RecFormer: One Model To Rule Them All

Derrick Adams
Department of Chemical and
Biomolecular Engineering, KAIST
Daejeon, South Korea
derrad@kaist.ac.kr

Dongjae Lee
Graduate School of AI, KAIST
Daejeon, South Korea
dongjae.lee@kaist.ac.kr

Nikita Sokovnin
School of Computing, KAIST
Daejeon, South Korea
sokovnik@fel.cvut.cz

ABSTRACT

This work explores several strategies to develop the most accurate models to solve two problems: identifying a cuisine when a recipe is given and inferring an ingredient missing from a recipe. Therefore, we focus on two tasks: classification and completion. As an exploratory stage, we implement two categories of models: baseline and target. Baseline models consist of Logistic Regression, Random Forest Classifier, Multinomial Naive Bayes, Linear Support Vector Classifier, and Collaborative Filtering, while our target model is based on Transformer architecture. We propose a model named **RecFormer - Recipe Transformer - that can perform both classification and completion tasks in a single model with similar or better accuracy than our baseline models**. Finally, we employ model blending to achieve maximum performance. An ensemble of 2 models, LSVC and RecFormer, achieves **79.20%** and **14.74%** validation accuracy for classification and completion tasks.

1 INTRODUCTION

This project involves designing, implementing, and evaluating an approach to finding recipe labels and inferring missing ingredients in recipes. Therefore, given a data set with 20 classes of cuisines, the goal is to identify the type of cuisine of a recipe. Additionally, when there is a missing ingredient in a given recipe, we should be able to predict that. Finding a solution to this problem is essential for searching for recipes, making recommendations to customers, and enabling easy contribution to user recipe websites. For example, when users want to provide information about a recipe, they are often faced with long fields of forms that are likely to be abandoned. Therefore, one way to assist the easy entry of this information is to develop an algorithm that automatically categorizes cuisines when provided recipes.

However, solving those problems is challenging for the following reasons. First, classifying cuisine based on the ingredient is complex because its recipes vary substantially in their type of dishes, such as dessert, main dish, etc. Second, finding the missing ingredient is extremely hard due to the large set of possible answer candidates. To solve these problems, we present a Transformer based model that efficiently classifies cuisines and finds the missing ingredient. The contributions of our approach are as follows:

- Based on detailed analysis of various conventional machine learning models, we demonstrate its accuracy and on given dataset to find the best baseline model.
- Our proposed RecFormer model is a multi-task model which can both perform classification and completion tasks with high performance.
- We solve classification and completion task efficiently by creating an ensemble model of best baseline model and RecFormer which shows **79.20%** and **14.74%** for classification and completion task respectively.

The rest of the report is organized as follows. Section 2 explains our model development approach, and Section 3, 4 discusses popular conventional machine algorithms and introduces our proposed model RecFormer respectively. Section 5 evaluates conventional machine learning models, RecFormer, and ensemble model. Lastly, Section 6 discusses future work of RecFormer and concludes the report.

2 MODEL DEVELOPMENT APPROACH

To develop the most efficient and computationally sound model for the given tasks, we designed a plan for the best model discovery, as shown in Figure 1. We categorized our model exploration into baseline and target models. The baseline models are conventional machine learning algorithms such as logistic regression (LogR), random forest classifier (RFC), multinomial naïve Bayes (MNB), linear support vector classifier (LSVC) and collaborative filtering (CF). On the other hand, the target model is a transformer-based algorithm with the flexibility of dual training. The idea for going by this approach was to judge whether our target model is worth exploring based on the accuracy of classification and model flexibility.

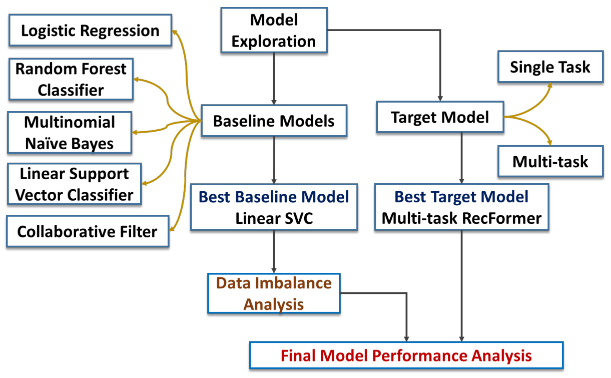


Figure 1: Process flow diagram for model development approach.

