Classification of Rare Recipes Requires Linguistic Features as Special Ingredients

Elham Mohammadi[1,2,3], Nada Naji[1], Louis Marceau[1], Marc Queudot[1,3], Eric Charton[1], Leila Kosseim[2], and Marie-Jean Meurs[3][0000−0001−8196−2153]

[1] Banque Nationale du Canada (BNC), Montreal, QC, Canada
[2] Concordia University, Montreal, QC, Canada
[3] Université du Québec à Montréal (UQAM), Montreal, QC, Canada
{nada.naji,louis.marceau,eric.charton}@bnc.ca,
{elham.mohammadi,leila.kosseim}@concordia.ca
queudot.marc@courrier.uqam.ca
meurs.marie-jean@uqam.ca

Abstract. In this paper, we propose a joint model, composed of neural and linguistic sub-models, to address classification tasks in which the distribution of labels over samples is imbalanced. Different experiments are performed on tasks 1 and 2 of the DEFT 2013 shared task [10]. In one set of experiments, the joint model is used for both classification tasks, whereas the second set of experiments involves using the neural sub-model, independently. This allows us to measure the impact of using linguistic features in the joint model. The results for both tasks show that adding the linguistic sub-model improves classification performance on the rare classes. This improvement is more significant in the case of task 1, where state-of-the-art results are achieved in terms of micro and macro-average F1 scores.

Keywords: Text classification · Deep learning · Linguistic features.

1 Introduction

Different techniques have been used to address the task of text classification throughout the years. In the era of statistical Natural Language Processing (NLP), machine learning approaches were used to automatically extract discriminative linguistic features that would be helpful for the specific task at hand. With the availability of larger corpora and the advent of deep learning, neural network architectures have become increasingly popular in performing different NLP tasks [1], including text classification. However, despite the advances made by deep learning and achieving state-of-the-art results in many tasks, not many studies have addressed the challenge of an imbalanced dataset, which is the case in many real-life scenarios and applications [13].

In this paper, we focus on 2 multi-class classification tasks with an imbalanced distribution of labels over data and measure the effectiveness of different methods in handling such a challenge. The tasks in question are the classification of cooking recipes into one of 4 difficulty levels (*Very Easy*, *Easy*, *Fairly Difficult*, and *Difficult*), and the classification of recipes based on meal type (*Starter*, *Main Dish*, or *Dessert*). The datasets, which are both imbalanced, are taken from the DEFT (Defi Fouille de Texte) 2013 shared task [10], tasks 1 and 2. To address the tasks, we experiment with two different architectures. The first architecture is a neural model which uses pretrained embeddings as input features. The second architecture is a joint model that is composed of neural and linguistic sub-models. We perform experiments using both architectures and 2 types of pretrained embeddings and measure the effectiveness of using a joint model in handling an imbalanced class distribution.

The rest of this paper is organized as follows: Section 2 goes over related previous work. Section 3 presents a statistics summary of the datasets that are used. In Section 4, the overall model architecture, the sub-models, and different utilized model configurations are presented. Section 5 includes the results achieved by different models and discusses their implications. Finally, Section 6 concludes this work.

2 Related Work

According to [13], 3 general approaches for handling imbalanced data in machine learning have been proposed:

1. Data-level approaches: These approaches involve changing the class distribution through under-sampling and over-sampling of training data to mitigate the class imbalance. However, both under and over-sampling can pose new challenges. Under-sampling decreases the number of samples, thus ignoring information that is available to further train the model. Over-sampling can lead to the creation of synthetic samples that are biased, since not all of the minority samples are included in the over-sampling process [18].
2. Algorithmic approaches: Instead of making modifications to the training data, algorithmic approaches adapt the process of learning to take into account the class imbalance. An example of this would be using class weights in the loss function to assign more penalty to a mistake made on a less frequent class compared to a more frequent one. The challenge in using these approaches is finding an optimal penalty matrix which can result in a better and less biased learning. Moreover, for extremely imbalanced data, such a method can make the classifier more prone to making mistakes on the more frequent classes, leading to a drop in overall performance [14].
3. Hybrid approaches: Data-level and algorithmic approaches can be combined for a better handling of an imbalanced distribution.

