DEPARTMENT OF COMPUTER SCIENCE

Zero-Shot Text Classification Using an Independent Bayesian Combination

[redacted]

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Bachelor of Science in the Faculty of Engineering.

Wednesday 3rd May, 2023

Abstract

This project aims to assess the viability of the use of an Independent Bayesian Classifier Combination, IBCC, for zero-shot text classification. My experimental hypothesis is that IBCC will outperform a majority vote combination, as well as be resilient to poor-performing ensemble members. This project describes experiments performed to assess this. Experiments are also run for comparison to other similar methods for language classification. The results reveal that IBCC is a worthwhile method in this field, always outperforming majority vote, while providing useful information on the base classifiers in the combination. IBCC also displays the desired robustness against poor-performing base classifiers. This report provides a thorough analysis of the strengths of the method in comparison to others, culminating in the conclusion that there is a place for IBCC in zero-shot text classification tasks.

Practical contributions for the project include:

- I researched in depth existing uses of combinations for classification, which are laid out in this report.
- I wrote roughly 1000 lines of Python code to implement a majority voting system and utilise an IBCC.
- I designed and implemented several experiments to present comparable results for IBCC.
- I provide an extensive analysis on the strengths and limitations for this method in zero-shot classification.

Declaration

[redacted] Wednesday 3rd May, 2023

Contents

List of Figures

List of Tables

Ethics Statement

This project did not require ethical review, as determined by my supervisor, Dr. Edwin Simpson.

Supporting Technologies

As follows is a summary of resources used to support the project.

- I used the Python `pandas` library to support data usage and transformation.
- I used the Huggingface `transformers` library to access pre-trained natural language processing models for use in my implementation.
- I used the Python `seaborn` library for visualisation of confusion matrices.

Chapter 1

Introduction

This report presents the motivation, design, and results of an exploration into the use of combinations of natural language classifiers to improve performance and reduce training requirements.

Natural Language Processing (NLP) is a relevant and continuously evolving subsection of machine learning. It encapsulates the field which analyses the natural use of language in speech and writing and links it with machines, allowing information about the language to be processed more quickly and on a larger scale. At the time of writing, major developments are continuously made in the field, for example, GPT-4 [25] represents a significant advancement in deep learning, which supports image generation from text prompts. The basis of this relies on the computer's ability to understand the text given to it in order to generate meaningful images and responses. While this project does not have the same scope as GPT-4, it does explore the link between the machine and input texts.

This project aims to create a classifier that uses a combination of smaller base classifiers to obtain a better performance when deciding the topic of input texts. This is justified by the notion of the wisdom of the crowd - a phenomenon where the collective opinion of many sources often outperforms an expert in the field, even without the guarantee that the individual sources are in any way accurate. In short, this is because the individuals' errors tend not to be strongly linked, effectively causing their errors to cancel out and meet at a more accurate answer. This is used daily to gather information in informal surveys, but also used for research purposes in websites such as Zooniverse [5], which uses millions of registered users to supply data on a multitude of tasks, providing accurate data given enough respondents. I want to test the effectiveness of this idea when applied to computers, which do not think in the same way as humans do, and will often be incorrect in similar ways when making mistakes. This philosophy is an important topic to explore in NLP because the equivalent of a single expert is a model trained on a vast amount of pre-labelled data over a long period of time. Pre-labelled data is often hard to come across. It required a lot of data and someone to spend a lot of human hours on labelling it, which is expensive. This requirement can also render the approach completely ineffective. For example, in a scenario that has never happened, there will be no relevant data and new data will arrive in real time. A government might want to gauge the sentiment towards proposed measures to reduce the spread of COVID-19 in order to choose the most effective one. There will not be sufficient data to train a language model to identify opinions towards COVID, but such a decision may need to be made quickly, with a risk to life involved. These factors alone are restrictive, but there is also a climate impact [21] of training such models, which gives more motivation to avoid unnecessary training by using multiple cheaper sources. If the combination can eliminate the need for large amounts of data for training, but still provide a good classifier, the requirement for gathering and labelling vast data and the time commitment associated with it is removed.

Because of this, the project combines zero-shot base classifiers. These are pre-trained models courtesy of Hugging Face [14] which have been externally trained on large data sets of text once. Hugging Face recommend to, for specific tasks, "fine-tune" the model so it can better understand the context of the text input it will receive for the main task. However, to keep the base classifiers as simple as possible, I avoid this in order to take take the zero-shot approach - the model receives zero extra information about the task. This also ensures that the project follows the wisdom of the crowd philosophy - with humans, there is no guarantee that they have access to any extra information pertaining to a specific task.

Remaining on this path, I took inspiration from Schick et al. [27], who use "cloze-questions" - questions with a gap to fill - to generate classifications. This is analogous to humans again, since we learn like this from a very young age. Think about what word goes in the gap in the sentence "The cat ___ on the mat." Figuring out that the word is likely to be "sat" is easy for humans to do, but getting a computer to think in this way without all of the context we have from our youth is a lot more challenging. The Hugging Face library has models designed for this task, entitled "mask-filling" models. These fill a word that has been masked over, in the same way a word has been changed for ___ in the previous example. Subsequently, a sentence with a gap can be tagged onto the end of the data to be classified for the model to fill. This will be called a prompt. For example, "The cat sat on the mat. This sentence is about ___". This is somewhat similar to how humans would decide what the sentence is about. You can imagine someone presenting you the previous sentence and asking you "What is this sentence about?", to which you might reply, if you were required to use a full, standalone sentence, "This sentence is about cats". Asking the model this question results in one base classification, but similar classifications can be created by altering the prompt, for example "The cat sat on the mat. The animal here is ___." The downside to this is that these extra prompts need to be engineered, which also takes time.

With lower-level models available, I explore the effectiveness of an Independent Bayesian Classifier Combination (IBCC) on them, to provide one singular overall classification. Bayesian combinations for classification were introduced by Kim and Ghahramani [18] and work by trying to capture each individual's accuracy when assigning each class. For example, someone might be able to correctly identify a dog 100% of the time, but when given a cat they will also say it is a dog 50% of the time. This model takes this into account and realises that whenever the person selects "cat" as the answer, the thing they are labelling cannot be a dog, as they will always have predicted that as a dog. Under the hood, the combination is learning Bayesian probabilities of each class based on the prediction the individual classifier gives. This means that it can ignore an answer if it knows the classifier will always be wrong, or, on the contrary, prioritise one if it is extremely accurate. This alleviates some of the importance of engineering well-performing prompts, as the combination will be able to determine which prompts do not perform as well and down-weight their predictions.

To contrast this with other methods, I also implemented a majority vote combination, which takes the modal prediction from the models and cannot account for any that are always wrong or always right. Furthermore, I have a benchmark from Schick et al. [27], as they created a few-shot classifier using the mask-filling method - while still using a combination, they created a classifier that is fine-tuned on only a small amount of task-specific data, corresponding to the prompts which require words to be filled. This is an alternative method of reducing the time and data constraints, and it will be useful to compare the effectiveness of a small amount of tuning against a combination of zero-shot answers.

The high-level objective of this project is to investigate whether an IBCC can be used with zero-shot classifiers to provide a principled method for training the ensemble of classifiers and test whether it can perform comparatively with other more computationally intensive methods. More specifically, the concrete aims are:

1. Gain a solid understanding of previous literature on classification in NLP.
2. Gain a technical understanding of how a Bayesian combination works.
3. Design relevant cloze-question prompts and implement a transformation of data points accordingly.
4. Evaluate pre-trained mask-filling models when classifying prompts into categories in a zero-shot setting.
5. Implement a majority voting system and use an IBCC to combine base classifications and compare performances between them.
6. Present results on the effectiveness and resilience of IBCC to show that this approach is worthwhile.

Chapter 2

Background

This chapter will give a background into why NLP is a significant topic and some information about previous research into classification tasks in NLP. It will also discuss some available tools that suit the problem, and a few of their flaws, completing Objective 1. This chapter will also discuss the solution which I will run experiments on to test whether it can offer a valuable alternative method.

2.1 NLP

Natural language processing encapsulates the field in which research is done to work towards computers' understanding of natural language in how we use it in daily life. This is an important field of research because lots of key information in a variety of fields can be gathered from the information contained within words. For example, with a sufficient language model, general opinions towards a certain topic can be gathered in real-time from social media posts, giving a unique advantage to anyone trying to tackle such a topic in the media, or someone trying to design a solution to a problem. Another example of a use for natural language processing is the automated summary of articles or papers. A language model will be able to "read" a complete academic article much faster than a human, and oftentimes a summary is all that is required in order to identify a paper as useful or not. This can save a lot of human time. These are just a few examples of the uses of this field. However, as a somewhat new area of research, there are still many obstacles to overcome. The subfield this report focuses on is classification. Given an input text, the aim of classification is to decide the topic in which it falls. Usually, the topics are provided, and the computer's job is to designate a topic for each piece of text it is given. However, due to the nature of classification tasks, in order to train a language model to be accurate, it needs to learn from correct answers. This is the heart of the limitation of this model, as obtaining correct answers is extremely challenging. Primarily, designating classes to excerpts of text requires human input and time. Current language models require immense data for training. The most popular sentiment analysis model in the Hugging Face library of language models, which will be explored further later, is a model entitled "Twitter-roBERTa-base for Sentiment Analysis" [6]. This is trained on roughly 58 million tweets, in order to learn the structure of tweets. It is then fine-tuned further to enable the detection of sentiment using a separate benchmark specified on classification, which defines seven different evaluation metrics. These include whether a tweet is hateful or not hateful or whether it is ironic or not ironic, which requires further labelled data. In total, the labelled data in the training set is almost 200,000 tweets. Creating similar data sets for other tasks is nearly impossible to do practically, which constrains the classification task. Additionally, it cannot be guaranteed that the classifications provided are correct. Human error is unavoidable, and in large data sets human error is almost certain to play a part.