Hence, the order of best model discovery is as follows. First, a preliminary performance test was done for the baseline models except for the collaborative filter on the classification task using two vectorizers: term frequency-inverse document frequency (Tfidf) and count vectorizers. Meanwhile, the two versions of the target model were also analyzed. The resulting best models of the two categories were then fine-tuned. Next, several strategies were implemented to understand its impact on model performance due to the imbalance in the received data. Finally, the best baseline and target models with the appropriate data and vectorizer were selected for model performance analysis.

3 BASELINE MODELS

In this section, we introduce popular conventional methods which are the baseline candidates and its limitations. The quantitative analysis will be done in Section 5.

3.1 Logistic Regression

Logistic Regression is a parameterized classification model often widely used in statistical analysis. It has also gained ground in machine learning, where the model's optimal parameters are determined based on historical data. However, since logistic regression can only classify 2 classes, we used multi-label logistic regression. This is achieved by using the softmax function that gives you the probability distribution of the cuisines based on the recipe. The highly probable label is chosen for any given recipe based on the distribution. Although this algorithm is widely used, the major limitation of logistic regression is the assumption of linearity between the dependent variable and the independent variables.

3.2 Random Forest Classifier

Random Forest Classifier is an ensemble model built on multiple decision trees merged to make a prediction. In this classifier, cuisine types are predicted as votes from the different trees. The model's prediction considers the cuisine that receives the most votes as the most probable class. Random decision forests correct for decision trees' habit of overfitting to their training set[1]. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance[2].

3.3 Multinomial Naive Bayes

Naive Bayes classifier assumes that all of the input feature elements are conditionally independent to the given label which is called Naive Bayes assumption. [3] Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers. [4]. However, due to the limitation of Naive Bayes assumption which implies the independence of input features, Multinomial Naive Bayes classifier shows its limitation to complex classification tasks.

3.4 Support Vector Classifier

Support vector machine is maximum margin methods that allow the model to be written as a sum of the influences of a subset of the training instances. It is a supervised learning method used for classification which is widely used for a number of reasons First, it is a discriminant-based method and uses Vapnik's principle to never solve a more complex problem as a first step before the actual problem. Second, After training, the parameter of the linear model, the weight vector, can be written down in terms of a subset of the training set, which are the so-called support vectors, thus lead to memory efficient training. Lastly, kernel-based algorithms are formulated as convex optimization problems, and there is a single optimum that we can solve for analytically.[5].

However, support vector machine have some limitations. First, support vector machine does not provide probability estimates directly, there is no probabilistic clarification for the classification. Second, it is difficult to understand and interpret the final model, variable weights and individual impact. Lastly, support vector machine has difficulties doing small calibrations to the model.

3.5 Collaborative Filtering

Collaborative filtering is a conventional recommendation algorithm based on a simple assumption. It is based on a idea that similar food recipes will have similar ingredients included. For example, if we are trying find a missing ingredient for a 'pepperoni pizza', we can reference all of the pizza recipes to figure out the missing ingredient. Collaborative algorithm has gained popularity after winning the Netflix competition[6]. To explain in detail, collaborative filtering is implemented by an algorithm called Matrix factorization. Matrix factorization algorithms work by decomposing the user-item correlation matrix into the inner product of two lower dimension matrices. However, this algorithm has limitations. Since this algorithm is based on linear production operation, which is suitable for only finding simple relationships, it has challenges to figure out the complex interaction between the cuisine and the recipe. To overcome this challenge, a more deep and complex model need to be applied for completion task.

4 TARGET MODEL

We now introduce and explain our novel proposed model named RecFormer, which is highly inspired by the Transformer model.

4.1 Why Transformer?

Our solution is based on transformers [7], a famous architecture for working with sequences. Transformers were initially used to solve Natural Language Processing tasks [8]. Nowadays, transformers have become a general architecture that can solve many deep learning problems. They have a much lower inductive bias than task-specific architectures such as LSTMs or CNNs. For example, some works adapt transformers for the field of computer vision [9].

Transformers is an encoder-decoder architecture. The main building block of transformers is a self-attention mechanism that computes a representation of the sequence. Basically, self-attention uses inner products of elements in the sequence to create weights, which are used to better represent the element in the context. Therefore, the output of each token depends not only on the token itself but also on other tokens. In order to capture different types of relationship between elements, transformers use multi-head self-attention. The intuitive idea behind multi-head attention: "Each head looks at a different thing". Conventional transformer-based neural networks also use positional encoding to incorporate information about the order of tokens, which is useful for the task where positions play an important role.