However, the mentioned approaches might not be helpful in the case of an extreme class imbalance. One possible avenue is extracting discriminative features, taking into account the imbalance present in the training samples [14]. Discriminative features which are extracted and used alongside distributed representations in a deep architecture have been shown to improve results in a variety of tasks, even when the main challenge is not class imbalance. To do non-factoid answer reranking, [3] use a recurrent architecture that encodes questions and answers, separately. A similarity matrix is calculated on the encoded pairs, followed by a Multi-Layer Perceptron (MLP) that performs the prediction. The results show that it is possible to improve the model by passing additional discourse features to the MLP, alongside features that are learnt through the recurrent architecture.

Researchers in [2] develop a model for Part-Of-Speech (POS) tagging that is made up of a bidirectional Long Short-Term Memory (BiLSTM) network, followed by a Conditional Random Field (CRF) layer that predicts the tags. They enrich this model with manually designed features at the embedding layer and achieve an improved performance, showing that combining manually designed and automatically learnt features can benefit such a task in the absence of a large annotated dataset.

For the classification of short texts, [19] employ a joint model which utilizes both implicit (pretrained word embeddings and character embeddings) and explicit (principal concepts in a text, extracted through a knowledge base) representations of texts. Feeding these representations to a multi-branch convolutional model, they achieve state-of-the-art results. Their results indicate that enriching the sample features through a knowledge base can result in a better classification.

In this work, we experiment with the addition of linguistic features to neural-based models and measure the difference in performance, with a focus on minority classes.

3 Datasets

The datasets that were used for our experiments have been taken from the DEFT 2013 shared task [10], task 1 and task 2. The dataset for task 1 consists of French cooking recipes that have been labelled with their respective level of difficulty on a 4 point scale that ranges from *Very Easy* to *Difficult*. As Table 1 shows, the

Table 1. Statistics of the train, development, and test datasets for task 1

Difficulty Level	**Train**		**Development**		**Test**	
	# of Samples	**Percentage**	**# of Samples**	**Percentage**	**# of Samples**	**Percentage**
Very Easy	5569	50.2%	1393	50.2%	1132	49.0%
Easy	4601	41.5%	1151	41.5%	968	41.9%
Fairly Difficult	855	7.7%	213	7.7%	189	8.2%
Difficult	64	0.6%	16	0.6%	20	0.9%
Total	11089	100.0%	2773	100.0%	2309	100.0%

Table 2. Statistics of the train, development, and test datasets for task 2

Meal Type	**Train**		**Development**		**Test**	
	# of Samples	**Percentage**	**# of Samples**	**Percentage**	**# of Samples**	**Percentage**
Starter	2599	23.4%	647	23.3%	562	24.4%
Main Dish	5167	46.6%	1280	46.1%	1084	47.0%
Dessert	3323	30.0%	846	30.5%	661	28.6%
Total	11089	100.0%	2773	100.0%	2307	100.0%

distribution of labels in this dataset is very imbalanced, with more than 90% of the samples with either a *Very Easy* or *Easy* label, and a significantly smaller number of samples with a *Fairly Difficult* or *Difficult* label.

The dataset for task 2 consists of French cooking recipes that have been labelled with the recipe's meal type, *Starter*, *Main Dish*, or *Dessert*. Although the distribution of labels in this dataset is not as imbalanced as the dataset for task 1, nearly half of the samples belong to the class *Main Dish*, causing the challenge of an imbalanced dataset in task 2, as well.

Finally, it should be noted that originally, for the DEFT 2013 shared task, the data for both tasks was released in the two stages of training and testing. For the experiments reported in this paper, 20% of the released training data was set aside for model validation (referred to as *Development* data in Tables 1 and 2).

4 Model Design

The joint model that we have developed for the classification of recipes is composed of two sub-models. The first sub-model is neural-based and the second sub-model utilizes linguistic features.

In this section, first, the architecture of the neural sub-model is presented. We then describe the linguistic features that were used to complement the extracted neural features and how the two parts of the model are combined for the final classification.

4.1 The Neural Sub-model

The Embedding Layer. The embedding layer is used to transform the concatenation of the preparation section (the main body) and the title of a recipe into dense vectors. In this work, two different types of pretrained transformer-based embeddings are used: the multilingual cased version of BERT embeddings [9], and CamemBERT embeddings [17] which have been trained only on French data using a BERT model. For both BERT and CamemBERT, only the features from the last layer of the models are extracted, resulting in a contextual dense representation of size 768 for each token.

