# COMP30027 Assignment 2 Report

**Anonymous**

## 1 Introduction

The purpose of this report is to evaluate performance of different machine learning (ML) algorithms in the task of classifying cooking methods into three levels of cooking duration (quick, medium and slow). We experimented Recipes and Interactions dataset[1] crawled from Food.com[2] (formerly GeniusKitchen) by (Majumder et al., 2019). Our ML methods based on five main features; while name, steps, ingredients have the form of string, number of step and number of ingredients are integers. Overall, training set consisting of 40000 recipes is highly imbalanced out of which there are 17705 quick recipes, 20246 medium recipes and 2049 slow ones. Each recipe has at least 3 steps, and between 4 and 20 ingredients. Logistic Regression trained on BOW achieves best result with 80% accuracy, 76% Macro-f1 and 80% Micro-f1. This is also the model we choose to submit to Kaggle.

## 2 Related Work

Text classification plays an important role in Natural Language Processing (NLP). This problem has been researched for decades. At the heart of this problem are classification algorithm as well as feature extraction methods. A classical method is Bag-of-Words (BOW), a simple extension of one hot encoding, often produce surprisingly performance despite its simplicity (Le and Mikolov, 2014). Also, other word representations that take word frequency into account such as Term Frequency (TF) and Term Frequency-Inverse Document Frequency (TF-IDF) were introduced later. However, these representations did not capture syntactic and semantic meaning of the whole sentences or documents. In recent years, many novel word embedding have been researched to solve this problem. Word to vector (word2vec) (Mikolov et al., 2013) and Document to vector (doc2vec) (Le and Mikolov, 2014) were some of outstanding sentence-embedding before BERT (Devlin et al., 2018) was presented and considered one of the greatest breakthroughs in NLP area. Thousand of papers have evaluated the performance of these feature extract methods, however, (Shao et al., 2018) has a close resemblance to the task this report is solving. In particular, various feature extractors such as word2vec, doc2vec and BOW-1,2-gram are trained with Support Vector Machine (SVM) to classify computer product description into 6 modalities.

[1] `https://www.kaggle.com/shuyangli94/food-com-recipes-and-user-interactions`

[2] `https://www.food.com/`

## 3 Method

### 3.1 Experiment Setup

We evaluate different combinations of ML classifier against extracted feature from dataset including: Doc2vec, BOW as well as metadata. Performance of classifiers trained on two metadata are summarised in Section 3.2. Classifiers trained on Doc2vec and BOW are reported in Section 3.3 and Section 3.4, respectively.

In order to choose the best training size that both maximises performance and minimises overfitting, we compare training accuracy and validating accuracy evaluated by a fixed default Logistic Regression classifier against different training size. At each level of splitting, dataset is randomly split 5 times, average accuracy for each level is then summarised.

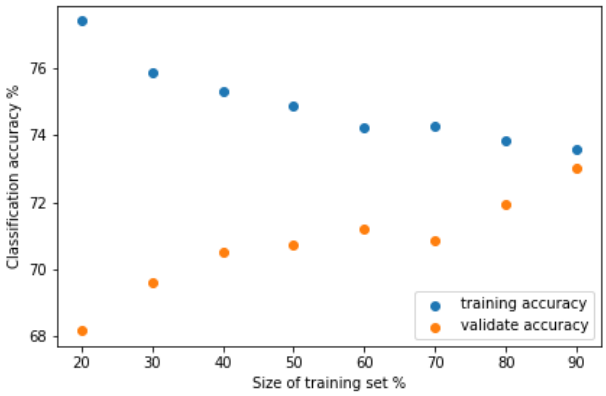


Figure 1: Effect of split size on training and validating accuracy

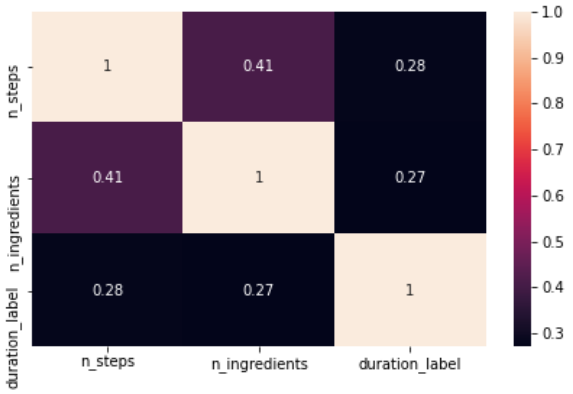


Figure 2: Correlation between metadata features

Based on Figure 1, training size of 0.9 is applied across all models in this report. Moreover, while producing this plot, we noticed that validating and training accuracy are stable throughout all random split. As a result, a simple random training set split at 0.9 and 0.1 for validating set is our training strategies. This enables us to experiment with more combinations of classifiers and feature extraction methods.