Classification models are also used for sentiment analysis. A sentiment analysis or stance detection model [19] will be able to decide whether input text is in support of or against a given topic, where the classes it can decide between are positive and negative. An example where a classification model is useful is for analysing multiple views for a location or product. The classifier could identify what part of the location the review refers to, for example, the restaurant or play park at a theme park, and a sentiment analysis model could determine whether the review was positive or negative. However, the vast data required for a model to understand each niche and context surrounding reviews is impractical. My approach aims to assess whether a combination can be used to relax this.

2.2 Ensemble Models

When classifying data, there are many different methods to get a classification for a data point. It is common that an improvement in the performance of these methods can be obtained by combining multiple models and taking a prediction which utilises them all. The combination of these is called an ensemble model. This is motivated by a phenomenon called "the wisdom of the crowd". It is credited to have first been written about by Aristotle [24] and encapsulates the phenomenon that the average guess of a crowd of people will be accurate to the true value, even if no one in the crowd is an expert on the topic and some guesses may be wildly inaccurate. A famous example of this is given by Francis Galton [17] in 1906. He conducted an experiment asking attendees of a fair to estimate the weight of an ox. He noted that the median guess was accurate to the true weight of the ox, to within 1%. This concept can be applied in machine learning and NLP, where the crowd can be thought of as many individual language models that might not be experts in the task that they are provided, contrasting with models that are trained extensively for the same task. Combining them can leverage their knowledge and average their answers to supply a more accurate response. Like Galton took the median of the answers, there are lots of different ways to combine base models in machine learning. Examples of the use of ensembles in the context of machine learning can be found in Pattern Recognition and Machine Learning by Bishop [7], and prove to be a powerful tool in boosting the performance of individual machine learning methods. I will use this concept to create an ensemble of NLP models who are not experienced in or trained for the specific task they will be given, and subsequently experiment with methods to combine them to get a more accurate classification for any given piece of text.

2.3 SemEval 2016

In order to understand previous works and performance with regard to language classification tasks, my research led towards SemEval workshops. SemEval is a semantic evaluation international workshop run every year, where computing tasks centred on semantic evaluation and NLP are set to be solved. A year of interest for us is SemEval-2016 and its sixth task [23]. This task was entitled "an ensemble model for stance detection in Twitter". This task was split into two: task A requested a fully supervised classifier, using 70% of tweets as training and the rest as testing, to detect whether a tweet was in favour of, or against, a given target topic. Task B requested a classifier trained on one set of topics to classify the stance of a new set of tweets on a different, untrained topic. The performance of classifiers is measured by the F-score, which is a measure of accuracy on binary classification, given by Equation 2.1.

$$F = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{2.1}$$

Where TP is the number of true positives, where the true class is 1 and the prediction is 1, FP is the number of false positives, where the true class is 0 but the prediction is 1, and FN is the number of false negatives, where the true class is 1 but the prediction is 0. The score is larger when the classifier performs better and lower when the classifier performs worse. The submissions are scored on the average F-score, which takes the mean of the F-scores when class 0 is both "favour" and "against".

The best classification score for task A was using a supervised method, a support vector machine, which obtained an average F-score of 67.82%, and for task B the highest score was 56.28%. There is a stark drop in performance for the topic which has no training data, which indicates that it is significantly harder to infer the stance towards a target which has not been trained upon. Some baseline classifiers were provided to benchmark the tasks. The better-performing benchmark SVMs were done using word and character n-grams, which consider sets of up to three consecutive words or up to five consecutive characters. This supervised benchmark obtained an overall F-score of 68.98%. This score is good, but it necessarily requires training data for each of the topics. Another benchmark provided is a majority class classifier, which takes the most common answer and guesses it every time. They note that this performs surprisingly well, with an F-score of 65.22%, but this is because the training set has a large majority of tweets labelled as "against" for a lot of the target classes, so always guessing "against" will provide a good score. For task B, the classifier is required to predict on an untrained topic. This is a semi-supervised task because the classifier is still trained on the other topics in the context of the task, so it may know some general features that pertain to the "favour" and "against" class, but it has no extra information regarding the specific topic. Fewer submissions were entered for this task. The submission which received the best F-score of 56.28% trained a deep convolutional neural network after defining a rule to annotate the domain in order for it to differentiate between the two classes. This result is notable for this project because the supervised task requires lots of labelled data, something I am trying to avoid, while also not providing any extremely outstanding results. This further illustrates the drawbacks of highly supervised learning with regard to larger label requirements which yield little improvement. Furthermore, the semi-supervised task still requires initial labelled data, but requires deep learning and lots of computational time to get the best average score for differentiating each class. This motivates the exploration of how ensemble models can fit into this space of NLP further, because it illustrates the level of performance supervised methods can give while providing a success criterion to determine whether an ensemble route is worthwhile.

2.4 Schick et al.

Few-shot classification has the same motivations as zero-shot learning. In Few-shot learning, there is some data already available, but only a small amount. The improvement a few data points can supply when classifying text of a very specific task can be large. Usually, this approach is used where some pre-labelled data is available, or it will not take very much time to generate a few labels. This progresses to training an NLP classifier on this small amount of labelled data. A "cloze phrase" is one which has had words removed, needing to be replaced, for example, "Paris is the ___ of France". Schick et al. [27] adapt this to a few-shot setting, giving a pre-trained classifier extra information in the form of a task description, and use Pattern-Exploiting Training (PET), a semi-supervised NLP training procedure to learn about the required cloze-style ph*ases to fill in. Firstly, for each pattern, they fine-tune a pre-trained language model on a small training set. Secondly, the ensemble of all models is used to annotate a large unlabelled data set with soft labels. This makes up a larger training set, which should have some more correct labels to learn from. Once this larger data set is acquired, a standard classifier is trained on it. This reformats the problem from deciding a class label, for example "positive" or "negative", which has no inherent meaning to the model without context, to choosing which of a set of labels, for example "yes" or "no", is the most likely choice to fill the gap in the prompt sentence. They further suggest an iterative version, iPET, which trains many generations of models based on previous generations' soft-label outputs. This means that the final data set to train on is much larger and contains less mislabelled data than a data set produced by one iteration.

In order to utilise their ensemble of models, they fine-tune each separate language model for each pattern they use. This means that each member of their ensemble will be specialised at answering one question. For example, one model is fine-tuned on a restaurant reviews data set, with the prompt as "It was ___. `[review]`", where `[review]` is the text given in the data set. It is fine-tuned with only the answers it can give, which are five one-word gaps ranging from "terrible" to "great". They fine-tune like this because their few-shot training set is small, so the fine-tuning remains relatively cheap. Their aim is that the ensemble of models, who are individually good at answering one question each, can together provide an accurate classification. Their ensemble is combined by giving each model in the ensemble a weighting, which can be uniform or otherwise, and transforming their weighted scores into a probability distribution across each class. The probabilistic approach is similar to IBCC, which is the combination function I will experiment on.

The method of mask-filling is interesting to me because it mirrors the thought process of a human answering a question. When required to classify something, it can be thought that we fill in a gap in a pre-determined sentence. For example, when asked what a piece of literature is about, we might think to fill in the gap in the sentence "This literature is about ___." It seems clunky - in reality, we will not answer in a robotic full sentence like this, but the thought process may still be there. Further, it is not out of the realm of possibility that the word we fill the gap with will be contained in the literature we are classifying, or heavily implied by it. Therefore we are using the knowledge we already have about the language we speak to fill in this gap. I will not assert that this is exactly how humans classify things all of the time. However, it is not far-fetched to imagine yourself employing this when answering questions. It will be interesting to see if a language model can be used in a similar way to mimic human thought.

An approach which also requires an understanding of patterns has been applied in a similar way by Puri and Catanzaro [26], but they use a question-answering methodology instead of mask filling for their unsupervised classification. Question-answering models have a similar specialisation, where they are given some text to read along with a question to answer. It is possible to adapt these models to classification too, by only allowing them to answer in a particular way. Their best model achieves an average of 62.9% accuracy on the AG News data set which I will be testing on.

Overall, I aim to adapt Schick et al.'s work to a zero-shot scenario. Their work on mask-filling for classification is interesting to me, so I will build off this method. Moreover, mask-filling is more adaptable to other scenarios, since sometimes instead of classifying into an objective category, it might be necessary to categorise opinions. Instead of engineering new questions and the options for answers to give in a question-answering model, I can simply change the context of the answer sentence with the mask to fill and the one-word answer options, which represents less hands-on work for the user. Furthermore, Schick et al. use an ensemble in their few-shot scenario, something I aim to do as well, however I will not be specialising each model to answer each question, instead getting the same model to answer all of them.

2.5 Hugging Face

Hugging Face [14] is a community and data science platform which develops open-source tools for machine learning. Their Transformers library [16] is of particular interest, as it is built to facilitate easy access to many pre-trained models that have been developed in previously completed novel research. Generally, transformer models are trained as self-supervised language models. They are trained on raw text, without the need for humans to label them. This library supports both PyTorch and TensorFlow and allows anyone to upload a model they have trained or a data set they have created. These can subsequently be downloaded and used by anyone. They have models for computer vision, audio, and many other tasks, but the models of most interest are their natural language processing models and data sets. All of their transformer models are provided in their Transformers library, and its use will be discussed later.

