A project report comparing different models for paper classification, and an improvement upon the random forest paper recommendation model that is used by [www.benty-fields.com](http://www.benty-fields.com)

Improving the benty-fields.com Paper Recommendation Engine

Alexander Thomson-Strong

# Introduction

Benty-fields uses a machine learning algorithm to make and rank paper recommendations to its users [Beutler, 2021]. The aim of this project is to improve upon the current model, in terms of accuracy without significantly increasing the processing time.

The model that benty-fields currently uses [Beutler, 2020] is a random forest classifier with 100 estimators, along with a Tf-Idf vectorizer applied to the abstracts, titles and authors of the papers belonging to each user. The vectorizer takes up to 3000 individual words (unigrams) and builds a vocabulary, removes the stop words, and creates a vector, using the term frequency inverse-document frequency, for each paper. This results in an accuracy of around 90%.

The alternative model that I propose [Thomson-Strong, 2021] uses the same Tf-Idf vectorizer, except with different hyper parameters. It builds a vocabulary of size 30,000, including strings of between 1 and 4 words. Based on how correlated each feature is, the most informative 10% of these are determined using a chi-square test, and the rest are removed. This change results in an improvement of around 0.5% for the random forest classifier and was carried forward.

The model also replaces the random forest classifier with an extra trees classifier with 100 estimators, which produces less correlated decision trees. This results in more accurate predictions with the caveat that the processing time for users with a large number of papers is slightly slower. For users with a small number of papers, the processing time is slightly faster.

In the final section before the conclusion, the other approaches which were implemented and subsequently rejected as less successful than those above are discussed.

# The Classifiers

The first two models that are compared here are the random forest model currently used by benty-fields, and an extra (extremely randomized) trees classifier, which also uses the larger vocabulary and feature selection technique described above.

The Difference Between the Two Classifiers

Both Classifiers construct decision trees, using a subset of the features in the training set of data. They construct multiple trees and take the average of the predictions from each tree to produce their overall prediction. However, where they diverge is the method by which they construct the trees. The random forest classifier chooses the split thresholds for each feature iteratively, and the splitting rule is generated by maximizing the information gain at each split, usually by using the Gini index [Beutler, 2017]. The extra trees classifier uses a slightly different method. The key difference is that decision thresholds are chosen randomly for each candidate feature and the best threshold is determined (by maximizing information gain) and is applied as the splitting rule.

The extra randomness introduced by doing this reduces the variance when compared to the random forest classifier, and the extra trees classifier is more robust to overfitting as a result. This also means that you can fit a greater number of estimators (decision trees) without risking overfitting the model. Generally, with an extra trees classifier, you can keep adding estimators to increase the accuracy, albeit with diminishing returns and an increase in processing time [Ceballos, 2019].

The code corresponding to the key features of the original model is:

And the main changes that were made are: 

# Results

Using the whole data set to train and test the model is prohibitively time-consuming for the purposes of working on improving the model, since there are 1403 users with up to 14,000 papers each. The users were indexed in an order roughly corresponding to the number of papers, in descending order, therefore the following samples seemed appropriate: every 20th user, denoted by (0, 1403, 20); the first 200 users, denoted by (0, 200); and the last 200 denoted by (1203, 1403).

The metrics used to evaluate are as follows: the mean AUC for the sample, as a measure of the overall accuracy; the graph of the AUC vs the number of papers associated with each user; the total processing time (on my 2017 MacBook Air: 1.8GHz Dual-Core, 8Gb RAM); and the graph of processing time against the number of papers for each user.

To get an estimate for the standard deviation of the mean AUC score the extra trees model was run 5 times on the (0, 1403, 20) sample and the standard deviation was calculated. The results are below:

|  |  |
| --- | --- |
|  | Mean AUC Score |
| 1 | 0.9202442640831292 |
| 2 | 0.92238239611259 |
| 3 | 0.9217593235821647 |
| 4 | 0.9249565842652316 |
| 5 | 0.922693256468949 |
| mean | 0.9224071649 |

The standard deviation is therefore estimated to be:

$$σ^{2}=\frac{Σ\left(X- \overbar{X}\right)^{2}}{n-1}$$

$$σ=2.93 × 10^{-3}$$

The error is therefore approximately ±0.003

(0, 1403, 20)

# Chart  Description automatically generated

Figure 1: The AUC vs number of papers for both models on the (0,1403,20) sample. The random forest model is in red, the extra trees model is in blue. The error bars are the standard deviation of the results from 10 cross-validation steps. The x-axis uses a log scale.



Figure 2: The processing time vs data points for both models on the (0,1403,20) sample. The random forest model is orange, the extra trees model is blue. The x-axis is on a log scale.

