# Assignment 2 Data Modelling and Presentation

Name: Riya Minesh Amin

Student ID: S3807007

## OVERVIEW

Machine learning techniques for training the model in a ranking problem are referred to as learning to rank. Many applications in Information Retrieval, Natural Language Processing, and Data Mining benefit from learning to rank. This brief report illustrates on the pairwise linear regression and Linear SVR machine learning models used to rank a large query-document pair. To train the model, feature extraction in documents and queries is conducted, and a word2vec model is trained with a training dataset.

## MODEL DESCRIPTION

Linear regression is a machine learning approach for supervised learning. It does a regression task. Regression models a goal prediction value based on independent variables. Along with established parameters, a linear regression model was utilised to predict the NDCG scores for the document-query pair dataset. The 'positive' parameter was set to "False", and the 'fit intercept' parameter was set either "True" or "False". The default values for all other parameters were used.

From a GridsearchCV, we can observe the optimal parameters and results from fitting the training set in the figure below. The optimum parameter is 'fit intercept': False, 'positive': False, which results in an average NDCG score of 0.847168 and a standard test score of 0.005694.

```
..  dict_keys(['copy_X', 'fit_intercept', 'n_jobs', 'normalize', 'positive'])
    Fitting data
    {'fit_intercept': False, 'positive': False}
    0.8471681548843385
       mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
    0       0.408219      0.012110         0.013895        0.005957
    1       0.280615      0.183188         0.015246        0.008079

      param_fit_intercept param_positive  \
    0                True          False
    1               False          False

                                        params  split0_test_score  \
    0   {'fit_intercept': True, 'positive': False}           0.847022
    1  {'fit_intercept': False, 'positive': False}           0.845674

       split1_test_score  split2_test_score  split3_test_score  split4_test_score
    0           0.848644           0.855589           0.841417           0.842919
    1           0.850505           0.856448           0.842245           0.840969

       mean_test_score  std_test_score  rank_test_score
    0         0.847118        0.004985                2
    1         0.847168        0.005694                1
```

*Figure1: LinearRegression() best parameters results.*

The model was then put through a validation holdout test using the best parameters, yielding an NDCG score of 0.90910084, indicating that the parameters had been properly tuned.

## FEATURE EXTRACTION

The document and query pair datasets were used to extract features before training the model. To map from a bigger issue space to a smaller feature space, documents were divided into smaller fragments. Dictionary of English contractions were utilised to extend and replace these contractions to reduce the dimensionality of each document fragment. All the fragments of the documents.tsv and queries.tsv dataset was then pre-processed and cleaned in order to perform vectorisation.

For vectorization, the Word2vec model was used. Word2vec computes a word's distributed vector representation. The major benefit of distributed representations is that related words in the vector space are close together, making generalisation to novel patterns easier and model estimate more reliable. Many natural language processing applications, such as named entity identification, disambiguation, parsing, tagging, and machine translation, have been shown to benefit from distributed vector representation.

Gensim was used to create the Word2Vec model. The training dataset was used to train the Word2Vec model. Queries and Doc ids were retrieved from the documents chunk and queries.tsv that were solely in the training set for the purpose of training and incorporated in the model to transform the queries into vectors.

Following the transformation of the queries into vectors, a cosine function was created to identify the feature space for similarity between the terms using their vector representation on training and testing datasets. The inner product of the two vectors may simply be computed:

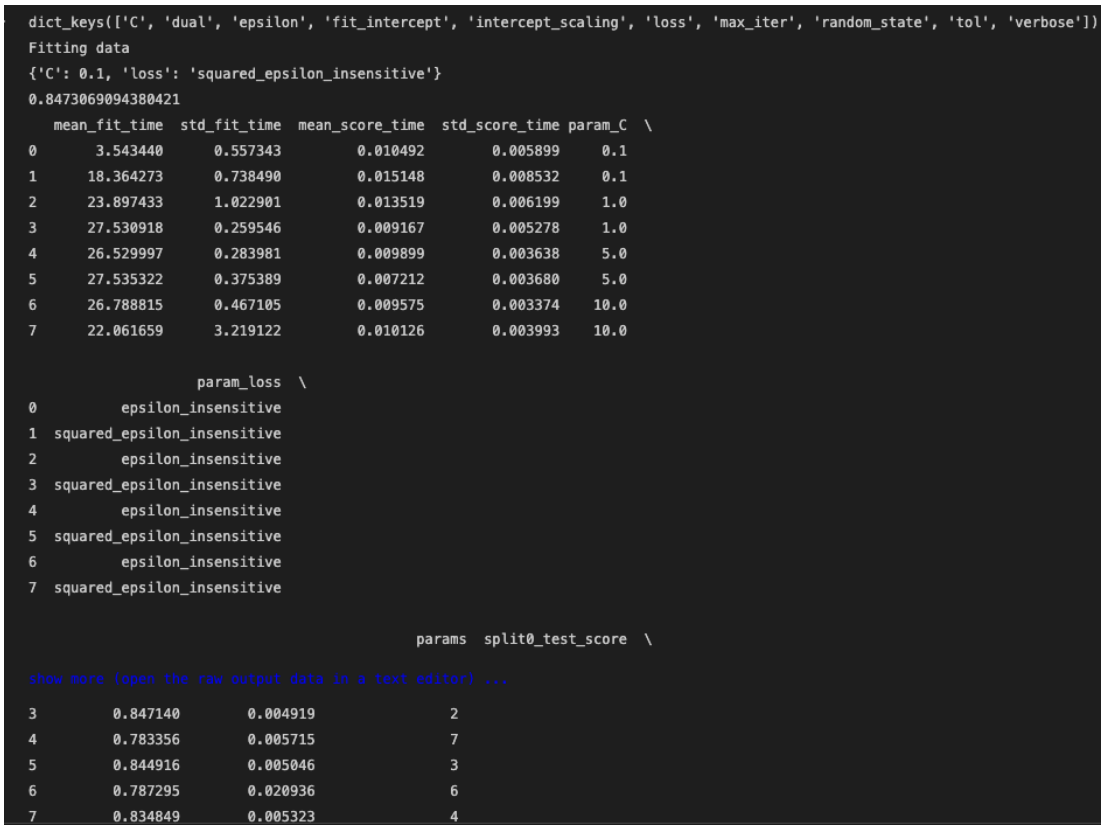
$$\cos\alpha = \frac{\vec{a}\cdot\vec{b}}{|\vec{a}|\,|\vec{b}|},$$