Zero-R is used in this report as baseline to evaluate our models, has accuracy of 50.6% proportion to corresponding to proportion of most common classes in training and validating set (medium cooking time) given that these two are stratifying split. And for final note, all statistics within this report are validating result if not mentioned, otherwise, it is either training result or testing result if explicitly mentioned.

### 3.2 Metadata

Metadata plays an important roles in a dataset. We trained Decision Tree (DT) and Logistic Regression (LR) on two metadata features. Parameters of these two models are tuned by Grid Search, however, parameters does not affect the performance. Accuracy is 64% while marcro-f1 and micro-f1 are 44% and 63%, respectively. The following Figure 2 summarised correlation between metadata.

The plot implies there is weak correlation among features and between features and labels. This might be not very good since features with more redundant and superfluous provide more unique information to our learning model.

### 3.3 Doc2vec

We suspect that 50 and 100 are not sufficient to capture information extracted from the dataset. By rerunning the code provided and updating vector size to 200, 300, 400, we want to determine whether there is a boost in performance if we increase the size of feature vector. We train default LR classifier with different vector size to output performance in Table 1

Table 1: Doc2vec's performance with different size

| Vector Size | Accuracy | Macro F1 | Weighted F1 |
|---|---|---|---|
| 50 | 61 | 53 | 63 |
| 100 | 67 | 56 | 67 |
| 200 | 71 | 65 | 71 |
| 300 | 71 | 65 | 71 |
| 400 | 71 | 66 | 71 |

Overall, the 400-feature vector doc2vec achieved the best performance. There is a huge gap between 200, 300 and 400 embedding compared to 50 and 100 ones. We will use doc2vec with 400 feature vector size when evaluating other classifiers.

Aiming to speed up training time, we experiment feature extraction method Principal Component Analysis (PCA) on Doc2vec features. Before feeding into a Logistic Regression with default hyperparameters, each feature

name, steps and ingredients are sequentially fitted with separate PCA model then concatenated into a single vector. We have iterated number of extracted feature between 50 and 350, however, as number of features decreased, the performance dropped. This indicates information has lost when we reducing dimension of features. Therefore, for this embedding, PCA are not applied.

#### 3.3.1 Tree-based classifier

We first experiment with Decision Tree (DT) due to its interpretability and small number of hyperparameters. Nevertheless, this algorithms is vulnerable to imbalance classification and overfitting. To minimize the latter weakness, we have tuned max_depth hyperparameter against training and validating accuracy. Based on Figure 3, it can be determined that the range between 3 and 6 are most suitable to lessen this problem. However, due to imbalance among classes, the performance has been adversely affected as expectation. Despite using Grid Search, the classifier can not seem to learn pattern of minor class (slow cooking time) as shown in Table 2.

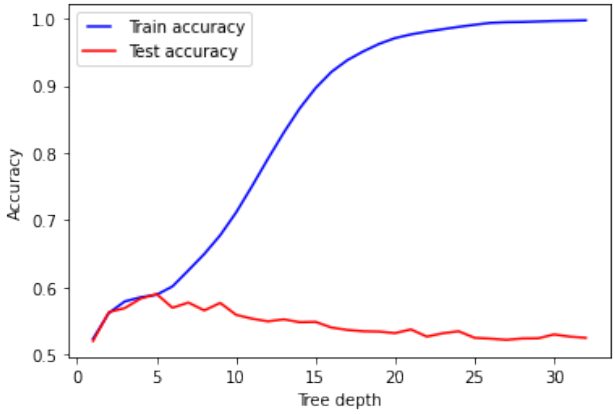


Figure 3: Performance of DT on doc2vec

Table 2: Performance of DT on doc2vec

| Class | Precision | Recall | F1-score |
|---|---|---|---|
| quick | 0.62 | 0.38 | 0.47 |
| medium | 0.57 | 0.82 | 0.67 |
| slow | 0.00 | 0.00 | 0.00 |
| Accuracy | | | 0.58 |

Random Forest (RF), an ensemble learning technique, aims to improve performance of DT but still running the risk of imbalance classes. One of solutions to this is OverBagging (oversampling minority class during bootstrap) and UnderBagging (undersampling majority during class). Imbalance-learn[3] provides implementation of RF with this ideas. Up to our expectation, the performance has improved significantly in both precision and recall of slow cooking time classes.