It should be noted that all samples are limited to their first 100 tokens, and zero padding is used for the samples with less than 100 tokens.

The output of the embedding layer is then passed to either a recurrent or a convolutional architecture.

The Recurrent Architecture. For the hidden layer of the recurrent architecture, Gated Recurrent Units (GRUs) [6] were used, since GRUs are less prone to overfitting [7] because of having a smaller number of parameters (compared to LSTMs [12]). A bidirectional GRU is used to process the embeddings consecutively in a forward and a backward pass. The output of the GRU layer is then passed to an attention layer which calculates its weighted average using Equation 1:

$$Attention = \sum_{t=1}^{n} y_t \omega_t \tag{1}$$

In the equation above, y_t stands for the output of the GRU layer at timestep t and ω_t refers to the weight assigned to y_t by the attention mechanism. The weight vector ω is calculated using the following process: First, a single N-to-1 fully connected layer is applied on the output of the hidden layer at each timestep (N being the size of the output representation at a timestep), resulting in a scalar for each timestep. The scalars for all timesteps are then concatenated and a softmax activation function is applied over them, producing the weight vector ω.

The Convolutional Architecture. In some experiments, a convolutional architecture is used instead of a recurrent one. In these experiments, first, a Convolutional Neural Network (CNN) [15] is used to process N-grams of input (N consecutive token representations) using convolution filters of size N. The hidden layer is followed by a pooling layer which is either average or max pooling or a combination of the two. Average pooling computes an average over the outputs of the hidden layer, while for max pooling, first, the output vectors of the hidden layer are passed through a Concatenated Rectified Linear Unit (CReLU) activation function.

4.2 The Linguistic Sub-model

The primary part of the linguistic sub-model is a feature extractor, which transforms each sample into a set of linguistic features proposed by [5]. The features extracted for each task are explained below.

Task 1 Features. For task 1, i.e. the classification of recipes based on their level of difficulty, the following features are used:

The number of tokens in the recipe title, the number of tokens in the recipe preparation part, the number of ingredients mentioned in the ingredient list of the recipe, the cost of the meal on a 3-point scale, the presence of 22 predefined words that have been identified to be discriminative, the presence of 48 predefined discriminative trigrams, and finally, the number of verbs in the recipe that belong to 3 different discriminative verb groups.

The extraction of these features results in a vector of size 77 for each recipe, which is used by the linguistic sub-model.

Task 2 Features. For this task, which is the classification of recipes based on meal type, the following features are used:

Similar to task 1, the number of tokens in the recipe title, the number of tokens in the preparation section, and the number of ingredients on the ingredient list are computed as the first 3 features. The 4th features is the cost associated with the meal on a 3-point scale. The remaining features consist of the presence or absence of 1231 ingredient names in the recipe, the presence of 48 discriminatve trigrams, and the number of verbs in the recipe belonging to each of the three predefined verb families. In the end, a feature vector of size 1286 is extracted for each recipe.

For a complete description of the selection process of the linguistic features, see [5]. The extracted feature vectors are then passed into a single-layer feedforward neural network, mapping each feature vector to a vector of the same size, resulting in output representations by the linguistic sub-model.

4.3 The Fusion Component

The fusion component first concatenates the output of the two sub-models, then applies a fully connected layer over the resulting vector, mapping them to vectors of size 4 in the case of task 1, and to vectors of size 3 in the case of task 2. The fully connected layer is followed by a softmax activation function that outputs the probabilities of the classes.

In order to measure the effect of the linguistic sub-model, some experiments involved utilizing only the neural sub-model. In those experiments, the fusion component was replaced with a fully connected layer, mapping the output of the attention/pooling layer to the number of classes, followed by a softmax activation function which produced the probability distribution over different classes.

4.4 Training

In order to train the models, a batch size of 32 was used. All models were trained for 20 epochs. The final model parameters were taken from the epoch that included the best micro score on the development dataset. Table 3 contains details regarding the hyperparameters of different models. Some specific important aspects of the training process are explained below.