2.5.1 Models

The models Hugging Face provide pre-trained models which aim to succeed using a process called transfer learning. This is where a model is fine-tuned for the specific task it will be used for. Pre-training is where a model is trained from scratch. The weights for each parameter in the model are initialised randomly, and the model is provided a very large corpus of words and a lot of time to train and understand the language it is being trained on. This is time-consuming, expensive and can have a negative environmental impact [21]. Therefore it is only done once per model. Fine-tuning is training performed after the model has been pre-trained. This requires a task-specific, labelled data set. However, since the model already has some knowledge about language, it requires less data to be able to learn about the specific task. However, since labelled data can be rare ad I am attempting a zero-shot scenario, I will not be fine-tuning models any further. Because pre-training the model has already been done, it will still be possible to leverage individual models' general knowledge about language to answer the prompts I give, while saving the requirement to train for each task. While this may make an IBCC perform better on each individual task, it will require more computational time to transfer the methodology to another task, which contradicts the purpose of the experiment on the ensemble.

There are three subsections of transformer language models, encoder models, decoder models, and models that are a combination of both, sequence-to-sequence models. Encoder models are good for tasks that pertain to the input, decoder models are good for text generation, and sequence-to-sequence models are used when both are required. Since I am performing classification, which requires knowledge of the input, I will be using an encoder model. Encoder models are best when predicting words with surrounding context. Bidirectional information is crucial in a masking model - it needs to know what goes on either side of the masked word, and this is exactly what an encoder model gets. It can always see all words in the input sentence, so will be best at using the whole input to decide which word fits in the gap, and therefore which class the text will fall into.

I must acknowledge the Hugging Face introduction course [15] for bringing me up to speed on the basics of the Transformers library, and the underlying mechanisms of language models.

2.5.2 Mask-Filling Models

Since the method I want to take for classification is mask-filling, it is useful to have a model that is pre-trained to do this. Hugging Face has a section on this, so I will go there for my base classifiers.

One named model of interest to us is "BERT-base-uncased" [9]. This model is used for mask filling. When given a prompt such as "I got the job! This is a [MASK] thing.", where [MASK] is a word to be predicted, this model produces outputs of "good", "new" and "big". This is the premise of using prompts to classify statements; I will add a prompt to the end of the string to be classified, such as the one above, and let the language model fill the gap. The aim is that the model will be able to transfer the knowledge it has gained on filling gaps in sentences to do the same for input sentences it has not seen before. It may also be interesting to experiment with which prompts give better results. Eventually, the aim is to perform classification with many prompts, and weight prompts accordingly.

"BERT-base-uncased" is an uncased model, meaning it does not take capital letters into account, and converts all input into lowercase when it receives it. This serves the purpose of reducing the size of its vocabulary - the words it has access to - to around 30,000, which in turn reduces the amount it needs to know about the language to be functional. This model was trained on BookCorpus [2], which is a data set that contains over ten thousand unpublished books, and English Wikipedia [3]. It is trained with two objectives - masked language modelling and next-sentence prediction. When training on masked language modelling, the model masks 15% of the words in a sentence randomly, then has to predict what the masked words are. Next sentence prediction is where it takes two masked sentences as input and has to predict whether these sentences were next to each other in the original text. This means that when it is trained, it is simultaneously scored on its ability to do both of these tasks and will aim to perform well at both equally. I am interested in a model's skill in masked language modelling, so choosing this model makes sense to take advantage of its pre-training towards this. Furthermore, BERT-base has 110M parameters. In the context of NLP, this means it is not too large, so it will not take too long to complete classifications, but also not too small so it can perform well against its competitors.

2.5.3 Zero-Shot Pipeline

Hugging Face also provide what they call "pipelines". Given a desired model, these automatically align it with the required pre-processing and post-processing steps. These steps turn the input of words into things the model can understand, usually tokens. Tokens are simply a mapping of every word and punctuation in the vocabulary to a number that the underlying model can use. Then once the model has been run on the input, the post-processing turns it back into output that the user can use.

In their models library, Hugging Face provide a pipeline for zero-shot classification. This is a model which does not require any fine-tuning, in the spirit of zero-shot scenarios, but is a great tool to compare other methods to. Given a sentence and a list of candidate labels, it outputs predicted probabilities of each label being the class of the input. It is flexible because any set of candidate labels can be used, so it can determine between "positive" and "negative" sentences with an appropriate list, or categorise text from a list of given categories. This model does not use a mask fill; it is built and pre-trained for categorisation only, so on any input it will only output a classification. It is zero-shot, so it reads the words in the input and outputs the category it thinks fits that sequence of words best. Since it only takes the sentence to be classified as input, it requires less pre-processing and engineering but requires more pre-training for generalist knowledge.

The most popular zero-shot model in the Hugging Face library is called "facebook/BART-large-MNLI", found at [1]. This is an extension of the "BART-large" model [20], after being trained on the Multi-Genre Natural Language Inference (MNLI) data set. The BART model is a sequence-to-sequence model, whose encoder is bidirectional, like BERT's. The MNLI data set is a corpus of over 400,000 sentence pairs, annotated with whether the sentences entail or contradict each other. The data set is crowdsourced, so has required a lot of time resources in the past. Yin et al. [28] proposed "BART-large-MNLI" by using "BART" as a ready-made zero-shot sequence classifier. This means it uses the sentence to be classified in the context of the MNLI data set - the sentence will be paired with a hypothesis sentence, for example, "This text is about politics". The classification is decided by whether the model assigns higher probabilities for entailment or contradiction for the pair of target sentence and hypothesis sentence.

It will be useful to compare the results of an ensemble combination to a single model whose aim is the same as mine, but works differently. Once results are collected, it will also be constructive to compare the running times and parameters in each model. It is clear that more computational time has been put into this pipeline model, so I will be able to assess the effectiveness of this extra work against my own method of combination, which is explained in the following section.

2.6 IBCC

The combination function I will be testing the effectiveness of is an Independent Bayesian Classifier Combination (IBCC). In order to complete Objective 2, this section will provide a technical background explaining how the combination works. This combination used for general machine learning tasks was proposed by Kim and Ghahramani [18], which provides an extension to the model introduced by Dawid and Skene [8] in 1979.

This model requires the assumption that the base classifiers are conditionally independent. This is to ensure that when the classifiers make errors they are not correlated, so the average of their errors will be closer to the true value required to predict. In reality, this will rarely be the case. However, I will continue to make this assumption despite the base classifiers being trained on the same word corpus. The limitations that this requirement causes will be discussed later.

2.6.1 Probabilistic Method

When completing a classification task, there is a set of data points x_i indexed from 1 to N, where we assume that the ith data point has true label t_i, which is contained in a set of class labels $\{1, \ldots, J\}$. It is assumed there are a finite number of labels in a classification task, and the labels t_i are generated from a Multinomial distribution, where the vector of probabilities for each class is $\mathbf{p} : p(t_i = j|\mathbf{p}) = p_j$. Then with K base classifiers, the output of classifier k for data point i is $c_i^{(k)} = l$, where $l \in \{1, \ldots, J\}$, which is also assumed to be generated from a Multinomial distribution which is influenced by the true label, controlled by parameters $\pi_j^{(k)}$: $p(c_i^{(k)} = l|t_i = j, \pi_j^{(k)}) = \pi_{jl}^{(k)}$. The entire matrix $\pi^{(k)}$ is called the "confusion matrix" for the kth classifier. In words, it contains the probabilities that the output of the kth classifier is each possible class label, given that the true class for the same data point is each possible class label.

The assumption of independence conditioned on the true label t_i is key to progress further. This means that the joint probability of classifier prediction c_i with true class t_i is given by Equation 2.2.

$$p(c_i, t_i|\mathbf{p}, \boldsymbol{\pi}) = p_{t_i} \prod_{k=1}^{K} \pi_{t_i, c_i^{(k)}}^{(k)} \tag{2.2}$$

Then, with the assumption that the class labels across data points are independent and identically distributed, the total likelihood across all labels is derived from Equation 2.2 by multiplying over all possible data points to give Equation 2.3.

$$p(\mathbf{c}, \mathbf{t}|\mathbf{p}, \boldsymbol{\pi}) = \prod_{i=1}^{N} \left\{ p_{t_i} \prod_{k=1}^{K} \pi_{t_i, c_i^{(k)}}^{(k)} \right\} \tag{2.3}$$

With the closed form equation for the likelihood of each prediction and true label combination, it is possible to perform maximum likelihood estimation (MLE) to pick the values for $\mathbf{p}$ and $\boldsymbol{\pi}$ which maximise the probability of our observations. However, since both parameters need optimising, it follows to use a Bayesian approach.