4.2 Proposed Approach: RecFormer

Given the data format for our tasks, the transformer seems to be a suitable architecture since it can effectively deal with variable-length input sequences. In our case, tokens are single ingredients. The main difference between our architecture and the classic architecture is that we do not use positional encoding because the ingredients in a recipe do not have a meaningful order. In addition, we incorporate the concept of multitask learning, which helps the model to generalize better by using common layers for multiple tasks. Our model has an Encoding layer to learn embeddings that represent ingredients, a transformer layer that computes features for each element in the sequence, and two heads: one for the classification task and one for the completion task. Both classification and completion heads have the same input dimension but different output dimensions: the number of cuisines and the number of ingredients, respectively. Our architecture is illustrated in Figure 2.

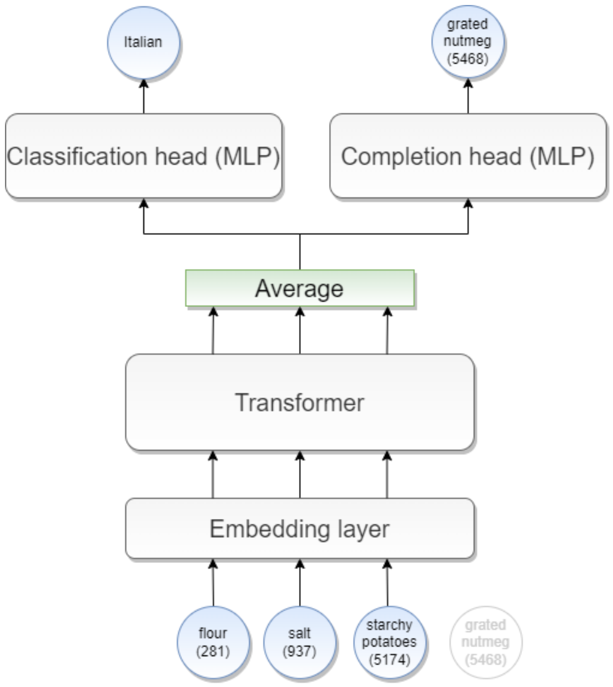


Figure 2: Architecture of RecFormer.

5 EXPERIMENTS

In this section, we discuss not only RecFormer, but also baseline models. The reason we discuss baseline model in detail is to judge whether RecFormer is worth exploring based on the accuracy for both tasks and its flexibility.

5.1 Dataset Exploration

To gain insight into the distribution of the training data, we counted the number of each cuisine, as shown in Figure 3. From the distribution, the Italian and Mexican cuisine dominates the data. This could introduce some bias into the behavior of the model. Therefore, data imbalance strategies were used to determine the impact of this distribution on the model performance. The effect of the strategies is presented in Section 5.3.1.

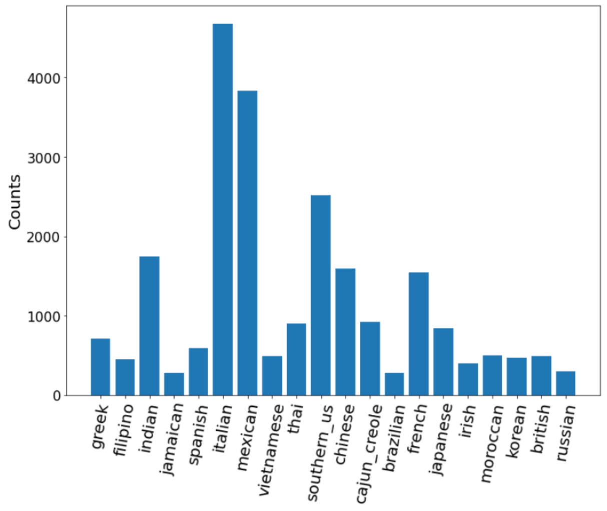


Figure 3: Counts of all cuisines in the training dataset.

5.2 Dataset construction

5.2.1 Classification task. Since the received data had IDs for the recipe ingredients, we used IDs for the model training. First, the cuisines for each recipe were separated to create labels. The IDs were then converted into strings to form recipe strings, which were used as input for the models. The most straightforward way to obtain a multi-task data set is to use the same approach for the completion task and add the second label for each instance, cuisine. In this case, nothing changes for completion, but the number of ingredients for classification will be lower, which may cause worse accuracy. However, during training, the model will encounter the same recipe multiple times with different missing ingredients and may learn to generalize better.