(0, 200)



Figure 3: The AUC vs data points for both models on the (0,200) sample.



Figure 4: The processing time vs data points for both models on the (0,200) sample.

(1203, 1403)



Figure 5: The AUC vs the data points for both models on the (1203,1403) sample.



Figure 6: The processing time vs data points for both models on the (1203,1403) sample.

Evidently, the extra trees classifier performs at a higher accuracy than the random forest classifier. This difference is statistically significant when the number of papers is large, and when it is small. Over the whole range of users, this amounts to a 2% increase.

The processing time for the extra trees classifier is slower overall. This is mostly due to the larger vocabularies which are used with this model. The limit for the size of the vocabulary only affects users with papers that contain more words than the limit. As the users with the fewest papers are likely not affected by this limit, increasing it will not affect the size of the corresponding vocabulary – here the limiting factor is the number of papers. Therefore, for these users, this won’t affect the processing time. Furthermore, if you are using the same size vocabulary, the extra trees classifier can construct decision trees more quickly. This is because the splitting thresholds are chosen randomly, as opposed to iteratively. This explains why the increase in processing time is largest when the number of papers is largest.

A slightly greater increase in accuracy can be observed by increasing the number of estimators. An extra trees model is more robust to overfitting, since the decision trees which are produced are less strongly correlated. Therefore, it is possible to keep increasing the number of estimators to achieve greater accuracy without risking a model which is overfitted. The number of estimators has a significant impact on the processing time, and the benefit to adding estimators decreases as you add them, thus it is necessary to find a compromise. The value of 100 estimators was eventually chosen as this produced a significant increase in the accuracy, without making the model too slow.

# Other Approaches

Other Classifiers

The K-nearest neighbours classifier (KNN) uses the feature vectors to place each paper in a high dimensional vector space, and then calculates the Euclidean distance between the papers in the training and testing sets. For each paper in the testing set, a majority vote of the nearest K training papers is used to determine the class that it is predicted to have. This is a very simple algorithm and is relatively fast.

It was found to perform significantly faster than the random forest model, especially when dealing with large numbers of papers, although this came at the cost of a reduction in accuracy.



Figure 7: The AUC against data points for the KNN (blue) and random forest (red) models. The KNN model uses the 10 nearest neighbours.



Figure 8: Processing time against data points for the KNN (blue) and random forest (orange) models.

From the processing time graph, it is evident that the KNN model displays the most significant improvement in speed when the number of papers is very large. If the average number of papers was significantly greater, a KNN model might be more appropriate.

A more accurate variation of the KNN model was then implemented using grid search cross validation. 

This works by iterating over a grid of values for the hyper parameters of an estimator and finds the optimized combination of values. This was applied to the KNN classifier, such that for each cross-validation step, for each user, the classifier had the best hyper parameters from those provided in the grid. This significantly increases the processing time that is required, but this is also accompanied by a significant increase in accuracy, although this is not necessarily enough to beat the random forest or extra trees models.

Figure 9: AUC against data points for the KNN (blue) and random forest (red) models. The grid search was only applied to papers with fewer than 1000 papers, to reduce the processing time.

It should be noted that the effect that using this method has on the accuracy and speed of a model is dependent on the parameter grid that is used. A large range of values will give the best outcome in terms of accuracy but will also be significantly slower than a smaller grid.

Other classifiers, including the support vector classifier (SVC), Multi-layer perceptron classifier (MLP), and decision tree classifiers were also implemented, but these resulted in no significant improvement upon the random forest model.

The last classifier to be implemented was the voting classifier. This takes a list of classifiers as an input and fits them on the data individually. It then aggregates the predictions from each individual classifier (either by weighting the results based on the accuracy of each individual classifier, or by a simple majority vote).

In cases where you have multiple classifiers which perform with similar accuracy, and they are trained in significantly different ways, this method can outperform the best individual classifier.

In this case, using the random forest, extra trees, and KNN classifiers, this approach roughly reproduced the accuracy of the extra trees model, and the processing time was roughly the sum of each of the individual classifiers.

If you had multiple classifiers which performed at a similar level to the extra trees model, the voting classifier might be a more worthwhile approach.

Feature Selection

Recursive feature elimination with cross validation takes a subset of the features and applies a statistical test to determine which group of features is the least significant and removes them recursively, until an optimal subset of features is found. This can be useful for reducing overfitting and for reducing processing time by removing irrelevant, or strongly correlated features. In practice, this was found to be impractically slow.

Instead, changing the hyper parameters of the vectorizer and using select percentile was implemented for feature selection.