2.6.2 Bayesian Model

In machine learning, the value of $c_i^{(k)}$ is known, because it has been given by the base classifier, or by a human in a crowdsourcing setting. The value of t_i is unknown and is the target to be predicted. This leaves only $\mathbf{p}$ and $\boldsymbol{\pi}$ to estimate. Using a Bayesian EM algorithm, these values can be learned iteratively. This means that one of these values is optimised with the other fixed, and then vice versa, until convergence. This will produce learned versions of the confusion matrices $\pi^{(k)}$, which provide an interpretable result that can quantify the performance of each individual classifier, which will allow the model to ignore a poor-performing classifier. A learned version of $\mathbf{p}$ is an estimate of the proportion of each true label there are in the data set, With a Bayesian approach, the parameters to learn need a prior distribution. Previous works from Kim and Ghahramani [18] and the implementation used by Simpson et al. [11] model the priors for a single row of the confusion matrix with a Dirichlet distribution with hyperparameters $\boldsymbol{\alpha}_j^{(k)}$, which in turn have an exponential prior distribution with parameters $\lambda_{j,l}$. This exponential distribution is not used in the implementation used for this report, because for the purposes of the experiment, simpler prior distributions can be provided to avoid extra complex calculations. The $\boldsymbol{\alpha}$s are only used to optimise the convergence of the confusion matrix to correct values. A detailed explanation of these expressions is not required here, because we expect the conditional probabilities in the confusion matrix to converge to similar values with any reasonable prior, for example, a uniformly distributed prior. It is a similar story for the prior of $\mathbf{p}$, which are also initialised with Dirichlet hyperparameters $\boldsymbol{\nu}$, but it is also expected that this class proportion vector eventually converges correctly from any reasonable prior as well.

A graphical diagram of the model is provided by Simpson et al. [11] in Figure 2.1. The shaded node represents the only observed values and the two parent nodes of it are the learned values that are of interest for this task.

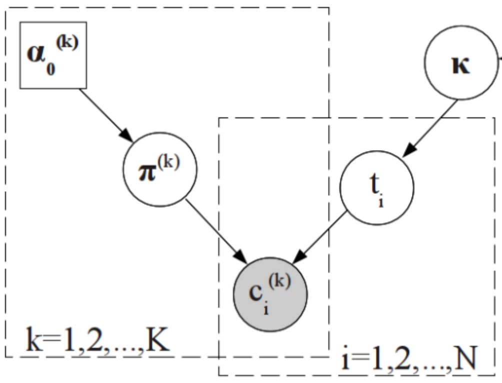

Figure 2.1: Graphical Diagram of IBCC from Fig 1.1 of Simpson et al. [11]

The hypothesis to test regarding this method is that it will perform better than a majority vote combination function. While this function does take into account all of the predicted labels, it can be hindered by base classifiers that perform worse. It is therefore expected that this form of combination would perform better than an individual base classifier, but not as good as an IBCC, which should be able to identify any worse classifiers and down-weight them accordingly. This form of selection is important because it can reduce computational time further down the road. If a certain base classifier is deemed redundant by an IBCC, there is no requirement for any more predictions from the model. Additionally, if the IBCC identifies a classifier is weaker at a certain class, it will be easier to give it further training pertaining to that specific class, to increase its performance with more refined data.

2.6.3 Assumption Analysis

The two assumptions made when using IBCC are as follows.

1. The base classifiers are independent of each other given a true label t_i.
2. The true labels for each data point are independent and identically distributed.

The first assumption is quite a strong one. It requires each classifier to be completely independent of all other classifiers when predicting a value for a data point with true class t_i. In applied scenarios, this is rarely the case. Since the base classifier models are trained on the same corpus, their mistakes are likely to be at least somewhat correlated. With the desire to reduce the requirement for specified models, this is a sacrifice that comes with ensemble models, because it is not within the philosophy to focus on providing data for training which will ensure independent predictions. In reality, while classifiers will not make independent errors, they can be encouraged to make less correlated errors by providing sufficiently different prompts for each classifier to answer. If it can be engineered so that there is enough independence that the ensemble can still provide successful classifications, this method can still be deemed successful, despite the assumption drawbacks.

Fortunately, the second assumption is less significant. There is generally much less dependence between the true labels for two data points in the same class. It is certain that the data points already in a class will have a small effect on new data points being classified, but it is expected that the majority of the reasoning behind an objective classification regards only the data point in question and the class label it will be assigned. This assumption of an independent and identically distributed data set is standard and will be continued throughout.

2.7 AG News Data Set

The data set I have chosen to use for my experiments is the AG News data set. This data set is included in the Hugging Face data set library [13]. AG is a collection of over 1 million news articles, from over 2000 sources. The data set specified for topic classification was constructed by Zhang et al. in [29]. Their data set contains over 127 thousand entries, and is split into a training set with 120 thousand rows, and a test set of 7600 rows. Since the aim is zero-shot classification, I will be using only the test set, to align with other work. The test set contains an equal distribution of the true classes, so a model needs to be balanced in all of them to perform well. It is important to note that this data set is only used to test the effectiveness of IBCC. The aim of the task is not to classify news articles extremely well, but to test the viability of IBCC for zero-shot classification in general, including when applied to other topics. This further justifies not fine-tuning the model to be good at news classification.

Each data point in this data set consists of a pair of headline and body text corresponding to an individual news article, along with an integer category label according to which category the article was posted to. There are four categories: 1: “world”, 2: “sports”, 3: “business” and 4: “sci/tech”. These four are the categories with most articles in them in the original news corpus, so were chosen to form the classes in the classification data set. Overall, the data set has three columns: “Class Label”, “Title” and “Description”, which I will refer to as “body” from now on, for clarity.

Since the news articles in this data set are from so many different news sources, it can be drawn into question whether the categories they have been given are objective. Since different people will have labelled the articles at different times, it is possible that if an article had been labelled by a different person it may be in a different category. However, I am confident that this will have a minimal effect on any results, because the four categories given are quite distinct, and therefore errors made in classification will be uncommon and will not affect the results significantly. Therefore I am happy to go ahead with using this data set to test my method.

Chapter 3

Project Execution

This chapter will describe the creation of individual zero-shot classifiers using language models, and then move to describe the path of experiments I took when designing more prompts for new classifiers. The chapter will include results for each successive experiment and use these to justify the next steps in the process. I will also analyse what my IBCC model has learned about each individual classifier in order to improve the ensemble's performance, culminating in an evaluation of the method's performance.

3.1 Choice of Software

As discussed in Section 2.5, Hugging Face is a tool to access pre-trained NLP models developed in previous research, it is an obvious choice to utilise. I will use their data sets data base to download the AG News data set from Section 2.7. Further, I will use their transformers library to perform automatic encoding and decoding of inputs to give to the model.

The Hugging Face transformers library [16] is designed for Python, so this is the language I choose to work in. This plays into my familiarity with the language for machine learning tasks. My familiarity extends further to useful Python libraries, pandas [22] and NumPy [10]. The pandas library will be useful to load and organise the data sets I will be using, and NumPy arrays will be useful when combining classifiers by storing predicted labels. I will also use the seaborn library [12] for visualisation of confusion matrices.

I started on the AG News data set [29]. This is so I could have a benchmark for comparison with previous work. The labels for the categories given for news articles are 1: "business", 2: "sports", 3: "world" and 4: "sci/tech". However, "sci/tech" is not a word in the vocabulary of "BERT-base-cased". Therefore, to combat this, I split "sci/tech" into two words: "science" and "technology", and declared a classification correct if the model chose either of these categories when the true label was "sci/tech". It is intuitive that this is what the category should contain, therefore if the model chooses either of these two categories, the label it has decided upon is class 4.

3.2 Single Prompts

The most computationally heavy part of this classifier combination task is generating the annotations of the language models for each prompt. Thus I will do this first and only once. This requires the design of sensible prompts to begin with. Even with the consideration that my eventual combination should be able to identify poor-quality prompts, it is still important initially to get an estimate for how well "good" prompts perform individually.

From here onward, `[title]` will refer to the text in the "Title" column of the AG News data set, `[body]` will refer to the text in the "Description" column of the data set, and [MASK] will continue to represent the mask token that the model attempts to fill.

For the initial experiment, I designed the following prompt: "`[title]. [body].` This article is in the [MASK] category." and use it on its own. This prompt follows the cloze-phrase pattern discussed in Section 2.4. I designed it this way in order to mimic a human's response well, as well as to play to the strengths of the model, by providing context on both sides of the mask token. For each headline-body pair in the data set, the base language model will predict which word from "business", "sports", "world", "science" and "technology" has the highest probability of fitting in the [MASK] space. This generates a list of $n = 7600$ classifications, each taking a value from 1-4. To determine whether the classification is correct, the "Class Index" column of the AG News data, which also contains a value from 1-4, is compared to the prediction. The proportion of correct predictions across the whole data set gives its accuracy. Since it is not a binary task, F-score is an inappropriate measure, and I am particularly interested in positive performance so vanilla accuracy is a sufficient measure for this.

In total, 58.8% were correctly classified. Since there are four categories, a random assignment would expect to be 25% accurate. Therefore, even individually, this prompt scores surprisingly well - despite not being tuned on news articles or on answering questions in this format, the model still predicts the majority of data points correctly. This suggests that the method of mask-filling for classification might have some successes. This is promising and motivates the following.

3.2.1 Creating Prompts

The natural next step is to combine multiple prompts, using the principle that an ensemble will be more accurate than an individual. The new set of prompts is listed in Table 3.1. From now on, I will reference the prompts with a prompt ID number given in this table. Prompt 1 is the same as in the first experiment. Prompt 2 is also designed by me. In this one I chose to remove the bidirectional information, to get a scale of how much this affects performance in an encoder model. Prompts 3-5 are the same as the ones proposed in Schick et al. Since they fine-tune on each of these patterns, I wanted to experiment with how these patterns function as classifiers when the models have less knowledge of how to use them. It is noticeable in the prompts that these are much less like human sentences - they take a closer resemblance to how the category label for the article might appear on a website. In particular, Prompt 3 contains no additional context beyond the article itself and a pair of punctuational parentheses. This relies a lot more on the model's understanding of the use of parentheses in this manner, which it may not have without fine-tuning to the pattern. The individual performances of each of these new prompts are also given in this table.