## EXPERIMENTS

On the query-document dataset, two models were implemented: linear regression and linear support vector regression. The cosine similarity between the vectors was calculated for both models using feature extraction. The data was divided into 80:20 training and validation groups and categorised according to queries. Normalization was applied to the Train, Validation, and Test datasets, and K-fold Cross Validation was generated by assigning group numbers to queries in the training datasets. GroupKFold was utilised and separated into 5 groups to ensure that the same group was not presented in both the training and validation sets. The NDCG scoring technique was used. For both models, GridsearchCV was used to determine the best parameters. On the training dataset, the GridsearchCV results were used to fit both models. Validation holdout was put into the model to evaluate the model, and the mean NDCG score was attained.

We saw in figure 1 the optimal parameters of Linear regression and the mean NDCG score.

For Linear Support Vector Regression model, regularisation parameter was set to ‘C’: [0.1, 1.0, 5., 10.] and the loss function was set to l1 and L2: ‘loss’:['epsilon_insensitive', 'squared_epsilon_insensitive'], all other parameters were set to default. From the GridsearchCV, we observe the optimal parameter for the LinearSVR model in figure 2. We see that the optimal regularisation parameter is ‘0.1’ and the loss function is ‘'squared_epsilon_insensitive' Though the mean NDCG score is 0.8473 when the model is tested on the validation holdout the NDCG score is 0.8304.



```
dict_keys(['C', 'dual', 'epsilon', 'fit_intercept', 'intercept_scaling', 'loss', 'max_iter', 'random_state', 'tol', 'verbose'])
Fitting data
{'C': 0.1, 'loss': 'squared_epsilon_insensitive'}
0.8473069094380421
   mean_fit_time  std_fit_time  mean_score_time  std_score_time param_C  \
0       3.543440      0.557343         0.010492        0.005899     0.1
1      18.364273      0.738490         0.015148        0.008532     0.1
2      23.897433      1.022901         0.013519        0.006199     1.0
3      27.530918      0.259546         0.009167        0.005278     1.0
4      26.529997      0.283981         0.009899        0.003638     5.0
5      27.535322      0.375389         0.007212        0.003680     5.0
6      26.788815      0.467105         0.009575        0.003374    10.0
7      22.061659      3.219122         0.010126        0.003993    10.0

                   param_loss  \
0         epsilon_insensitive
1  squared_epsilon_insensitive
2         epsilon_insensitive
3  squared_epsilon_insensitive
4         epsilon_insensitive
5  squared_epsilon_insensitive
6         epsilon_insensitive
7  squared_epsilon_insensitive

                                        params  split0_test_score  \
show more (open the raw output data in a text editor) ...
3        0.847140        0.004919              2
4        0.783356        0.005715              7
5        0.844916        0.005046              3
6        0.787295        0.020936              6
7        0.834849        0.005323              4
```

*Figure2: LinearSVR() best parameters results.*

From the 2-model comparison, we see that the Linear Regression model preforms better compared to Linear Support Vector Regression model.

Therefore, the test dataset is then fit into the Linear Regression model.

## References:


- Gensim Tutorial - A Complete Beginners Guide - Machine Learning Plus. (2021). Retrieved 10 October 2021, from https://www.machinelearningplus.com/nlp/gensim-tutorial/
- *Web-Mining/information-retrieval.ipynb at 3c961ea02fef8cec2416f47fcaee0d6401ba0bf5 · BelemoualemChaimae/Web-Mining*. GitHub. (2021).
- *GitHub - nadyadtm/Document-Retrieval: Tugas NLP*. GitHub. (2021). https://github.com/nadyadtm/Document-Retrieval.
- Wang, B., & Klabjan, D. (2017). *An Attention-Based Deep Net for Learning to Rank*. Cs.purdue.edu. https://www.cs.purdue.edu/homes/liu1740/report.pdf.
- *NLP - Expand contractions in Text Processing - GeeksforGeeks*. GeeksforGeeks. (2021). https://www.geeksforgeeks.org/nlp-expand-contractions-in-text-processing/.