As is mentioned above, the hyper parameters of the vectorizer were tuned to achieve a more informative set of features. Obviously, increasing the maximum size of the vocabulary is limited by the available computing resources. However, the size of the vocabulary was increased by a factor of 10, and strings of up to four words were also included without making the model prohibitively slow. It was found that increasing the limit mostly affected the users with more papers, as they were the only ones that were significantly constrained by the previous smaller limit.

Constructing decision trees with a 10 times larger vocabulary could be problematic in terms of dimensionality. Therefore, the feature selection technique select percentile was implemented. It was found that accuracy was generally greatest when around 10-20% of the features were selected. A chi-square test was used (by supplying ‘chi2’ as a parameter to select percentile) to choose the least correlated set of features to minimize redundancy. This resulted in a set of features which was roughly the same size as was used with the original model, that contains strings of up to four words, instead of only unigrams, and which contains fewer words which offer very little information.

With the random forest model, this change resulted in an increase in accuracy of around 0.5%, and it was what was used with the extra trees classifier in the final model.

Sentiment analysis

Using Doc2Vec from Gensim, a sentiment analysis approach was attempted. This was based on the code from [Li, 2018] which produces a distributed bag of words (DBoW) and then applies a linear regression classifier.

Doc2Vec produces feature vectors for each paper by training a neural network on the task of predicting missing words within the text, given the words which appear around the missing words. The relative positions of these vectors in n-dimensional space should give an indication of how similar they are in sentiment.

A logistic regression classifier was then applied to these vectors, in order to make predictions.



This proved to be somewhat ineffective for this task. It was found that the accuracy was between 60-70%, although the processing time was significantly faster.



Figure 10: AUC against number of papers for the Doc2Vec model, using a linear regression classifier. The sample used was the first 200 users (0,200). The error in these results was not estimated, although it was observed to be on the order of $10^{-2} for the mean accuracy$. The metric for accuracy used here was the accuracy score from sci-kit learn, not the AUC, as this is what was used in [Li, 2018].

There were a lot of hyper parameters that could have affected this result, which were mostly left to the default values. Tuning these may yield more accurate predictions, but this wasn’t investigated rigorously.

With this approach, any other classifier can also be applied instead of the logistic regression classifier. This could also result in more accurate results, although the extra trees and random forest classifiers didn’t produce significantly better results.

Downloading the Full Paper Text

Using the arXiv API, it is possible to download entire LaTeX files for the papers. Therefore, you can extract the entire text of the paper, and you will have a lot more available features than if you are only using the abstract, title, and authors.

Regular expressions can be used to clean the text of LaTeX commands, equations, and graphs (which all contain very little information) and then it can be tokenized and vectorized in the same way as before. It was found that a series of ‘if line starts with:’ statements was sufficient to crudely clean the file.

Unfortunately, a significant number of the papers were uploaded as scanned pdf’s, not LaTeX files. This means that the text cannot be easily read from file. One approach that was attempted was to convert the file to a jpeg and then trying to recognize the text from the image using an image processing package in python. This proved to be too complicated for me to implement.

For some users, as many as 1/5 of the full papers were unavailable. To compensate for this, the papers were used along with the abstracts that were originally used.

This set of features was tested for a group of users with a small number of papers each. A slight improvement in the accuracy of the model was observed (<1%). The improvement would likely have been greater if the cleaning of the text had been less crude and if all the papers could have been included.

This wasn’t developed further because it was too time consuming to download all the papers for each user and as mentioned above, extracting the text from the PDF’s was too complicated. If you can solve this problem, you might be able to achieve a better improvement in the accuracy of the model.

This would presumably have the greatest impact for users with a small number of papers, as they have the least amount of data for the model to work with.

# Conclusion

The extra trees classifier performs better than the random forest classifier in all the samples in terms of accuracy, which covers the first and last 200 users, and a sample of users which represents the whole data range. The AUC score is around 2% better over the whole range of users. This difference is statistically significant.

The trade-off is that the processing time for users with the most papers is slightly slower, although it is faster for the users with the fewest papers. Presumably, this has to do with the larger vocabulary used.

Using an extra trees classifier with a greater number of estimators, you can gain an even greater improvement in accuracy at the expense of the speed of the model. The challenge is optimising the balance between these two factors. The number of estimators used here was chosen because it resulted in a significant improvement from the original model in accuracy without being prohibitively slow.

A further increase in accuracy could possibly be achieved if you use the full text from each paper. The increase from trying this was small, but presumably it would be greater if you used a more sophisticated method for cleaning the LaTeX files and you were able to extract the full text from every paper. You would then have to evaluate whether this increase in accuracy is worth the inevitable increase in processing time.

It appears that the sentiment analysis approach that was attempted here is not particularly effective for this task. This method was significantly faster than all the other approaches, although it was also the least accurate. The accuracy might be improved by changing the hyper parameters involved or by using a different classifier, but this approach wasn’t explored rigorously.