ID	Prompt	Accuracy (%)
1	"`[title]`. `[body]`. This article is in the [MASK] category."	**58.8**
2	"`[title]`. `[body]`. This article is about [MASK]."	58.2
3	"`[title]`. ([MASK]) `[body]`."	52.6
4	"[Category: [MASK]] `[title]`. `[body]`."	57.6
5	"[MASK] News: `[title]`. `[body]`."	58.3

Table 3.1: Prompts

The first prompt still performs the best out of the cohort. The performance of prompt 2, which used a similar sentence is only slightly worse. This may be an indication that my hypothesis that only having context on one side of the mask token reduces performance, however, I am hesitant to assert this with limited data and only a small drop. Prompt 3 is the lowest performing of these, giving an absolute drop of at least 5% when compared to the others. I suspect this more significant drop in performance is indeed due to the lack of context given to the model and its lack of understanding of parentheses. However, it does correctly classify more than half of the prompts. This demonstrates the understanding carried in the base classifiers about language, even despite the model not being excessively large. Prompt 4 performs slightly worse than the first two, but not significantly, and it seems the addition of the context word "category" might help the model understand what it is doing. Another point of note is that Prompt 5 performs on par with Prompt 2, despite only having one extra context word. It is possible the word "News" to contextualise the data point itself is valuable to the model. Since the AG News data set has an even distribution of data points in each class, a baseline accuracy score is comparable to the F-score given in Section 2.3. The prompt accuracies do not outperform the best individual classifier scores in task A of the SemEval workshop [23]. However, interestingly, some of the prompts outperform the F-score of the classifiers for the unseen task B. This is also promising - the base classifiers are quite strong on their own but require less computation, so this gives a strong baseline for any combination function to make up the difference to larger, more trained individual models. These results indicate that the designed prompts are relevant and successful, displaying achievement of Objectives 3 and 4 laid out in Chapter 1.

Since these classifications take the most time to compute, the labels that each model predicts for each data point are stored in a $P \times N$ array, where P is the number of prompts used and N is the number of data points. This is then saved to a .csv file using pandas to be accessed later. This ensures that results stay consistent and the base classifiers do not need to be run more than once. The format of the classifications when stored is a matrix that contains the values in the prompts columns of Table 3.2.

ID	Prompts				
1	3	2	3	3	3
2	4	4	4	2	3
⋮	⋮	⋮	⋮	⋮	⋮
7600	1	4	1	4	1

Table 3.2: Storage of predictions

3.2.2 Resilience Testing

In order to test the effectiveness and resilience of different methods of combining classifiers, I also chose to create some prompts that I expect to perform worse. The motivation behind this is that intuitively it is expected that this will have a negative effect on a majority vote classification, because the defunct predictions will contribute the same as good ones, and I hope that a more sophisticated Bayesian combination will be able to down-weight these accordingly and not lose any performance.

Firstly, prompt 6 still reads the article, but fills in a mask in an unrelated sentence. I designed it this way because I expect this to give a poorer but not bad performance. I have provided the base classifier with no meaningful context about the task itself, but it will still assume that I have done this by classifying the word to the best of its ability. In fact, I chose to use the word "unrelated" to try and leverage the model's base understanding of the word to move the prediction away from being linked to the data point.

Prompt 7 has similar aims to prompt 6 - the sentence is unrelated. However, this time it is clear that the word that should be filled in should be "Paris" or something similar. The context provided to this model is not useful when classifying. Despite a suitable word not being in the candidate labels, this could mean the model gives a predicted label that it deems most similar to such a word. This opens up the possibility for one class to be more common, and therefore more frequently incorrect. Prompt 8 cuts each of the headline and title down to twenty characters. This means that the model will only get to read the first few words in each headline and body of each article. Hence, the classifier will get less context from the data so will be less accurate in its classification, even though the mask token is in a relevant sentence to the task.

In an attempt to get some really poor prompts, I included two different extreme cases for the last two prompts. Prompt 9 does not read the article, and takes the same additional sentence as in the first prompt. Since the input containing the mask token now never changes, the model will make the same prediction every time. This will result in a constant assignment, which will have no use and should be detrimental to data points not in the class it chooses. Additionally, prompt 10 does not use a language classifier and is random. It takes a random number from 1-4 and chooses that as its classification. It is clear this will also not be useful and detrimental to a combination. All of the new prompts and their individual performances are listed in Table 3.3.

ID	Prompt	Accuracy (%)
6	"`[title]`. `[body]`. Unrelated [MASK] sentence."	48.9
7	"`[title]`. `[body]`. [MASK] is the capital of France."	51.9
8	"`[title][:20]`. `[body][:20]`. This article is in the [MASK] category."	49.8
9	"This article is in the [MASK] category."	25.0
10	Random assignment of a number 1-4	25.2

Table 3.3: Poorer prompts

The accuracy for prompts 6-8 is still quite good, classifying around half correctly. This score is not useful in a practical sense, but only dropping an absolute 10 percentage points after being designed to perform poorly is also not a bad result either. Prompts 6 and 7 show that the sentence the mask token is in does not matter as much, as since the article is longer, and the model is bidirectional, it will still get a lot of information from the text it is given. It is likely that the reason prompt 7 scored highest out of these is because the mask token immediately follows the end of the data point, so takes the most influence from it. On the contrary, the context of the sentence in prompt 8 allows the model to get enough context from only twenty characters to still classify correctly half the time. These results were surprising, but they still provided a 10% drop in performance and introduced some extra bias into the classifications. These extra prompts will be useful to test the resilience and bias resistance of different combinations.

Prompts 9 and 10 perform as expected. Prompt 9 ends up choosing category 2, "sports", for all of its labels. It gets a score of 25%, displaying that a quarter of the data points in the test set are in the sports category. The random assignment also gets a score of 25% because all categories have equal representation in the test set. These prompts should have an increased detrimental effect on a majority vote combination, because they provide no useful information about the data point but dilute the answer pool when voting.

3.3 Combining Prompts

With the set of prompts designed and their individual performances assessed, the natural next step is to begin combining them. It is expected that, due to the wisdom of the crowd philosophy, the general accuracy of an ensemble of models should be higher than of the individuals. With multiple predictions from multiple models, the aim of the ensemble of models is to use the predictions from the base classifiers to output one singular classification for the same data point. Therefore, for my application, the ensemble prediction will have an architecture like the one given in Figure 3.1; a list of predictions from each prompt $1, \ldots, P$ will be fed into a combination function, which will result in one prediction for that data point, p.

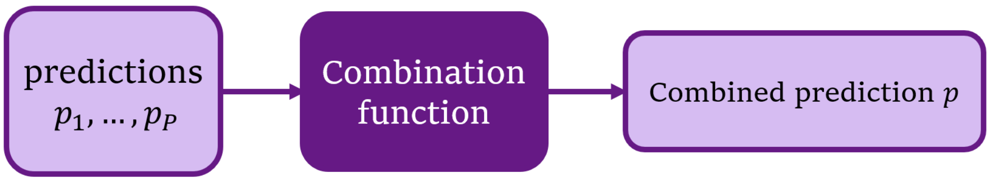

Figure 3.1: Ensemble of model prediction

The simplest method that can be used in the place of "Combination function" is a majority vote function, taking the most common prediction in the input set. This is the combination I will use first.

3.3.1 Majority Vote

This simple method only requires setting up a vote that each base classifier can make to provide the overall ensemble prediction. Since I already have the individual classifiers' predictions saved in a matrix in the form given in Table 3.2, I can load them and access them easily. The algorithm to calculate a majority vote iterates through every row of the matrix and takes the modal value in the row.

Initially, I used a majority vote to combine the classification of a sentence by the first five prompts. It must be noted that this process can now produce a tie for the modal label from the base classifiers. In this case, my classifier chooses the class based on the modal value which appears first in the list of annotations. This will introduce a very slight weighting towards the first few prompts in the given list, only ever influenced in a tie. Therefore, even though this will only provide a small change in performance, it must be noted that by following the natural course of the experiments, the stronger prompts are at the beginning of my list, so will be the predominant tiebreakers. While this slight weighting turns out to be beneficial in this case, generally it will not be the case that stronger classifiers are first. Using this combination resulted in a success rate of 63.7% over the whole test set. This is a promising result, as it represents a gain of 5 percentage points over the best-performing single classifier used. Since it outperforms all of the individual base classifiers, it is clear that the ensemble leverages collective knowledge to provide a more accurate prediction.

Then, to test how combinations are weighted down by poor prompts, I added prompt 6. I had designed this prompt to be worse, but not bad. However, when running the majority vote combination with this prompt in addition to all previous prompts, the ensemble now obtains an accuracy of 64.9%. This is an improvement on the previous ensemble, which demonstrates how it can still be useful to include poor annotations in ensemble predictions. It could be the case that this prompt breaks a few ties to ensure a correct answer.

Following this improvement, I added the remaining four “bad” prompts. The aim of this is not necessarily to get an improved classification, but to test resilience against incorrect individual annotations. If a model is created in a zero-shot manner, it will be more common that individual classifiers will be less accurate, or confidently wrong. Therefore, in a perfect world, even classifications designed to be detrimental would not affect the performance of an ensemble, in order to save the effort of identifying and removing poor-performing prompts while maintaining a good performance. However, majority voting is naturally susceptible to poor prompts, as they have an equal impact on the final classification as good prompts. The results of this experiment prove this susceptibility, as the accuracy drops down to 63.0%. This is a worse score than the ensemble of the first five prompts and displays the risk of adding poor classifications to a majority vote combination. A full results table for majority vote experiments is given in Table 3.4.