5.2.2 Completion task. To construct a training data set for the completion task, we exclude one different ingredient N times and consider the excluded ingredient as a label for each recipe of length N. The size of the obtained data set is 253453, which is more than 10 times larger than the initial data set.

5.3 Model Exploration

5.3.1 Best baseline model selection. Figure 4 shows the preliminary test results of the baseline models, except for the collaborative filter on classification. This test used stratified three-fold cross-validation to analyze the baseline models on the classification train data. The recipes were vectorized using count and Tfidf vectorization for all the models. The results show that the MNB has the lowest overfitting behavior for both vectorizers, although its f1 scores were on average the smallest. On the other hand, RFC highly overfitted the data, which is evident in its f1 scores of the validation data, closely following the performance of the MNB on the validation data. Logistic regression was the best with the count vectorizer (f1 score: 0.7456), while LSVC was the best with the Tfidf vectorizer (f1 score: 0.7557). Therefore, **LSVC with Tfidf vectorizer was chosen as the best baseline model** for further analysis and comparison with the target model.

Overfitting reduction and class imbalance contribution strategies. Although LSVC with the Tfidf vectorizer gave the best results in the preliminary test, the score for its training data set was significantly higher than the validation set, as with the other baseline models. Therefore, we tried to deal with this overfitting and possible class imbalance contributions. We used balanced weight, oversampling, undersampling, and Chi-square to examine the impact on the model performance. The results are presented in Table 1. The balanced weight class argument weighs classes inversely proportional to their frequency. Oversampling randomly selects instances from the dataset with minor classes, replacing and adding to the training data. Undersampling behaves the opposite to oversampling. Finally, Chi-square uses the chi-square test for the ingredient features and selects the top K significant ones to use for the model's training.

Strategy	Accuracy (%)
Original LSVC	78.05
Balanced Weight	77.69
Oversample	73.53
Undersample	67.67
Chi-square	73.02

Table 1: Results for overfitting and class imbalance contribution exploration. Accuracy is measured on the validation dataset.

From Table 1, none of the strategies implemented on the LSVC improved the accuracy. The undersampling strategy gave the worst validation data accuracy. Nevertheless, some of them reduced the overfitting in the training and validation f1-scores for the cuisines, which is detailed in Table B1 in the Appendix. For example, the chi-square approach significantly reduced overfitting in all cuisines, comparing their f1-scores with those of the original LSVC. However, a substantial fraction of the accuracy was sacrificed, resulting in a lower accuracy than the original LSVC. A similar observation can be made for the undersample LSVC. The balanced weight LSVC was very close to the original at a margin of 0.36% in accuracy. Still, the overfitting wasn't reduced. Head-to-head comparison of cuisines for the original and balanced weight LSVC reveals that in instances where the training f1-scores slightly increased, its validation f1 scores decreased. In general, the strategies were unable to provide us with a good balance between overfitting and accuracy improvement. Therefore, the original LSVC was maintained for comparison with the target model. Additionally, since the results for these strategies were not promising and generally decreased, they were not implemented in our target model as intended.

5.3.2 RecFormer. To compare the performance of single task RecFormer and multi-task RecFormer, we have conducted experiments as shown in Table 2. Table 2 shows a considerable difference between models trained on single task and multi-task. The multitask

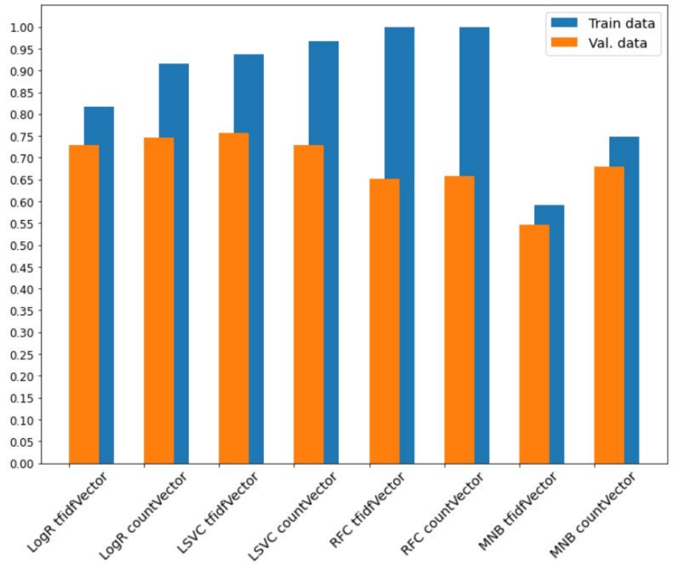


Figure 4: Train and validation data F1 scores over stratified three-fold cross-validation.

model shows better performance and generalizes better. Both heads mutually benefit from having a common transformer layer and training simultaneously on both tasks.