# Bibliography

Arxiv.org. 2021. *arXiv API User's Manual | arXiv e-print repository*. [online] Available at: <https://arxiv.org/help/api/user-manual#\_entry\_metadata> [Accessed 21 June 2021].

Beutler, F., 2017. *Building a random forest classifier from scratch in Python*. [online] Mystatisticsblog.blogspot.com. Available at: <https://mystatisticsblog.blogspot.com/2017/12/building-random-forest-classifier-from.html> [Accessed 24 June 2021].

Beutler, F., 2020. Paper Recommendation Engine. [online] GitHub. Available at: <https://github.com/fbeutler/benty\_fields\_ML\_model/blob/main/paper\_recommendation\_engine\_public.py> [Accessed 21 June 2021].

Beutler, F., 2021. *benty-fields - About*. [online] Benty-fields.com. Available at: <https://www.benty-fields.com/projects> [Accessed 21 June 2021].

Borcan, M., 2020. *Explained: Word2Vec Word Embeddings - Gensim Implementation Tutorial And Visualization*. [online] Programmer Backpack. Available at: <https://programmerbackpack.com/explained-word2vec-word-embeddings-gensim-implementation-tutorial-and-vizualization/> [Accessed 18 July 2021].

Breaker, L., 2019. *Voting Classifier in Machine Learning*. [online] Medium. Available at: <https://medium.com/analytics-vidhya/voting-classifier-in-machine-learning-9534504eba39> [Accessed 8 August 2021].

Ceballos, F., 2019. *An Intuitive Explanation of Random Forest and Extra Trees Classifiers*. [online] towardsdatascience.com. Available at: <https://towardsdatascience.com/an-intuitive-explanation-of-random-forest-and-extra-trees-classifiers-8507ac21d54b> [Accessed 6 August 2021].

Di Pietro, M., 2020. *Text Classification with NLP: Tf-Idf vs Word2Vec vs BERT*. [online] Medium. Available at: <https://towardsdatascience.com/text-classification-with-nlp-tf-idf-vs-word2vec-vs-bert-41ff868d1794> [Accessed 18 July 2021].

Horey, R., 2018. BERT Explained: State of the art language model for NLP. [online] Medium. Available at: <https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270> [Accessed 18 July 2021].

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L. and Brown, D., 2019. Text Classification Algorithms: A Survey. *Information*, [online] 10(4), p.150. Available at: <https://www.mdpi.com/2078-2489/10/4/150/htm> [Accessed 8 July 2021].

LaTeX Wiki. n.d. *List of LaTeX commands*. [online] Available at: <https://LaTeX.wikia.org/wiki/List\_of\_LaTeX\_commands> [Accessed 15 July 2021].

Li, S., 2018. Multi-Class Text Classification with Doc2Vec & Logistic Regression. [online] Medium. Available at: <https://towardsdatascience.com/multi-class-text-classification-with-doc2vec-logistic-regression-9da9947b43f4> [Accessed 24 July 2021].

Monkeylearn.com. n.d. *Sentiment Analysis: A Definitive Guide*. [online] Available at: <https://monkeylearn.com/sentiment-analysis/> [Accessed 20 July 2021].

Radimrehurek.com. 2021. *Gensim: topic modelling for humans*. [online] Available at: <https://radimrehurek.com/gensim/models/doc2vec.html> [Accessed 22 July 2021].

Radimrehurek.com. n.d. *Gensim: topic modelling for humans*. [online] Available at: <https://radimrehurek.com/gensim/auto\_examples/tutorials/run\_word2vec.html> [Accessed 18 July 2021].

Řehůřek, R. and Emerick, T., 2014. *Doc2vec tutorial | RARE Technologies*. [online] Rare-technologies.com. Available at: <https://rare-technologies.com/doc2vec-tutorial/> [Accessed 21 July 2021].

Scikit-learn.org. n.d. *1.11. Ensemble methods — scikit-learn 0.24.2 documentation*. [online] Available at: <https://scikit-learn.org/stable/modules/ensemble.html#forest> [Accessed 20 July 2021].

Scikit-learn.org. n.d. *sklearn.ensemble.ExtraTreesClassifier — scikit-learn 0.24.2 documentation*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html> [Accessed 20 July 2021].

Scikit-learn.org. n.d. sklearn.feature\_selection.SelectPercentile — scikit-learn 0.24.2 documentation. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.feature\_selection.SelectPercentile.html> [Accessed 3 July 2021].

Thomson-Strong, A., 2021. Extra Trees Classifier. [online] GitHub. Available at: <https://github.com/alexrt-s/Extra\_Trees\_Classifier/blob/main/Extra\_Trees.py> [Accessed 5 September 2021].