Majority vote	Accuracy (%)
Best single prompt (ID 1)	58.8
5 prompts	63.7
6 prompts	**64.9**
10 prompts	63.0

Table 3.4: Majority vote accuracy for different amounts of base prompts

3.3.2 IBCC

With the performance of the majority vote combination function assessed, the project is ready to move onto Objective 5 and assess an IBCC’s performance on the same task. I used the same subsets of prompts for the fairest comparison. I used an IBCC implementation provided by my supervisor for the task.[1] The equivalent IBCC result for the first experiment was an ensemble score of 64.3%. This is 0.7 percentage points better than the majority vote combination on the same prompts. Its accuracy is only slightly better, which does not give IBCC a mandate to be used over majority voting, however, later experiments will focus on resilience to poor prompts, so it cannot be eliminated as a useful method yet. Furthermore, in the process of providing the classification, the IBCC model has learned information about each prompt, which can be learned from later.

The first resilience test for majority voting was adding prompt 6, so the same was performed for the IBCC combination. Adding this single prompt also improved the classifier’s accuracy. This time, however, it improved the accuracy by 4.7 percentage points. This is a significant increase considering only one prompt was added, and it was designed to perform worse. This highlights the strengths of the IBCC model, because it has learned the benefits that the extra prompt provides, while being able to ignore it when it is bad. The confusion matrix that the combination has learned for prompt 6 will be useful to understand why adding this prompt presents a large increase in accuracy. This confusion matrix is learned alongside the confusion matrices for the other 5 prompts, and these matrices affect each other in the learning process. A deeper analysis of confusion matrices will be performed later.

Finally, I performed tests including all ten prompts, which includes the two useless ones. In this test, the score of the majority vote classifier dropped lower than it was for the initial five prompts. However, IBCC yields an accuracy of 72.4% on the test set. This is another improvement from the last result and a vast improvement from any of the individual classifiers involved. Since the classifications for the new prompts affect the estimation of the underlying categorical distribution that the classes take, the model may be taking advantage of having more classifications to learn this from. This estimation is used to generate the confusion matrices for each prompt, which, if learned more accurately, allows the combination to weight each base classifier closer to the truth, providing a better result. On the contrary, the model must also learn that the final two prompts are not useful in determining the distribution of classes, and must use their confusion matrices to down-weight their contribution almost completely. The model strikes a balance between these two to grant this improvement in performance. Overall, despite trying to provide poor prompts, the combination still uses the extra information positively and improves its result, displaying its resilience and robustness. The table of results for the IBCC experiments is given in Table 3.5

This result indicates the strength of IBCC when used for zero-shot classification tasks. Not only does the use of IBCC improve the classification performance when poor base classifiers are added, but it also

[1]Implementation can be found at https://github.com/UKPLab/arxiv2018-bayesian-ensembles

IBCC	Accuracy (%)
Best single prompt (ID 1)	58.8
5 prompts	64.3
6 prompts	69.0
10 prompts	**72.4**

Table 3.5: IBCC accuracy for different amounts of base prompts

provides useful information about these classifiers. A compiled results table for these experiments is given in Table 3.6. This table also includes performances from methods discussed in Sections 2.5.3 and 2.4. The score from Schick et al. [27] is taken from their zero-shot test. It measures how their PET technique performs with no training on the specific task. The zero-shot pipeline [1] was provided the headline and body of the article and asked to predict a category from the ones supplied.

Method	Accuracy (%)
Best single prompt (ID 1)	58.8
Majority vote with 5 prompts	63.7
Majority vote with 6 prompts	**64.9**
Majority vote with 10 prompts	63.0
IBCC with 5 prompts	64.3
IBCC with 6 prompts	69.0
IBCC with 10 varied prompts	**72.4**
Schick et al.	69.5
Zero-shot pipeline	75.3

Table 3.6: Benchmark comparisons with each combination method

Comparing with existing work shows that the IBCC is performing well. It takes the pattern-based method from Schick et al. [27] and gives an increase of 3 percentage points to their zero-shot application, marking an improvement without the requirement for few-shot learning. This meets the goal of removing the necessity of extra data in order to create an increase in performance for classification. This shows how using this IBCC can be used to meaningfully build on existing mask-filling work. The IBCC falls 3 percentage points short of the zero-shot pipeline however; this is partly because the pipeline was designed for topic classification instead of adapting mask-filling for this, but also because the zero-shot pipeline has over 400 million parameters, compared to BERT-base's 100M. This results in an increase in accuracy on the task but also results in more computational time, which will be discussed in the following section.

3.3.3 Time Constraints

It is important to state that there is a balance between computational time and classification accuracy. One key benefit of few- and zero-shot classification is that without needing to train on extra data, there is less overall time taken to get a classification. However, this effect would be nullified by including an unreasonable amount of prompts in the name of accuracy. Therefore I have also recorded how long the combination classification process takes.

A rough idea of the process taken to generate a full ensemble's predictions is given in Algorithm 1. This algorithm displays that no matter how efficiently the language model classifies data points, there is an unavoidable requirement to generate $N \times P$ predictions, where N is the total number of data points and P is the total number of prompts. The amount of data points to classify is uncontrollable, but the number of prompts used to classify can be. Therefore, creating many prompts to combine may increase the accuracy of the ensemble, but will also increase the time it takes to generate base classifications to combine, which defeats the purpose of using an ensemble to reduce computational time.

The zero-shot classification pipeline uses a large model, which means it will take longer to create a single classification. This corresponds to the computation required in line 4 of Algorithm 1. This also plays a big part in the efficiency of the classification, and in order to align with the objective to keep computational requirements lower, I recorded runtimes for generating each classification. Since the predictions from each base classifier are only made once, I recorded the time it took to generate different amounts of prompts separately, in order to create a comparison. This will be useful to balance the accuracy of the ensemble model with the speed at which it generates its predictions. The timings can be

Algorithm 1 Algorithm to generate ensemble predictions

```
1: for $p \in \{1, \dots, P\}$ do
2:     for $i \in \{1, \dots, N\}$ do
3:         Format data point $i$ into prompt $p$
4:         Input this to the language model and compute the prediction
5:     end for
6: end for
```

found in Table 3.7.

Method	Time (hh:mm)
Best single prompt (ID 1)	00 : 23
Generating 5 prompts' predictions	01 : 42
Generating 6 prompts' predictions	02 : 02
Generating 10 prompts' predictions	02 : 41
Zero-shot pipeline	05 : 44

Table 3.7: Results comparison for majority vote combination

The runtimes were not recorded in a controlled setting, and other processes taking place at the same time will have an effect on them. The aim of this table is simply to get a general feeling for efficiency, not a detailed assessment. It is clear from these results that extra parameters in the zero-shot pipeline mean it suffers a large cost to runtime. The pipeline takes one data point and performs a classification on it. Contrast this with the time it takes to generate predictions for all ten prompts from the "BERT-base" model; the pipeline takes approximately twice as long, despite classifying ten times fewer data points. This illustrates the power of using an ensemble of basic models. The time to classify the same amount of data points for the base model, 23 minutes, allows multiple base models to generate predictions with similar computational time, therefore suffering no cost to take an ensemble combination compared to the expert models.

Considering this, the accuracy of 72.4% that the IBCC achieves using ten prompts is an even bigger achievement. Since the pipeline scores 75.3%, the worthwhileness of an increase of 3 percentage points is drawn into question when considering that it could take twice the time to generate the prediction. Generating a combined prediction from the outputs takes less than a second, so this is a large success for the ensemble - it still gives a comparable classification performance, while cutting the need for extra data, extra training, or more parameters. This aligns with Objective 6 of the project laid out in Chapter 1 and clearly displays that the use of an ensemble combined with an IBCC is an excellent tool when classifying in a zero-shot NLP context.

3.3.4 Posterior Probabilities

Since the IBCC learns about the prompts in addition to an ensemble classification label, it will be useful to look at this to gather extra information about how the combination uses each individual prompt. This could lead to adaptations to the prompts later, but now will give some insight into how the underlying system is working.

One of the parameters that the IBCC learns in order to make a classification is the confusion matrix for each prompt, $\pi^{(k)}$. Each entry to this matrix, say $\pi_{ij}^{(k)}$, contains the estimated value of the probability that the prompt will predict class j, given that the true class is class i. Therefore the entry is an estimate of $p(c^{(k)} = j|t = j)$, where $c^{(k)}$ is the prediction that model k gives, and t is the true class. The sum across rows will always be 1. A perfect classifier would have its confusion matrix be $\pi = I_J$, where I_J is the $J \times J$ identity matrix and J is the number of class labels. This is because the diagonal values are all 1, which means that the probability that the model will predict any class is 1 if it is the true class and 0 if it is not. Hence, a prompt that performs well will have a confusion matrix close to the identity matrix.

I have converted the confusion matrices learned by the IBCC into heatmaps using the seaborn Python library [12], to make it easier to see where the larger values are. The confusion matrix for the first prompt is shown in Figure 3.2. Further, a reminder of the categories corresponding to each label is given in Table 3.8, and the prompts corresponding to each classifier are listed in Table 3.9.