Usage of pretrained word embeddings. We tried to use pretrained word embeddings to improve the multi-task RecFormer performance instead of training an Embedding layer. First, we took GloVe [10] word embeddings of dimension 100 to map each ingredient word and averaged them to receive a single vector per ingredient. Due to pre-trained embeddings, the model converged faster. However, the final accuracy turned out to be worse than before - **75.98%** and **10.95%** for classification and completion, respectively. Then we tried a larger dimension of word embeddings - 300. We managed to achieve **76.63%** and **12.70%**. Pre-trained word embeddings of dimension 300 are used in our final model architecture.

Task (%)	Classification head	Completion head	Multitask
Classification	74.29	-	**76.08**
Completion	-	11.38	**12.44**

Table 2: Comparison of accuracy between single-task and multitask models. Accuracy is measured on the validation set.

5.4 Performance Evaluation

In this subsection, we compare the baseline model with the best performance and the multi-task RecFormer.

5.4.1 Comparison of best models: LSVC and RecFormer. Table 3 shows the final results for the best LSVC and multi-task Recformer on the classification task. A head-to-head comparison of training and validation f1-scores for each cuisine between the LSVC and multi-task RecFormer shows higher values for LSVC, as shown in Table B2 in the Appendix. This demonstrates that the LSVC is better than our multi-task RecFormer at a marginal difference of 1.5% in accuracy. However, the LSVC overfits slightly more those data than the multi-task RecFormer, comparing their training and validation scores.

Dataset/Model	LSVC (%)	RecFormer (%)
Classification train	91.98	85.16
Classification val	78.05	76.63
Completion train	53.80	20.76
Completion val	13.66	12.70

Table 3: Comparison of accuracy of the RecFormer and the best baseline.

5.4.2 Comparison of RecFormer with other baselines. In this section, we compare the accuracy of our proposed idea with the other baselines. Due to the increased number of labels in the completion task, most of the baseline algorithms were unable to handle the complexity of the input vectorization. Hence, we only compare their classification accuracy with the RecFormer. In the completion task, we also implemented collaborative filtering based on matrix factorization. As shown in Table 4, RecFormer outperforms all other baseline algorithms. Beside the RecFormer outperforming all of them, it is able to handle the complexity that these baselines could not overcome in just one multi-way training. This implies that RecFormer is robust.

Model/Task	Classification (%)	Completion (%)
LR count	76.61	-
LR TF-idf	76.41	-
RF count	69.28	-
RF TF-idf	69.07	-
MNB count	72.26	-
MNB TF-idf	70.77	-
Collaborative filtering	-	6.10
RecFormer	**76.63**	**12.70**

Table 4: Accuracy of different baseline models and RecFormer measured on validation data.

5.4.3 Combination of LSVC and RecFormer: Ensemble Model. Considering the close performance of LSVC and RecFormer, we were interested in finding out how their combined effect would cause a change in accuracy. We were motivated by the fact that the combined effect of a neural network-based model RecFormer and the linear model LSVC could augment each other for improved performance. Hence, we formed an ensemble model with them to test the performance. The most straightforward way to combine a few models is to use stacking, or, more specifically, blending. Blending is simply taking a weighted average of the model predictions. Therefore, we blended the two models, which achieved **78.72%** accuracy for classification and **14.48%** accuracy for completion in the validation data. It was interesting to discover that the ensemble model achieves higher accuracy on both tasks than each of the models. Therefore, we used it to make final predictions on the test data.

6 CONCLUSIONS AND FUTURE WORK

Classifying cuisines and finding missing ingredients are essential for recipe websites in making recommendations and meeting customer needs. However, achieving this with a human is difficult and less efficient due to the complexity of input features. Hence, this project required us to solve these issues categorized as classification and completion tasks. To arrive at best-performing model, we devised two machine learning approaches namely baseline models and a target model. The baseline models served as reference model to evaluate our target model. Among the baseline models, LSVC outperformed the others, achieving validation classification and completion accuracy of 78.05% and 13.6%, respectively. On the other hand, the target model which is a Transformer-based multi-task model, RecFormer, utilizing pretrained embedding attained 76.63% and 12.70% accuracy for classification and completion tasks accordingly. While these are at marginal difference with best baseline model, it outperformed the other baseline models. The remarkable flexibility about our proposed model is that, it does the classification and completion task in just one training. This demonstrates its robustness as most of the baseline models could not run the completion task. Furthermore, we combined the best baseline model and our proposed model to form an ensemble model to benefit from their individual strengths. This increased the validation accuracy to **79.20%** and **14.74%** for classification and completion tasks respectively.