Optimizer AdamW [16] was used to optimize the training process. For all models in both tasks 1 and 2, an initial learning rate of 10^{-3} was used. This learning rate was adapted for CNN models to a rate of 10^{-4} after two epochs in task 1, and after five epochs in task 2.

Table 3. Hyperparameters for each model, including neural models and joint models with both neural and linguistic sub-models. *#HL / #KH*: Number of hidden layers in recurrent models or kernel height in CNNs. *#HN / #K*: Number of hidden nodes in each recurrent layer or number of kernels in CNNs.

	Model	**Task 1**			**Task 2**		
		#HL / #KH	**#HN / #K**	**Pooling**	**#HL / #KH**	**#HN / #K**	**Pooling**
Neural	CNN-BERT	300, 200, 100, 100	1, 2, 3, 4	max	2	200	max
	GRU-BERT	1	64	attention	2	32	attention
	CNN-CamemBERT	2, 3	250, 50	max	2	200	max, average
	GRU-CamemBERT	2	32	attention	2	64	attention
Joint	CNN-BERT	2	250	max	2	200	max, average
	GRU-BERT	64	1	attention	2	32	attention
	CNN-CamemBERT	1	400	max, average	2	400	max, average
	GRU-CamemBERT	2	32	attention	2	32	attention

Class Weights In order to counter the effect of the imbalanced distribution of labels in both tasks, class weights were used in the utilized cross-entropy loss function. For experiments that utilized only the neural model, weights were automatically calculated, taking into account the proportion of the samples in each class over the number of all training samples for the task. In experiments that used the joint model, weights were manually set to 0.1, 0.1, 0.2, 0.6 (corresponding to the *Very Easy*, *Easy*, *Fairly Difficult*, and *Difficult* classes, respectively) for task 1, and set to 0.6, 0.3, 0.1 (corresponding to classes *Starter*, *Main Dish*, and *Dessert*, respectively) for task 2.

Regularization In order to regularize the network, the optimizer was used with a weight decay rate of 0.02. Moreover, a dropout layer with a probability of 0.2 was applied on the concatenation of the output of the two sub-models in the fusion component in joint models, and on the output of the pooling/attention layer in neural models.

Fine-tuning of BERT and CamemBERT Models Since in the experiments which involved joint and neural models, the embedding layer was kept frozen, BERT and CamemBERT models were fine-tuned on the two tasks as additional experiments. The results achieved by the fine-tuned models are reported in Tables 4 and 6.

5 Results and Discussion

For both tasks, the results have been reported in terms of micro-average score, macro-average F1 score, macro-average precision and macro-average recall. The micro-average score stands for micro-average F1, precision, and recall, all 3 of which are equivalent in this work since evaluation is done on all classes.

The results of different experiments for task 1, alongside the results of DEFT 2013 teams achieving the best micro scores are shown in Table 4. Looking at the results achieved by the models that we have developed, it can be seen that in all cases, using a joint model has resulted in (often highly) superior performance in

Table 4. Task 1 results.

	Model	**Development**				**Test**			
		Micro Score	**Macro F1**	**Macro P**	**Macro R**	**Micro Score**	**Macro F1**	**Macro P**	**Macro R**
Neural	CNN-BERT	61.5	39.3	41.1	37.6	58.8	37.7	39.4	36.1
	GRU-BERT	59.9	38.5	38.5	38.4	58.0	36.8	37.0	36.7
	Finetuned BERT	56.9	39.3	42.9	36.2	55.9	36.0	36.8	35.3
	CNN-CamemBERT	60.9	42.7	43.4	42.0	59.3	38.6	39.9	37.3
	GRU-CamemBERT	62.4	36.1	38.1	34.3	60.1	36.9	40.1	34.1
	Finetuned CamemBERT	61.2	37.3	38.5	36.2	59.3	37.6	38.9	36.4
Joint	CNN-BERT	64.5	49.1	60.0	41.6	62.0	47.3	59.3	39.3
	GRU-BERT	65.8	41.7	45.5	38.5	63.1	39.3	42.1	36.8
	CNN-CamemBERT	**66.4**	50.3	58.5	**44.2**	**63.8**	**50.0**	62.0	**42.0**
	GRU-CamemBERT	65.3	**51.1**	**68.5**	40.8	63.1	40.5	42.5	38.7
Deft 2013 Top Teams	First Team [5]	-	-	-	-	62.5	48.4	**68.2**	37.5
	Second Team [8]	-	-	-	-	61.2	45.1	52.4	39.5
	Third Team [4]	-	-	-	-	59.2	45.3	63.3	35.3

terms of micro and macro F1 scores on both development and test data, compared to a solely neural model. The improvement in results can be observed in terms of macro precision and macro recall, as well. This shows that the features which are captured by a neural network can be complemented by linguistic features, resulting in a better classification.