Figure 3.2 reveals that when the true class is class 1, "world", the first prompt is actually more likely to predict class 2 than class 1. However, the other diagonal entries are stronger, meaning the combination

Label	Category
1	"world"
2	"sports"
3	"business"
4	"sci/tech"

Table 3.8: Categories and corresponding labels

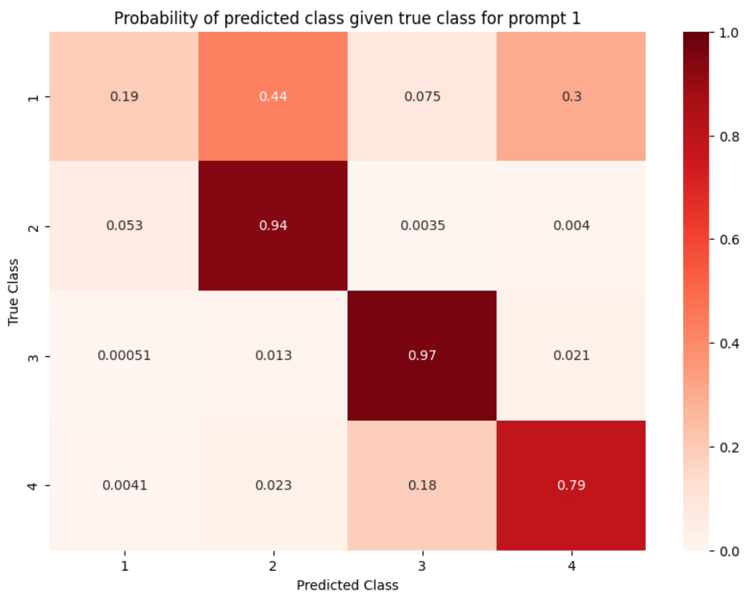

Figure 3.2: Heatmap for the confusion matrix of prompt 1

thinks this prompt tends to make errors in a certain way. The classifier asserts that the prompt is very good at predicting the business class, stating that 97% of the time the true class is business, this prompt will predict it. The IBCC will account for this when using this prompt's classification in its combination.

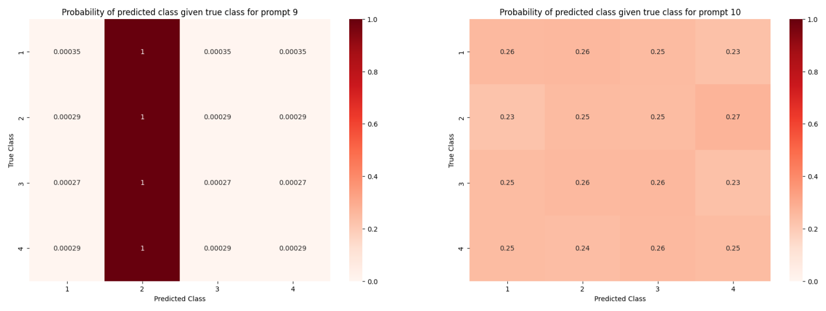

Figure 3.3: Heatmaps for the worst performing prompts

Another two examples which demonstrate the functionality of this combination are the final two, given in Figure 3.3. Recall that prompt 9 was a deterministic prompt, which always predicted the "sports" category. The combination has identified this and has assigned 1 to all values in the second column. This translates to the fact that no matter what the true class is, it will always predict class 2. Therefore, this prompt will be effectively disregarded in the combination - when it sees that this classifier has predicted class 2, it will multiply all the probabilities it already has for each class by 1, leaving them unchanged. Similarly, for prompt 10, which had a random class assignment, the combination has identified that no matter the true class, the classifier has a 25% chance of predicting each class. Therefore when incorporating this into the combination prediction, the contribution this prompt has will scale all classes equally, also not affecting which class the combination chooses.

Considering all of the confusion matrices in parallel to each other is also useful. Every confusion

ID	Prompt	Accuracy (%)
1	"`[title]`. `[body]`. This article is in the [MASK] category."	**58.8**
2	"`[title]`. `[body]`. This article is about [MASK]."	58.2
3	"`[title]`. ([MASK]) `[body]`."	52.6
4	"[Category: [MASK]] `[title]`. `[body]`."	57.6
5	"[MASK] News: `[title]`. `[body]`."	58.3

Table 3.9: Prompts

matrix is included in the appendix. For now, I would like to consider prompt 5 in comparison to all of the others. Prompt 5's mask token is at the beginning of the text to be classified, and it is followed by the word "news", the only one which uses this word. The confusion matrix for this prompt is shown in Figure 3.4.

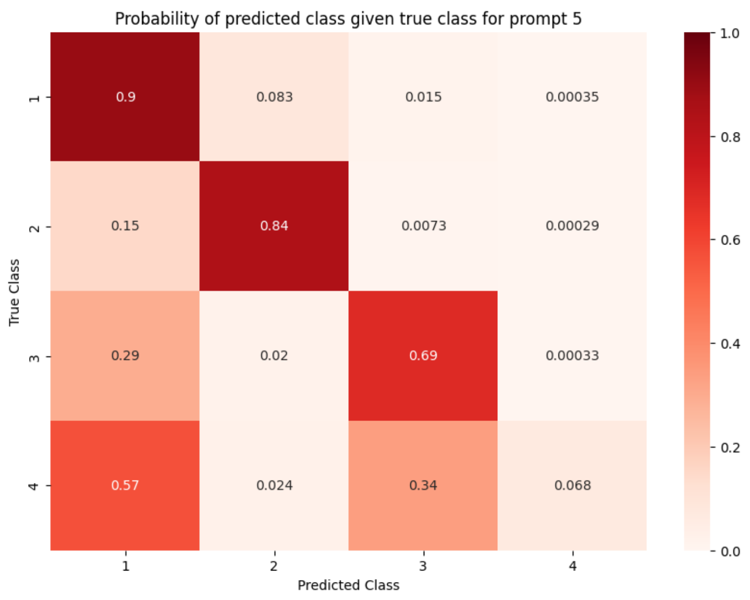

Figure 3.4: Heatmap for the confusion matrix of prompt 5

The other four prompts used in the initial experiment have similar confusion matrices to the one for prompt 1, which was shown in Figure 3.2. That is, most of them did not score well when classifying data points in class 1, "world". However prompt 5 flips this and performs exceptionally well at classifying these data points, but is not very good at classifying text from category 4, "sci/tech". This prompt displays that the use of different words can create noticeably different classifiers. I suspect the reason that the other classifiers do not perform very well on the "world" class is that their understanding of the word "world" differs from how it is used in this context. Since the models are not fine-tuned on news articles, their understanding of "world" will align more with the idea of the entire planet, or the collective of people that live on it. In this scenario, the word "world" is very nuanced, as it is often used to contain articles happening from elsewhere in the world, not about the world as a whole. Therefore a sentence such as "This article is in the world category" requires a lot more knowledge about the English language and its use in the context of news to understand. Since prompt 5 uses the word "news" immediately following it, this prompt is forced to use the context of news articles explicitly. "World news" is used more frequently as a combination of words, and its meaning is much closer to the understanding a model would need to classify this correctly, so this prompt is better at this class. Equally, phrases like "science news" and "technology news" might not be used as commonly by a human speaker, and this is reflected in the performance in category 4.

As mentioned in Section 2.6, a Bayesian combination works better when the classifiers are independent of each other. Since prompt 5 is good at class 1, and poor at class 4, it is different from the other prompts in this ensemble. This means its inclusion improves the IBCC performance by providing balance to the other base classifiers.

Such analysis would not be possible without the use of IBCC. This shows that this technique is effective at classifying, but it extends past this in order to provide useful information about the base classifiers. Knowledge gained about the prompts can then be carried forwards into further experiments, in order to balance the classifiers further.

A final confusion matrix of note is for prompt 6. This prompt reads the data point and appends “unrelated [MASK] sentence” onto it. Even despite this, its inclusion in the IBCC improved its performance by almost 5 percentage points. Its learned confusion matrix is given in Figure 3.5.

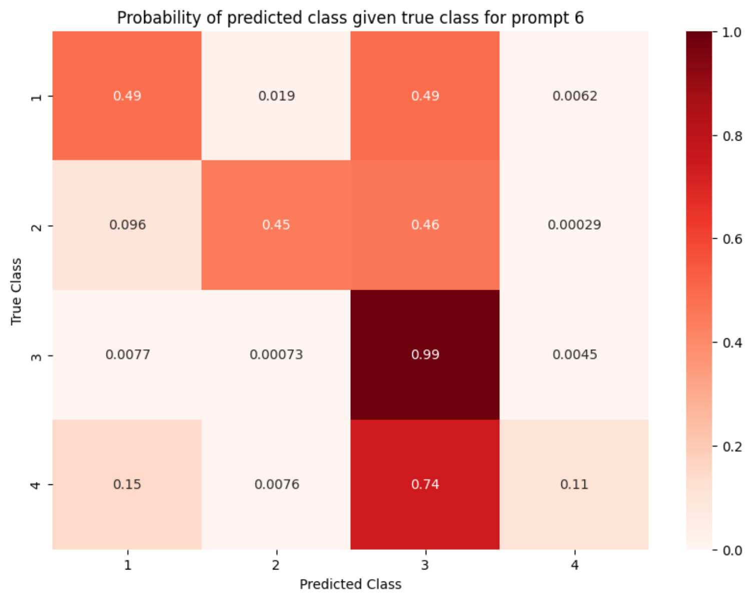

Figure 3.5: Heatmap for the confusion matrix of prompt 6

It can be seen that similar to other prompts mentioned, this prompt is really good at classifying into the business category, even though its mask sentence is unrelated. It can be seen that this is because it predicts class 3 most often - for each other true class the probability it picks class 3 is still high. This is what I aimed for when creating this prompt. It has a bias towards class 3 because that word fits best in the unrelated sentence for most data points. Even though for class 1 it predicts classes 1 and 3 equally often, the bias it has towards class 3 has been considered, so the combination can still use this prompt’s limited knowledge for class 1 to its advantage - it is still the case that if the prompt gives a prediction of class 2, then it is unlikely to be a class 1 data point. The independence of this classifier is valuable to the combination and improves its performance.