There were several ideas that we intended to implement but due to time constraints we could not carry them out. Therefore, we would like to share a few recommendations for possible performance improvement. First, words such as 'chocolate chips', 'chocolate chunks' can be reduced to the word 'chocolate' and remove words with cooking measures. By our experiment, the feature size of 6714 can be reduced to 5196 which is 22.6% decrease in the size of the feature vector. Second, words with little occurrence can be eliminated. We have carried out a simple experiment to eliminate ingredients less than 5 and the feature size of 6714 can be reduced to 2673, which is a 60% decrease of feature vector size.

REFERENCES

[1] Friedman Jerome Hastie Trevor, Tibshirani Robert. The elements of statistical learning (2nd ed.), 2008.
[2] Tamer E. Piryonesi, S. Madeh; El-Diraby. Using machine learning to examine impact of type of performance indicator on flexible pavement deterioration modeling, 2021.
[3] Andrew Ng. Cs229 lecture notes, 2019.
[4] Peter Russell, Stuart; Norvig. Artificial intelligence: A modern approach (2nd ed.), 2003.
[5] Ethem Alpaydin. Introduction to machine learning, 2014.
[6] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.
[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv*

preprint arXiv:1810.04805, 2018.

[9] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[10] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

A APPENDIX

A.1 Labor Division

The team performed the following tasks

- Model implementation of RecFormer [Nikita]
- Model implementation of conventional ML models for accuracy comparison [Derrick]
- Model implementation of collaborative filter based on matrix factorization for accuracy comparison[Dongjae]
- Research on data preprocessing [Nikita and Derrick]
- Data preprocessing implementation and experiment[Dongjae]
- Writing report [all]

A.2 Full disclosure wrt dissertations/projects

Derrick Adams: He is not doing any project or dissertation related to this project: his thesis is on model predictive inferential control for chemical processes.

Dongjae Lee: He is not doing any project or dissertation directly related to this project: his thesis is on Process-in-Memory for AI acceleration

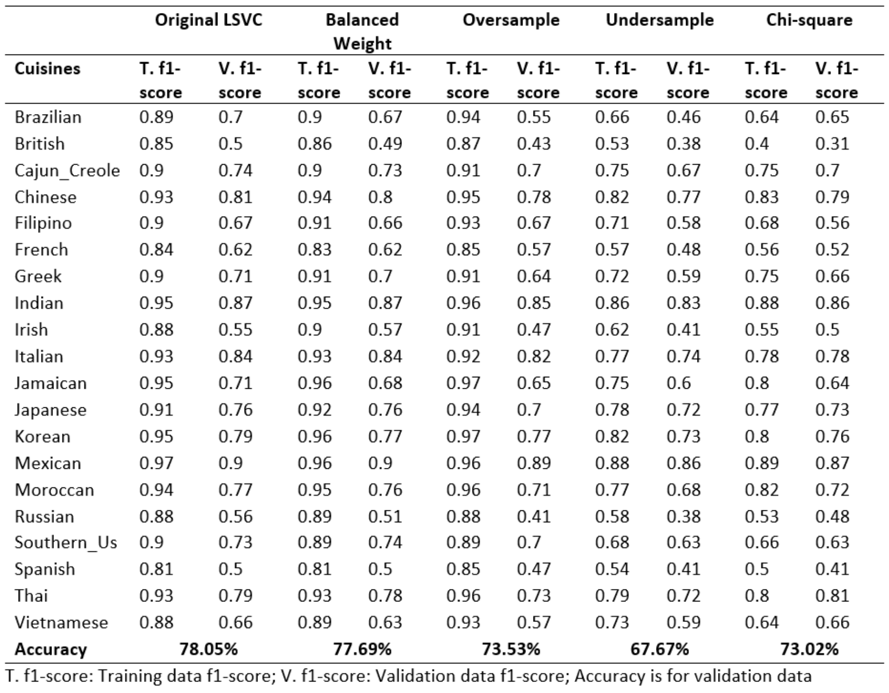
Nikita Sokovnin: His projects are not directly related to transformers, his bachelor thesis was about Recognizing Unknown Objects for Open-Set 3D Object Detection. However, he participated in several reading groups about transformers and has some experience in training such models.