A second observation is that the highest micro and macro scores (except for macro precision for which the highest score was achieved by [5]) belong to joint models that utilize pretrained CamemBERT embeddings in the neural sub-model. This is to be expected since CamemBERT embeddings have been trained exclusively on French data, as opposed to multilingual BERT embeddings used by other models. Furthermore, Table 4 shows that the joint CNN-CamemBERT model outperformed the best baseline in terms of micro score and macro F1 and recall by the highest margin among all models, achieving state-of-the-art results in task 1.

Table 5 includes per-class results in terms of F1 score for task 1. On both development and test datasets, in 3 out of 4 classes, the best F1 score is achieved by joint models, specifically the ones that utilize CamemBERT as pretrained embeddings. Looking at the results by the 4 models that use CamemBERT embeddings, it can be seen that in all but one case (the 0% F1 score on the *Difficult* class by the GRU-CamemBERT models), adding linguistic features has improved the per-class performance, while this cannot be said about the models that use BERT, showing that the linguistic features have complemented CamemBERT embeddings more effectively than the BERT embeddings.
On the test set, the joint CNN-CamemBERT model achieved F1 scores higher than the best baseline model. This joint model also resulted in the highest F1 score of 25% on the *Difficult* class, which is the rarest among all classes. It should be noted that only 3 out of the 8 models achieved an F1 score higher than 0 on the *Difficult* class, 2 of which are joint models that utilize linguistic features. The per-class results show the effectiveness of linguistic features when the task involves a highly imbalanced dataset.

Table 6 shows the results that were achieved on task 2, by the models that we have developed alongside the models by the three top-performing teams in

Table 5. Task 1 per-class results, in terms of F1 score.

	Model	**Development**				**Test**			
		Very Easy	**Easy**	**Fairly Difficult**	**Difficult**	**Very Easy**	**Easy**	**Fairly Difficult**	**Difficult**
Neural	CNN-BERT	66.3	61.6	25.1	0.0	63.1	**59.7**	23.5	0.0
	GRU-BERT	68.0	56.0	**29.9**	0.0	65.8	55.2	26.1	0.0
	CNN-CamemBERT	71.0	51.9	22.6	19.5	69.9	51.2	19.2	9.8
	GRU-CamemBERT	71.1	56.7	7.3	0.0	68.7	55.0	12.6	0.0
Joint	CNN-BERT	72.1	60.8	24.9	21.1	70.0	58.1	22.5	17.4
	GRU-BERT	73.9	61.1	22.6	0.0	72.0	58.1	18.9	0.0
	CNN-CamemBERT	74.0	**61.8**	27.0	**27.3**	72.2	59.0	25.2	**25.0**
	GRU-CamemBERT	**74.4**	58.3	27.9	11.8	**72.5**	56.3	**29.4**	0.0
Deft 2013 Top Teams	First Team [5]	-	-	-	-	71.7	56.2	18.8	9.5
	Second Team [8]	-	-	-	-	69.2	57.0	26.1	16.0
	Third Team [4]	-	-	-	-	68.6	52.5	15.6	9.5

DEFT 2013. The first observation is that, among our models, the finetuned CamemBERT model achieved the best overall performance on the development dataset. However, this model is outperformed by the joint CNN-CamemBERT model, which was our best model in task 1, in the test phase. This shows that, in general, the joint CNN-CamemBERT model can generalize better to new samples. This model achieved the highest macro F1 score, alongside the top-performing team of DEFT 2013, and also the highest recall among all models, while it fell short of achieving the highest micro score by 0.3% and the highest macro precision by 0.4%.
Table 6 shows that all joint models outperform their neural counterparts in terms of micro and macro scores. However, unlike task 1, this improvement is not big enough to result in state-of-the-art results in terms of the micro-average score. One reason behind this performance can be the linguistic features that were used for task 2. It is possible that, compared to task 1, these features are not as representative of the classes. The higher sparsity of the linguistic feature matrix for task 2 could be another factor. Looking at the results of task 1 in Table 5, it can be observed that after the addition of the linguistic sub-model, when there is improvement, the amount of improvement is significantly higher in the case of

Table 6. Task 2 results.

