ECS763P: Vector Space Semantics for Similarity between Eastenders Characters Report

Eleanor Prashamshini (220772291)
Queen Mary University of London
e.prashamshini@se22.qmul.ac.uk

Introduction

This report outlines the steps taken to improve the representation and distinction of characters from the Eastenders in the vector space. The baseline process implements simple pre-processing, feature extraction and vectorization methods. The aim is to achieve a mean rank of 1, which is interpreted as all the characters being correctly distinguished and identified. For training, the first 400 lines are used from the training.csv, which is split into 10% validation and 90% training dataset.

NLP Pipeline

1. Improve Pre-processing

The initial model has mean rank (rank) of 4.5, mean cosine similarity (sim) of 0.91 and accuracy (a) of 25%. We can create a generalised bag-of-words (bow) by removing the punctuations and lowercasing the words. Table 1 records the results of combining bow model with other pre-processing methods such as – stop words removal (swr), '_EOL_' character removal (re) and lemmatization (lm). Here, blue highlight's the baseline and green the selected improvement.

	rank	sim	a %
baseline	4.5	0.91	25.00
bow	4.3125	0.93	37.00
bow,re	3.125	0.88	31.25
bow,swr	2.5	0.94	56.25
bow,re,swr	2.3125	0.54	62.50
bow,swr,lm	2.4375	0.94	62.50
bow,re,swr,lm	2.125	0.56	68.75

Table 1: Pre-processing Results

2. Improve Linguistic Feature Extraction

Additional features such as part-of-speech tags (pos), bigrams (bi), trigrams (tri) and sentiment (s) can be incorporated to further improve the performance.

POS tags on their own don't carry much information, however when combined with lemmatization, it significantly improves the rank. Feature selection by percentiles (SelectPercentile()) helps reduce the number of features, using mutual information (MI) as a scoring metric. MI calculates the dependency between the features and the training labels. It is observed that the rank remains similar with only 33 percentile of the features. Thus, for a better generalization we will be selecting this model and proceeding.

	rank	cos	a %
unigram(u)	2.125	0.56	68.75
pos+u	1.625	0.63	68.75
pos+u,bi	1.75	0.52	56.25
pos+u,tri	1.8125	0.51	62.50
pos+u,bi,tri	1.875	0.47	62.50
pos+u+s	1.6875	0.63	68.75
fs(pos+u+s)	1.6875	0.64	75.00

Table 2: Extra Features and Feature Selection

No weighting methods have been added as the document contains all the lines and applying methods such as normalization did not make much difference to the count.

3. Analyse the Similarity Results

The image below captures the cosine similarity between held-out data (vertical axis) and the training representations (horizontal axis).

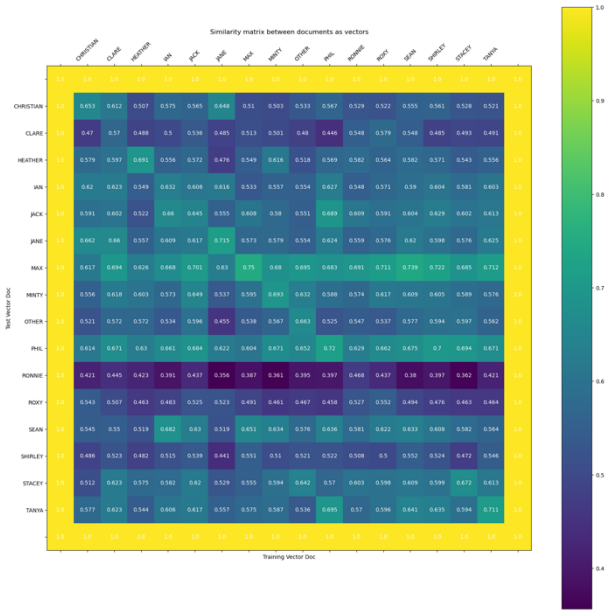

Image 1: Cosine Similarity Heatmap

Currently, the model has an accuracy of 75% and fails to recognize the similarity of 4 target characters from the held-out dataset. These characters are – CLARE, JACK, SEAN and

SHIRLEY.

- CLARE – observed to be close ROXY.
- JACK – observed to be close to IAN and PHIL.
- SEAN – observed to be close to IAN, MAX, MINTY and PHIL
- SHIRLEY – observed to be close to JACK, MAX, SEAN and TANYA

CLARE is observe to mostly have short sentences. JACK is observed to have similar language to IAN and PHIL. SEAN and SHIRLEY have been poorly represented as they match with a lot of characters.

Most of the characters are observed to be quite similar due to the short nature of sentences. Since it is a dialogue between characters, many of the sentences are simple responses. This doesn't help us identify distinctive features about the characters. The longer sentences capture more of the vocabulary used by characters and can be utilized for better representations.

All other characters have been represented to a good degree of matching in the training space. Two characters that stand-out from the matrix are – MAX and RONNIE.

- MAX has a mostly green line indicating that the held-out character vector was substantially distinguishable from the others.
- RONNIE has a predominantly purple line indicating that it is not very distict from the other character vectors. However, the matching is still a success indicating that the character has some features that are quite different from the others.

4. Add Dialogue Context & Scene Features

Upon analyzing the training dataset, we can see the characters often call each other's name or a nickname, additionally some characters are often seen in the same places. Using these features, we can improve the context. Specifically, previous line, next line and scene information have processed. We have extracted all forms of nouns have been as features and added them as extra features.

There is no improvement seen in the similarity performance and remains at rank 1.6875 and accuracy 75%. This is so because of the feature selection algorithm, and we cannot discard these features as we do not know the exact features that are being selected.

5. Improve the Vectorisation Method

The dictionary vectorizer(dv) only produces a sparse matrix of the features vectors. To improve this, we can combine it with term frequency – inverse document frequency (tfidf). TF-IDF helps evaluate how important a feature is to a character document in the collection of documents. The default TfidfTransformer() available in sklearn has the configuration – norm = 'l2', use_idf = True, smooth_idf = True, sublinear_tf = False. Table 3 records the performance of adjusting different parameters of the transformer.

	rank	cos	a %
dV	1.6875	0.64	75.00
tfidf	1.3125	0.48	81.25
tfidf(l1)	1.3125	0.48	81.25
tfidf(smooth_idf=F)	1.3125	0.46	81.25
tfidf(use_idf=F)	1.6875	0.63	68.75
tfidf(sublinear_tf=T)	1	0.39	100.00

Table 3: TF-IDF Performance

The most significant boost is seen by setting the sublinear_tf to True. This method applies sub-linear scaling to term-frequency (1 + log(tf)). Note that we are still using this with the SelectPercentile() method.

6. Run on Final Test Data

We re-initialize the three transformers used in our best vector semantic representation model – DictVectorizer(), SelectPercentile() and TfidfTransformer(). This is done to ensure that there is no fitting from the training data while testing. Table 4 compares the performance of the model to the training performance.

	rank	*cos*	*a %*
Initial Training	*4.5*	*0.91*	*25.00*
Training	*1*	*0.39*	*100.00*
Testing	*1.1875*	*0.41*	*87.50*

Table 4: Training and testing results

The semantic representation model succeeded in identifying 14 characters from the test set and failed to recognize 2 character – JACK and CLARE. This implies that the model is representing the characters as expected for the most. There is considerable improvement from the baseline model for both rank and accuracy. This was achieved by a trade-off with the mean cosine similarity.