A.3 RecFormer hyperparameters

- Pre-trained embeddings (GloVe) dimension: 300
- Number of encoder layers: 3
- Number of decoder layers: 1
- Number of self-attention heads: 4
- Dimension of feedforward network: 2048
- Dropout probability: 0.3
- Number of epochs: 32
- Classification loss weight: 0.5
- Completion loss weight: 2

B DETAILED RESULTS

B.1 Table B1: Results for overfitting and class imbalance contribution exploration.

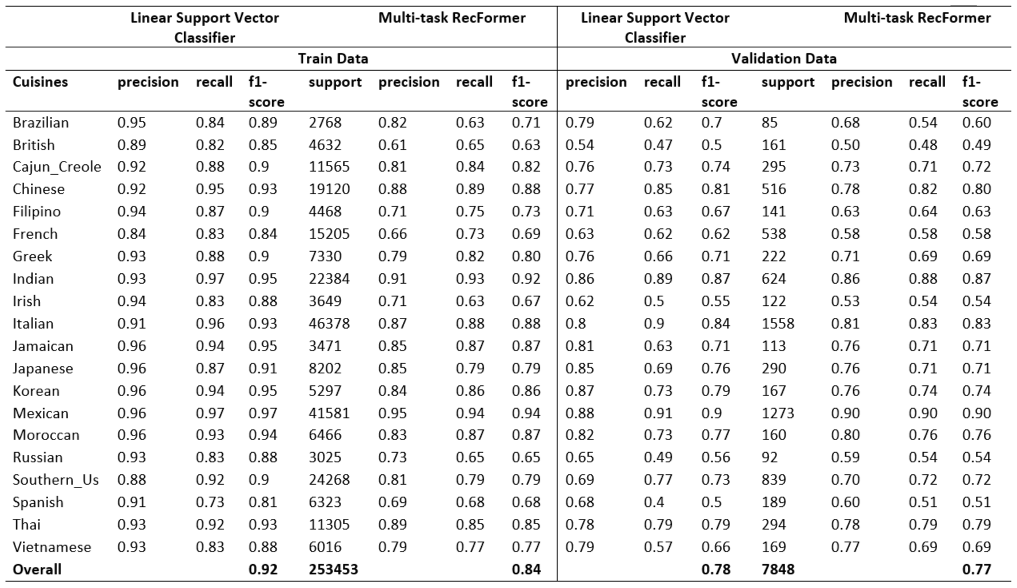


	Original LSVC		Balanced Weight		Oversample		Undersample		Chi-square	
Cuisines	**T. f1-score**	**V. f1-score**	**T. f1-score**	**V. f1-score**	**T. f1-score**	**V. f1-score**	**T. f1-score**	**V. f1-score**	**T. f1-score**	**V. f1-score**
Brazilian	0.89	0.7	0.9	0.67	0.94	0.55	0.66	0.46	0.64	0.65
British	0.85	0.5	0.86	0.49	0.87	0.43	0.53	0.38	0.4	0.31
Cajun_Creole	0.9	0.74	0.9	0.73	0.91	0.7	0.75	0.67	0.75	0.7
Chinese	0.93	0.81	0.94	0.8	0.95	0.78	0.82	0.77	0.83	0.79
Filipino	0.9	0.67	0.91	0.66	0.93	0.67	0.71	0.58	0.68	0.56
French	0.84	0.62	0.83	0.62	0.85	0.57	0.57	0.48	0.56	0.52
Greek	0.9	0.71	0.91	0.7	0.91	0.64	0.72	0.59	0.75	0.66
Indian	0.95	0.87	0.95	0.87	0.96	0.85	0.86	0.83	0.88	0.86
Irish	0.88	0.55	0.9	0.57	0.91	0.47	0.62	0.41	0.55	0.5
Italian	0.93	0.84	0.93	0.84	0.92	0.82	0.77	0.74	0.78	0.78
Jamaican	0.95	0.71	0.96	0.68	0.97	0.65	0.75	0.6	0.8	0.64
Japanese	0.91	0.76	0.92	0.76	0.94	0.7	0.78	0.72	0.77	0.73
Korean	0.95	0.79	0.96	0.77	0.97	0.77	0.82	0.73	0.8	0.76
Mexican	0.97	0.9	0.96	0.9	0.96	0.89	0.88	0.86	0.89	0.87
Moroccan	0.94	0.77	0.95	0.76	0.96	0.71	0.77	0.68	0.82	0.72
Russian	0.88	0.56	0.89	0.51	0.88	0.41	0.58	0.38	0.53	0.48
Southern_Us	0.9	0.73	0.89	0.74	0.89	0.7	0.68	0.63	0.66	0.63
Spanish	0.81	0.5	0.81	0.5	0.85	0.47	0.54	0.41	0.5	0.41
Thai	0.93	0.79	0.93	0.78	0.96	0.73	0.79	0.72	0.8	0.81
Vietnamese	0.88	0.66	0.89	0.63	0.93	0.57	0.73	0.59	0.64	0.66
Accuracy	**78.05%**		**77.69%**		**73.53%**		**67.67%**		**73.02%**	