	Model	**Development**				**Test**			
		Micro Score	**Macro F1**	**Macro P**	**Macro R**	**Micro Score**	**Macro F1**	**Macro P**	**Macro R**
Neural	CNN-BERT	86.4	84.9	85.7	84.2	85.9	84.9	85.6	84.2
	GRU-BERT	84.4	83.2	83.2	83.2	84.8	84.0	84.0	83.9
	Finetuned BERT	86.3	85.8	85.1	86.5	86.4	86.2	85.6	86.8
	CNN-CamemBERT	87.6	86.8	86.5	**87.2**	88.1	87.6	87.6	87.7
	GRU-CamemBERT	86.5	85.6	85.5	85.6	87.1	86.5	86.6	86.4
	Finetuned CamemBERT	**88.2**	**87.1**	**87.3**	86.9	88.1	87.4	87.5	87.3
Joint	CNN-BERT	86.0	85.2	84.9	85.4	87.0	86.5	86.3	86.7
	GRU-BERT	85.0	84.2	83.9	84.6	85.5	85.0	84.8	85.2
	CNN-CamemBERT	87.5	86.8	86.4	87.1	88.6	**88.2**	88.0	**88.3**
	GRU-CamemBERT	86.9	86.1	85.8	86.5	87.8	87.3	87.2	87.4
Deft 2013 Top Teams	First Team [4]	-	-	-	-	**88.9**	**88.2**	**88.4**	88.1
	Second Team [5]	-	-	-	-	85.6	84.7	85.0	84.3
	Third Team [11]	-	-	-	-	84.9	84.1	84.2	84.1

Table 7. Task 2 per-class results, in terms of F1 score.

	Model	**Development**			**Test**		
		Starter	**Main Dish**	**Dessert**	**Starter**	**Main Dish**	**Dessert**
Neural	CNN-BERT	70.2	86.5	97.6	71.0	86.4	96.8
	GRU-BERT	67.9	83.8	97.8	70.5	85.0	96.4
	CNN-CamemBERT	**75.1**	**87.1**	98.2	77.1	88.0	97.7
	GRU-CamemBERT	72.8	86.4	97.4	75.4	87.8	96.4
Joint	CNN-BERT	72.0	85.6	97.7	75.0	87.2	97.3
	GRU-BERT	70.7	84.5	97.2	72.6	85.8	96.5
	CNN-CamemBERT	74.7	86.9	**98.6**	**78.1**	88.5	98.0
	GRU-CamemBERT	73.4	86.3	98.5	76.7	87.8	97.5
Deft 2013 Top Teams	First Team [4]	-	-	-	77.3	**88.8**	**98.6**
	Second Team [5]	-	-	-	70.3	85.6	97.9
	Third Team [11]	-	-	-	69.4	84.8	98.2

rare classes. Therefore, it can be hypothesized that the strength of the proposed joint model is handling an imbalanced distribution of labels, resulting in a more significant improvement of results when the available dataset is more imbalanced. Knowing that the distribution of labels in task 1 is starkly more imbalanced than task 2, the joint model is more effective in the former than in the latter case.
Finally, Table 7 includes the per-class F1 scores achieved by different models on task 2. It also shows the per-class F1 scores achieved by the three top-performing teams in the DEFT 2013 shared task. Among the 8 models that we developed, the results show that the joint CNN-CamemBERT model achieves the highest F scores on all three classes on the test set. It also achieves state-of-the-art results on the class *Starter*, which is the rarest class in the dataset. This is in agreement with the hypothesis that the strength of the joint model is in the handling of rare classes.