Table 3: Performance of RF on doc2vec

| Class | Precision | Recall | F1-score |
|---|---|---|---|
| quick | 0.74 | 0.52 | 0.61 |
| medium | 0.64 | 0.87 | 0.74 |
| slow | 0.00 | 0.00 | 0.00 |
| Accuracy (RF sklearn) | | | 0.67 |
| quick | 0.73 | 0.59 | 0.65 |
| medium | 0.70 | 0.70 | 0.70 |
| slow | 0.26 | 0.73 | 0.39 |
| — Accuracy (RF imbalance-learn) | | | 0.65 |

The last algorithms belongs to this group that we evaluated is Extreme Boost (XGboost) introduced by (Chen and Guestrin, 2016) has gone viral in ML community. Its logic, similar to other boosting algorithms such as Adaboost or Gradient Boosting, involves combining several weak learners, sequentially correcting precious predictors to achieve a better predictor at the end of the procedure. Due to expensive computation, we have set the two hyperparameters max_depth and n_estimators as 10 and 100 whose value we consider the most suitable based on previous DT-based classifiers. XGboost have outperforms other classifier in the same category as shown in Table 4

#### 3.3.2 Support Vector Machine (SVM), Logistic Regression (LR)

These two algorithms are among most frequent used classifier in text classification. While LR is computationally inexpensive but still output strong performance, SVM tackles curse of dimensionality and overfitting with stability across various problems. Consistent with those reasoning, both our LR and SVM model have achieved best result on doc2vec embedding. Performance of all classifiers trained on doc2vec are shown in Table 4.

[3] https://imbalanced-learn.org/stable/

Table 4: Summary performance of classifiers on BOW

| Classifier | Accuracy | Macro-f1 | Micro-f1 |
|---|---|---|---|
| Zero-R | 0.63 | | |
| DT | 0.58 | 0.38 | 0.55 |
| RF sklearn | 0.67 | 0.45 | 0.67 |
| RF imlearn | 0.65 | 0.58 | 0.66 |
| XGbooost | 0.70 | 0.50 | 0.68 |
| LinearSVC | 0.72 | 0.66 | 0.72 |
| LR | 0.73 | 0.66 | 0.73 |

The fact both LinearSVC and LR achieve the best performance indicates features are linearly separable and non-overlapped classes, since LR does not work well with non-linear data. The performance are consistent with theoretical properties of these classifiers. While LinearSVC, LR are robust to overfitting, tree-based classifier except for XGboost are prone to overfitting despite hyperparameter tuning.

### 3.4 Bag-Of-Words (BOW)

Due to large dimension of vector size, dimension reduction is a suitable solution to reduce training time. While experimentation on PCA method, we aim to select smallest number of feature while still capturing all variability. To achieve this,

a range with suitable training time are determined for a given features, for example, for feature "steps" with vector size of around 17000, the range between 500 and 6500 is chosen to evaluate accuracy of model with different vector size. Then a closer range around chosen number from previous step are examined to output the final number of components for PCA model, for example 1000 chosen due to its performance, a range between 700 and 1200 are further examined. In the end, 700, 800 and 800 are chosen vector size for name, ingredients and steps, respectively.

These dimension-reduced feature are then concatenated together into a single vector before feeding into ML classifiers. Due to weakness mentioned in Section 3.3.1, DT, RF are not chosen to trained on BOW. We sequentially evaluating the following model performance: SVM, XGboost and Logistic Regression. Since these models are computationally expensive except for LR, we only have chances to examine some combination of hyperparameter that may improve the performance. Summary results of classifier trained on BOW are presented in Table 5.

Table 5: Summary performance of classifiers on doc2vec

| Classifier | Accuracy | Macro-f1 | Micro-f1 |
|---|---|---|---|
| Zero-R | 0.63 | | |
| Xgboost | 0.79 | 0.74 | 0.79 |
| LinearSVC | 0.80 | 0.77 | 0.80 |
| LR | 0.80 | 0.76 | 0.80 |

It is observed that, with both SVM and LR output outstanding result. Not only accuracy but also Macro-f1, demonstrating their power in dealing with imbalance classes and curse of dimension. Both these classifiers have really close performance, however, since, LR is much less computational expensive, we choose it as our best classifier

## 4 Discussion

Ensemble of method have been tried, however, due to training time is too large, we were not able to finish training process. Stacking of three model SVM, LR and XGboost with final classifier LR is likely to boost current result. For further improvement of performance, it is possible to extract n-gram feature, concatenating with BOW will provide more information for model.

## References


Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1188–II–1196. JMLR.org.

Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley. 2019. Generating personalized recipes from historical user preferences. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Yijun Shao, Stephanie Taylor, Nell Marshall, Craig Morioka, and Qing Zeng-Treitler. 2018. Clinical text classification with word embedding features vs. bag-of-words features. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2874–2878. IEEE.