T. f1-score: Training data f1-score; V. f1-score: Validation data f1-score; Accuracy is for validation data

Table B1

B.2 Table B2: Comparison of LSVC performance with Multi-task RecFormer on classification task.



	Linear Support Vector Classifier				Multi-task RecFormer			Linear Support Vector Classifier				Multi-task RecFormer		
	Train Data							Validation Data						
Cuisines	**precision**	**recall**	**f1-score**	**support**	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**	**support**	**precision**	**recall**	**f1-score**
Brazilian	0.95	0.84	0.89	2768	0.82	0.63	0.71	0.79	0.62	0.7	85	0.68	0.54	0.60
British	0.89	0.82	0.85	4632	0.61	0.65	0.63	0.54	0.47	0.5	161	0.50	0.48	0.49
Cajun_Creole	0.92	0.88	0.9	11565	0.81	0.84	0.82	0.76	0.73	0.74	295	0.73	0.71	0.72
Chinese	0.92	0.95	0.93	19120	0.88	0.89	0.88	0.77	0.85	0.81	516	0.78	0.82	0.80
Filipino	0.94	0.87	0.9	4468	0.71	0.75	0.73	0.71	0.63	0.67	141	0.63	0.64	0.63
French	0.84	0.83	0.84	15205	0.66	0.73	0.69	0.63	0.62	0.62	538	0.58	0.58	0.58
Greek	0.93	0.88	0.9	7330	0.79	0.82	0.80	0.76	0.66	0.71	222	0.71	0.69	0.69
Indian	0.93	0.97	0.95	22384	0.91	0.93	0.92	0.86	0.89	0.87	624	0.86	0.88	0.87
Irish	0.94	0.83	0.88	3649	0.71	0.63	0.67	0.62	0.5	0.55	122	0.53	0.54	0.54
Italian	0.91	0.96	0.93	46378	0.87	0.88	0.88	0.8	0.9	0.84	1558	0.81	0.83	0.83
Jamaican	0.96	0.94	0.95	3471	0.85	0.87	0.87	0.81	0.63	0.71	113	0.76	0.71	0.71
Japanese	0.96	0.87	0.91	8202	0.85	0.79	0.79	0.85	0.69	0.76	290	0.76	0.71	0.71
Korean	0.96	0.94	0.95	5297	0.84	0.86	0.86	0.87	0.73	0.79	167	0.76	0.74	0.74
Mexican	0.96	0.97	0.97	41581	0.95	0.94	0.94	0.88	0.91	0.9	1273	0.90	0.90	0.90
Moroccan	0.96	0.93	0.94	6466	0.83	0.87	0.87	0.82	0.73	0.77	160	0.80	0.76	0.76
Russian	0.93	0.83	0.88	3025	0.73	0.65	0.65	0.65	0.49	0.56	92	0.59	0.54	0.54
Southern_Us	0.88	0.92	0.9	24268	0.81	0.79	0.79	0.69	0.77	0.73	839	0.70	0.72	0.72
Spanish	0.91	0.73	0.81	6323	0.69	0.68	0.68	0.68	0.4	0.5	189	0.60	0.51	0.51
Thai	0.93	0.92	0.93	11305	0.89	0.85	0.85	0.78	0.79	0.79	294	0.78	0.79	0.79
Vietnamese	0.93	0.83	0.88	6016	0.79	0.77	0.77	0.79	0.57	0.66	169	0.77	0.69	0.69
Overall			**0.92**	**253453**			**0.84**			**0.78**	**7848**			**0.77**

Table B2