Learning from Data - Assignment 4 - Perceptrons and Word Embeddings

Remko Boschker
master student of information science at the Rijks Universiteit Groningen
s1282603, r.boschker@student.rug.nl

Abstract

In the following I use the word2vec toolkit and evaluate some of my findings. I also use word embeddings as features for a perceptron classifier. I develop a baseline classifier for comparison and tune the parameters of the algorithm. The perceptron performs a binary classification at an f-1 and accuracy score of 0.91 and a six-way classification at scores of 0.80. The parameter tuning shows that the parameters settings are interrelated and I found no clear optimum. Finally I look into the misclassifications and find items are often mis-categorised as organisations possibly because they occur in many contexts and a perceptron is a linear classifier.

1 Introduction

The use of word embeddings in natural language processing tasks has become successful due to the availability of large amounts of data, efficient pre-training of representations and its effectiveness. In the following experiments I first use the word2vec toolkit to look at words with similar embeddings and to find analogies between sets of words. Secondly I combine word embeddings as features with a perceptron classifier to categorise named entities.

2 Word2vec

I use the word2vec toolkit to experiment with word embeddings. The toolkit uses the *GoogleNews-vectors-negative300bin* pre-trained word embedding. First I use the distance tool with five words that have strong connotations or more than one meaning. I discuss any unexpected results and go a little into word sense ambiguity. Next I use the analogy tool to evaluate word2vec ability to capture analogies.

The distance tool lists the forty words that are closest to the word entered in terms of the cosine distance between vector representations of the words embeddings. Below I list and discuss the results for five words.

Java I expected there to be words related to the programming language and the Indonesian island. And there were for instance *Jakarta* and *J2EE*. I did not expect to find *Jumpin Juice*, a coffee and juice bar franchise. The franchise advertises a lot with the word *Java* as a reference to coffee.

Python For the word *python* I expected to also find word related to the snake and to the programming language. But I only found names of snakes and one or two other reptiles.

Dutch I expected to find words typically associated with being *Dutch* (at least in an American news corpus) such as cheese and tulips and coffeeshops. But I found mostly other nationalities such as *Belgian, German* and *Netherlands*. Also there were two Dutch newspapers *Volkskrant* and *Telegraaf*. Probably because they were quoted in the corpus. And a financial analyst and a Dutch spokesman for the IS UN taskforce *analyst_Pieter_Kulsen* and emphEdmond_Messchaert_spokesman.

economy Besides the expected terminlogy such as *job_rate, gross_domestic_product, recession*, I found an apparently very common spelling error *theeconomy*, a large investment firm *EFG_Hermes_Ziada* and the mysterious *Wehbey*, the last name of various government officials in the field of economics in Central and South America.

New York For *New York* I found the five boroughs *Manhattan, Brooklyn, Queens, Bronx, Staten Island* and *Long Island* as expected. I also found baseball related terms *RBI_ARodriguez, STRIKEOUTS_Mussina, STOLEN_BASES_Reyes* to do with the baseball team the New York Yankees. Unexpectedly I also found a New York Times photographer and journalist *Hiroko_Masuike, By_PENELOPE_GREEN*. These are probably the result of the news based corpus used for training.

In general it is to be expected that the embeddings for a particular word reflect the different meanings of a word. Whereas polysemous words for example *man, mole, bank, point, roll, wave* can be expected to have similar embeddings because these are words that have multiple but related meanings. For homonymic words such as *arm, bank, match, iron, bear* the different meanings are not related and different groups of related terms can be discerned.

Word sense disambiguation is the problem in natural language processing of selecting which of a words sense is the one intended in a sentence when there is more than one. Word embeddings are not a good feature for this task, because although the embeddings show words related to the different senses of a word it does not distinguish between them. And therefore given a context you might find words that are also in the embedding, but you have no way to derive the particular sense from that.

Also the word senses present in an embedding is influenced by the corpus uses for training the embedding. For instance the Google News vectors shows only words related to *cookie* as a baked good such as *peanut_butter_cookie, cupcake, oatmeal_cookie*. The alternate 'vectors.bin' training file also shows words related to the sense of a *cookie* that is set in a webbrowser such as *url, server, and password*. Apparently the vectors.bin vectors were trained on a corpus containing relatively more texts about internet and web-browser related content.

I also use the analogy tool. This tool can given three term find an analogues fourth by adding the first two vectors and subtracting the third. For instance $man + king - woman = queen$. I gave it a few tries before I found some relations that worked. For instance $grass + green - roses$ did not equal red. The *president/prime-minister* and *country* relation works well although the corpus is a bit dated.

France Hollande Germany Schröder

France Hollande Bolivia Chavez

Bolivia Chavez Spain Zapatero

Spain Zapatero Netherlands Balkenende

I expected *sporter* and *sport* to be an easy analogy. But when trying *Kramer skating Armstrong*, I only found cycling in third place. As it turns out there is also a skateboarder and another ice skater named *Armstrong* accounting for first and second place.

Kramer skating Armstrong skater

Messi Soccer Nadal tennis

Nadal tennis Cruijff soccer

A third relation worked a expected, the one between a make of car and the country it was (originally) made.

Fiat Italy Peugeot France

Citroen France Mercedes Germany

Opel Germany Volvo Sweden

3 Perceptron

The following experiments use a stochastic gradient descent algorithm to implement a perceptron classifier using word embeddings as features. First I look at the data, implement and evaluate a baseline system. Then I try to tune the parameters of the algorithm. Next I investigate if the classifier generalises well to classify names that are similar to ones in the training data, but are new. Lastly I evaluate misclassifications by inspecting the confusion matrix.

