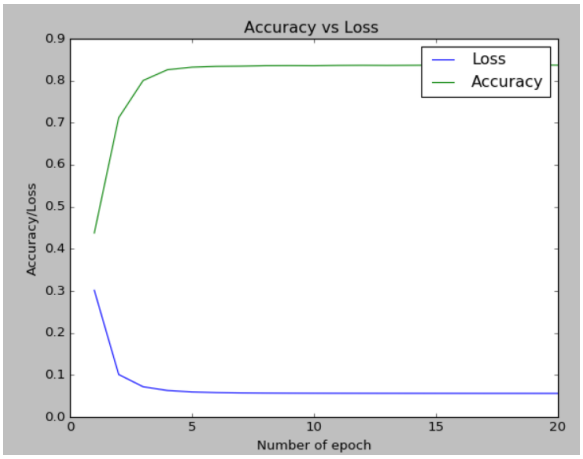

Figure 2: Multi classification increase of epochs.

Figures 3 and 4 show what happens when changing the batch size with epoch set to one. After a certain amount one can see that it leads to no differences in the outcome. For the batch size the optimal size would be around the 15 samples per batch according to the graphs.

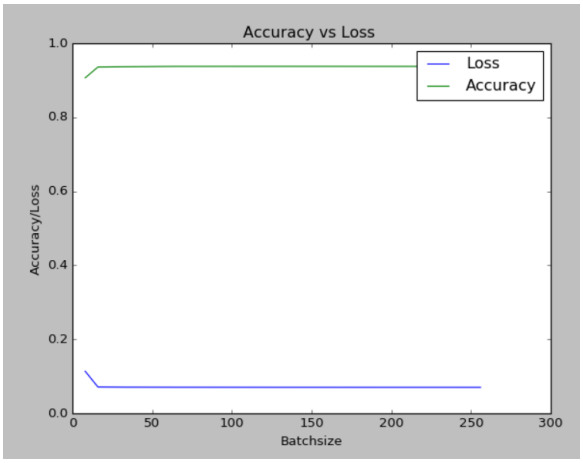

Figure 3: Increasing batchsize binary classification.

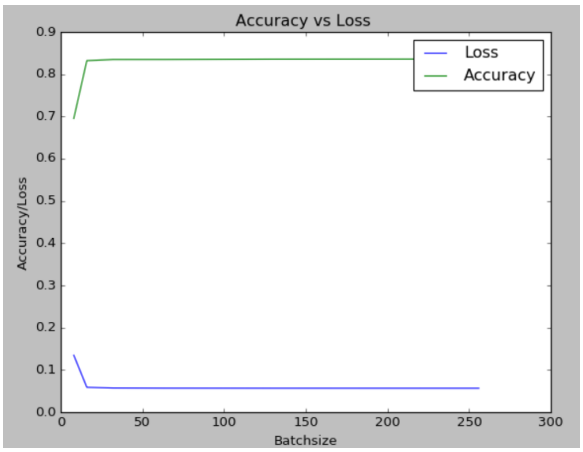

Figure 4: Increasing batchsize multi classification.

In figure 1 and 2 the linear function was used. Figure 4 and 5 contain the Sigmoid function and figure 6 and 7 the Tanh function. One can see that the Sigmoid function loss almost looks the same as the loss for the linear function. However, the accuracy outperforms the accuracy for the linear function by far. The Tahn function outperforms even the Sigmoid function as was expected.

4.5 Generalizing words

It is possible to predict the labels for words which are not included in the training data. Earlier three words were chosen which do not appear in the training data, namely Antwerpen, KFC and thirteen. It is expected that Antwerpen will get la-

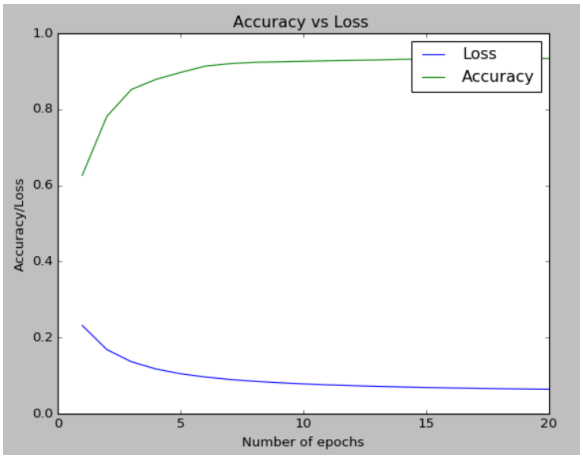

Figure 5: Using Sigmoid binary classification.