6 Conclusion

In this paper, we proposed a joint model for the classification of imbalanced data. The model, composed of a neural and a linguistic sub-model, was utilized to address tasks 1 and 2 of the DEFT 2013 shared task [10], which involved the classification of French recipes based on difficulty level and meal type, using the datasets and the evaluation metrics specific to the two tasks. In order to measure the effect of the linguistic sub-model, experiments were performed using the joint model, while a second set of experiments involved using only the neural sub-model, independently. The results from these experiments show that, in both tasks, the joint models outperform their neural counterparts. In task 1, the joint model could achieve state-of-the-art results in terms of both micro and macro-average F1 scores, showing the effectiveness of this model in cases of highly imbalanced data.

Reproducibility

To ensure reproducibility and comparisons between systems, our source code is publicly released as an open source software in the following repository:
https://github.com/cooking-classification/CAI2020.
The data could be obtained by contacting the DEFT 2013 shared task organizers (see https://deft.limsi.fr/2013/index.php?id=1&lang=en

References

1. Amini, H., Farahnak, F., Kosseim, L.: Natural Language Processing: An Overview. In: Frontiers in Pattern Recognition and Artificial Intelligence, vol. 5, chap. 3, pp. 35–55. World Scientific (June 2019)
2. Bach, N.X., Duy, T.K., Phuong, T.M.: A pos tagging model for vietnamese social media text using bilstm-crf with rich features. In: Pacific Rim International Conference on Artificial Intelligence. pp. 206–219. Springer (2019)

3. Bogdanova, D., Foster, J., Dzendzik, D., Liu, Q.: If you can't beat them join them: Handcrafted features complement neural nets for non-factoid answer reranking. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers. pp. 121–131 (2017)
4. Bost, X., Brunetti, I., Cabrera-Diego, L.A., Cossu, J.V., Linhares, A., Morchid, M., Torres-Moreno, J.M., El-Bèze, M., Dufour, R.: Systémes du LIA à DEFT 13. arXiv preprint arXiv:1702.06478 (2017)
5. Charton, E., Meurs, M.J., Jean-Louis, L., Gagnon, M.: Using collaborative tagging for text classification: From text classification to opinion mining. Informatics **1**(1), 32–51 (2014)
6. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014). pp. 1724–1734. Doha, Qatar (October 2014)
7. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of Gated Recurrent Neural Networks on sequence modeling. In: NIPS 2014 Deep Learning and Representation Learning Workshop. Montreal, Canada (December 2014)
8. Collin, O., Guerraz, A., Hiou, Y., Voisine, N.: Participation de orange labs à deft 2013. Actes du neuvième DÉfi Fouille de Textes pp. 67–79 (2013)
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (ACL/HLT 2019). pp. 4171–4186. Minneapolis, Minnesota (June 2019)
10. Grouin, C., Paroubek, P., Zweigenbaum, P.: DEFT2013 se met à table: présentation du défi et résultats. In: Actes de DEFT. TALN, Les Sables-d'Olonnes, France (21 juin 2013)
11. Hamon, T., Périnet, A., Grabar, N.: Efficacité combinée du flou et de l'exact des recettes de cuisine. Actes du neuvième DÉfi Fouille de Textes p. 18 (2013)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
13. Johnson, J.M., Khoshgoftaar, T.M.: Survey on deep learning with class imbalance. Journal of Big Data **6**(1), 27 (2019)
14. Krawczyk, B.: Learning from imbalanced data: open challenges and future directions. Progress in Artificial Intelligence **5**(4), 221–232 (2016)
15. LeCun, Y., Haffner, P., Bottou, L., Bengio, Y.: Object recognition with gradient-based learning. In: Shape, contour and grouping in computer vision, pp. 319–345. Springer (1999)
16. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: Proceedings of the 3rd International Conference on Learning Representations (ICLR 2019). New Orleans, Louisiana, USA (May 2019)
17. Martin, L., Muller, B., Suárez, P.J.O., Dupont, Y., Romary, L., de la Clergerie, É.V., Seddah, D., Sagot, B.: Camembert: A tasty french language model. arXiv preprint arXiv:1911.03894 (2019)
18. Oh, J.H., Hong, J.Y., Baek, J.G.: Oversampling method using outlier detectable generative adversarial network. Expert Systems with Applications **133**, 1–8 (2019)
19. Wang, J., Wang, Z., Zhang, D., Yan, J.: Combining knowledge with deep convolutional neural networks for short text classification. In: International Joint Conference on Artificial Intelligence (IJCAI). pp. 2915–2921 (2017)