3.1 Data

The following experiments use a list of 35595 named entities labeled with a category. Table 1 lists the distribution of the categories. They also use a binary file containing a Python dictionary containing 50-dimension word vectors representing word embeddings taken from https://nlp.stanford.edu/projects/glove/.

Table 1: category distribution for the named entity list

label	count	percentage
CARDINAL	5291	14.9%
GPE	11392	32.0%
PERSON	5955	16.7%
DATE	4213	11.9%
ORG	8131	22.8%
LOC	613	1.7%
location	12005	33.7%
non-location	23590	66.3%
tot	35595	100.0%

Table 2: results for baseline systems for classification

	provided		dist		most	
#	acc	f-1	acc	f-1	acc	f-1
2-way	0.81	0.80	0.56	0.55	0.65	0.52
6-way	0.79	0.79	0.22	0.22	0.32	0.16

3.2 Baseline

The provided script appears to perform quite well, but results can be caused by a skewed distribution in the dataset. Table 1 shows that this may wel be the case. In order to evaluate the performance I wrote two baseline systems. One system uses the binary distribution of a location and a non-location labels and labels every third items as a location. Another systems labels every items as a non-location. Similarly for the six-way classification one base-line system labels roughly according to the distribution and one labels all entities as the most common label *GPE*. The performance of the provided system and the baselines are listed in table 2. The results show that the perceptron performs well compared to the baseline systems.

3.3 Parameter tuning

I ran the experiments with different settings for the learning rate, penalty and number of iterations. The accuracy and f-1 scores for the configurations are listed in tabel 3. The parameters influence how the perceptron weights are adjusted. The learning rate can be constant throughout the training or it can decrease in proportion to the length of training or according to local optima. The penalty parameter provides a means of defining what if any regularisation is performed. Regularisation penalises complexity and improves generalisation. The number of iterations determines how many times the learning algorithm goes through the training set.

Table 3: results for different perceptron parameter settings

learning	penalty	iter	type	f1	acc
constant	none	5	2	0.80	0.81
constant	none	5	6	0.79	0.79
constant	none	10	2	**0.91**	**0.91**
constant	none	10	6	0.77	0.76
constant	none	25	2	0.88	0.88
constant	none	25	6	0.76	0.76
constant	l2	5	2	0.88	0.88
constant	l2	5	6	0.76	0.76
constant	l2	10	2	0.87	0.87
constant	l1	5	2	0.88	0.88
constant	l1	5	6	0.78	0.78
constant	l1	10	2	0.90	0.90
constant	l1	25	2	0.81	0.83
constant	elasticnet	5	2	0.78	0.78
constant	elasticnet	5	6	0.67	0.66
optimal	none	5	2	0.90	0.90
optimal	none	5	6	0.79	0.79
optimal	none	10	2	0.72	0.76
optimal	l1	5	2	0.82	0.83
optimal	l1	5	6	0.62	0.62
optimal	l1	10	2	**0.91**	**0.91**
optimal	l1	25	2	0.88	0.88
invscaling	none	5	2	**0.91**	**0.91**
invscaling	none	5	6	0.78	0.77
invscaling	none	10	2	0.74	0.76
invscaling	l1	5	6	**0.80**	**0.80**
invsclaing	l1	5	2	0.86	0.86
invscaling	l1	10	2	0.86	0.86
invsclaing	l1	25	2	0.89	0.89
invsclaing	l1	50	2	0.89	0.89
invscaling	l1	50	6	0.76	0.76
invscaling	l1	25	6	0.76	0.75
invscaling	l1	10	6	0.72	0.72

These three factors learning rate, regularisation and number of iterations are related and it is hard to isolate the influence of any one of them. The results in table 3 do show that the binary classification is easier in general. For six-way categorisation there a several combinations that perform within 1% difference from the the highest score of 80% for both accuracy and f-1. For the binary task there are three different combinations performing the same at 91%. These combinations have no clear advantageous setting in common.

3.4 Evaluation

Table 4: test cases for generalisation

test	label	examples
2030s	DATA	1970s, 60s, '80s, mid-30s
Lenovo	ORG	IBM, Microsoft, Apple, Motorola
Roosevelt	PERSON	Truman, Nixon, Eisenhower

I checked the classification of unseen names that are similar to ones in the data. Table 4 show

Table 5: counts of topic and sentiment labels

	card	date	gpe	loc	org	person
card	1178	1	10	2	115	5
date	46	664	1	0	306	0
gpe	3	1	2283	50	528	50
loc	1	0	6	147	17	6
org	42	24	232	49	1476	249
person	18	4	107	21	236	1021

the tests. They were all classified correctly and this indicates that the classifier might generalised well. However the accuracy and f-1 scores show large portion of misclassifications. Table 5 shows the confusion matrix for the six-way classification. Both cardinals and dates are often mistakenly classified as organisations, 9 and 30 percent respectively. The same goes for geopolitical entities and persons, 18 and 17 percent. Organisations on the other hand are mistakenly classified as persons or geopolitical entities, 12 and 11 percent, but not as cardinals or dates, 2 and 1 percent. Persons are also mistakenly classified as geopolitical entities, 7 percent of the persons. The perceptron is a linear classifier and this can explain some of the misclassifications as organisation names occur in many contexts.