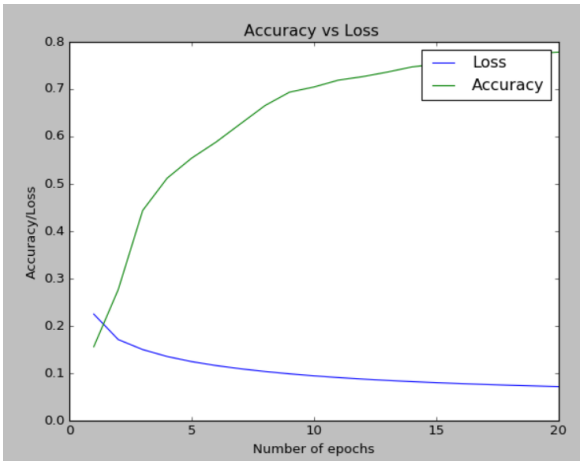

Figure 6: Using Sigmoid multi classification.

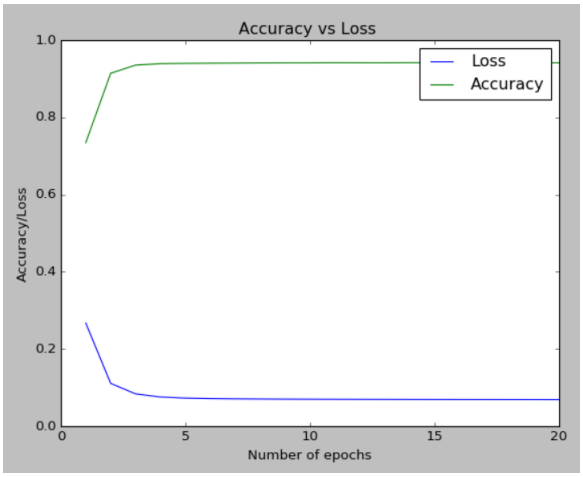

Figure 7: Using Tanh binary classification.

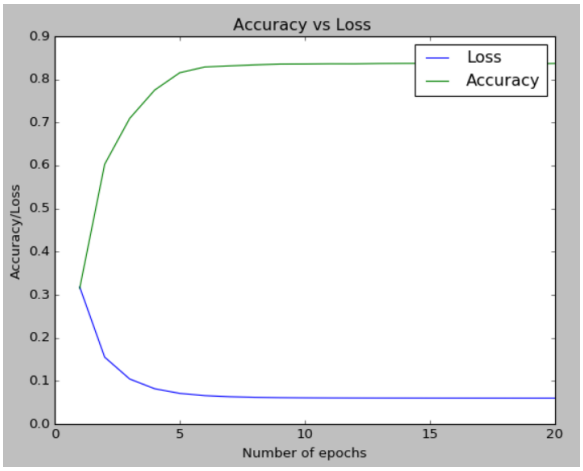

Figure 8: Using Tanh multi classification.

belled as GPE, KFC as ORG, and thirteen as CARDINAL. The results are shown in the table 4. As one can see these words were all incorrectly labelled. It seems that algorithm failed in its generalization task.

Word	**Predicted label**
Antwerpen	ORG
KFC	Cardinal
thirteen	ORG

Table 4: Predicted word labels

4.6 Confusion matrix

As can be seen in the confusion matrix the label LOC was not predicted at all. However, it seems the test data contained many instances which was correctly predicted as the label GPE. CARDINAL, ORG, and PERSON were also correctly predicted for the most part. This means most of the errors must come from the potential LOC instances. DATE also was also predicted many times as ORG. This could be improved also.

t/p	CARDINAL	DATE	GPE	LOC	ORG	PERSON
CARDINAL	1189.0	15.0	9.0	0.0	96.0	2.0
DATE	46.0	936.0	5.0	0.0	29.0	1.0
GPE	6.0	6.0	2771.0	0.0	80.0	52.0
LOC	12.0	0.0	142.0	0.0	16.0	7.0
ORG	52.0	131.0	327.0	0.0	1355.0	207.0
PERSON	22.0	15.0	63.0	0.0	221.0	1086.0

Figure 9: Using Tanh multi classification.

5 Discussion/Conclusion

This research set out to examine the potential of perceptrons and word embeddings. Word embed-

dings have opened a path on which one can predict multiple outcomes based on cosine similarity. Semantic and syntactic values can be found quite easily with the help of these embeddings as could be seen when examining the word2vec tool. These embeddings will help machine learning techniques to develop new algorithms to predict. In combination with perceptrons within neural networks one can see how well it performs when compared to the baseline. A perceptron classifier has outperformed the baseline for binary classification by 37% and multi classification by 42%. One must conclude that neural networks are highly efficient when combined with word embeddings. Further research could try to include other hyperparameters in order to obtain even better results.

References

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.