3.3.5 Stronger Prompts

Now that it has been established that an IBCC combination is robust to poor prompts and can leverage them to its advantage, my next goal is to test the limits of this method when I am not passing in base classifiers which I have designed to sabotage it. Therefore, I will design and use five more prompts which all have the aim to perform as well as possible. This extra set of 5, given in Table 3.10, will be used in combination with the initial 5 well-performing prompts to create a combination classifier of 10 base classifiers, all of which have good scores. The individual performances of the extra prompts are given in the table as well.

ID	Prompt	Accuracy (%)
11	“`[title]`. `[body]`: [MASK]”	29.7
12	“`[title]`. `[body]`. - [MASK] News.”	**61.8**
13	“AG News: ”`[title]`. “[MASK] Category. `[body]`”	58.3
14	“The following article is in the [MASK] category: `[title]`. `[body]`.”	59.3
15	“`[title]`. `[body]`. I think this article talks about [MASK].”	58.5

Table 3.10: Extra prompts

Prompt 11 is designed to be extremely simple, the sentence added to fill consists of a single colon, in order to symbolise that what comes before it implies the thing that comes after it. Prompt 12 was designed to also use the word “news”, since it has been shown that this extra context helps the model with some categories. Similarly, prompt 13 adds the word “news” at the start of the data point to try and supply some overall context, while the classification sentence is separate and in the middle, in order to ensure information is on either side of the mask token. This is the first prompt that appends two separate pieces of text to the raw data. Prompt 14 is an adaptation of prompt 1, with the classification sentence being at the beginning, relying on bidirectional information to classify the category. Finally, prompt 15 is similar to prompts 1 and 2, but it is designed using the word “think”, with the aim to provide a different point of view for classification by the model, by putting the classification in a subjective sentence rather than an objective one. All of these prompts are designed to be relevant to the task and perform well, contrasting the previous experiment which included base models which were intentionally wrong.

The most obvious thing to note after running the individual performance tests is the extremely poor performance that prompt 11 achieves. It is clear that the single punctuation mark followed by the mask token does not provide enough context about the task to the classifier, so it gets a large number of inputs wrong. However, the rest of the prompts attain a similar accuracy level to the first five prompts, with the best individual prompt now being prompt 12, scoring 61.8%, which supports the idea that utilising the word “news” in prompts is key to getting a good performance. Next, I performed an IBCC combination for the classifications made by base classifiers corresponding to prompts 1-5 and 11-15. For a fair comparison, I will also use a majority vote combination on the same prompts. The accuracy scores for this, along with results for previous experiments is given in Table 3.11.

Method	Accuracy (%)
Best single prompt (ID 12)	61.8
IBCC with 5 prompts	64.3
IBCC with 10 varied prompts	**72.4**
IBCC with 10 strong prompts	63.1
Majority vote with 10 strong prompts	62.1
Schick et al.	69.5
Zero-shot pipeline	75.3

Table 3.11: Results comparison for combination using better prompts

These results are surprisingly poor. The IBCC of the strong prompts now only achieves an accuracy of 63.1%, which only beats the majority vote combination of 62.1% by one percentage point. This is a large drop in performance from its equivalent using poor prompts. There are parallels between this experiment and the equivalent comparison with 5 strong prompts which I performed at the start; the IBCC only slightly outperformed the majority vote. This suggests that the better prompts given to the IBCC model are not diverse enough in their predictions for the IBCC to identify the true underlying patterns to the same extent. The number of prompts provided means there are more pairs of prompts that have correlated errors, so the combination will have more bias. While this was not my expectation entering into this experiment, it is still a useful result. It determines that in order for the IBCC to perform at its best, it needs bad prompts in order to balance out the inherent bias that the underlying language model provides. This experiment does not detract from the usefulness of IBCC though, because I can still assess the reason that prompt 11 performs so badly in the beginning. The confusion matrix learned by the model about this prompt is shown in Figure 3.6.

It can still be seen that the reason that prompt 11 performs badly is that it has a very weak diagonal - most diagonal values are far from 1. The stripe for predicted class 4 is similar to what happened in prompt 9, which was shown in Figure 3.3. A confusion matrix like this arises when the model predicts class 4 most of the time. Upon closer inspection, there is another, less severe stripe for class 1. Therefore, it can be concluded that this prompt usually predicts class 4, and sometimes predicts class 1, mostly irrespective of the true class of the data. Hence, it is clear that the prompt which appended “: [MASK]” to the data point induces a bias towards “science” or “technology”. This is because of a bias in the underlying model; it will have a natural preference for each category word proportional to how frequently each word appeared in its training, meaning that given little context it will be more likely to choose this category. This represents vital information about the model for creating more prompts in the future.

To conclude the experimental stage, a complete table of results is provided in Table 3.12. This table shows the similarity between the experiments with 5 strong prompts and 10 strong prompts, as well as displaying that an IBCC always outperformed its equivalent majority vote combination.

Overall, IBCC reaches a limit when used in this way due to its requirement for independent base classifiers and the inherent inability to ensure this when using the same base pre-trained language model. On

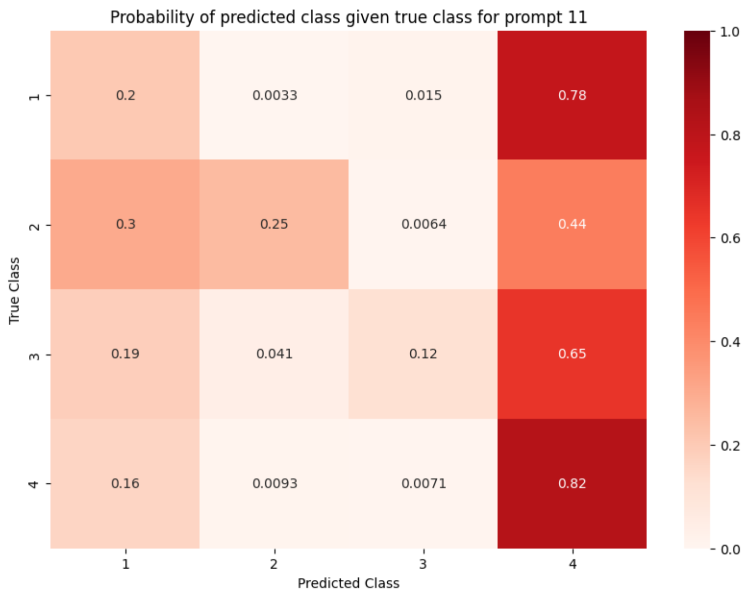

Figure 3.6: Heatmap for the confusion matrix of prompt 11

Method	Accuracy (%)
Best single prompt (ID 12)	61.8
Majority vote with 5 prompts	63.7
Majority vote with 6 prompts	64.9
Majority vote with 10 varied prompts	63.0
Majority vote with 10 strong prompts	62.1
IBCC with 5 prompts	64.3
IBCC with 6 prompts	69.0
IBCC with 10 varied prompts	**72.4**
IBCC with 10 strong prompts	63.1
Schick et. al (RoBERTa)	69.5
Zero-shot pipeline	75.3

Table 3.12: Results for all completed experiments

the other hand, this means that adding more prompts is not necessarily required for a good performance. As the IBCC always outperforms the majority vote combination, my experiments show that it is always worth using an IBCC combination, as the classification it supplies is still of sufficient quality.

Chapter 4

Critical Evaluation

This chapter will provide an assessment of the viability of an IBCC and its usage in NLP in order to answer the hypothesis. It will then assess the process of the project, in order to evaluate the strengths of the results presented.

4.1 IBCC Evaluation

Moving on to Objective 6 given in Chapter 1, this section will lay out the limitations and successes of IBCC when used for NLP and text classification using cloze-phrase sentences.

4.1.1 Limitations

When using an ensemble combined using an IBCC, the main limitations to the effectiveness of the method are as follows.

1. IBCC requires independent base classifiers.
2. IBCC requires $N \times P$ classifications.

The first limitation turns out to be quite significant. Despite providing a different prompt in every base classifier, using the same underlying language model to provide the predictions in each prompt provides a lot of correlation between each model. The inherent biases within the model can significantly affect classification in some cases. This holds the IBCC model back and allows some scenarios where the IBCC does not significantly outperform a majority vote combination. A potential remedy for this could be to fine-tune the models on each specific pattern, similar to Schick et al. [27], so that the model understands its individual prompt better. This could lessen the weight that the underlying model and its inherent biases have, making each model more independent from the others and improving results.

The second limitation means that there is a soft limit to how many prompts can be included in the combination, especially as the number of data points to classify increases. However, in combination with the previous limitation, this one is not as impactful, since using many prompts from the same language model would cause limited improvement due to the correlation between the many prompts that would be present.

4.1.2 Successes

Despite these limitations, this methodology is still very successful when it comes to classification in NLP. The main successes for this method are as follows.

1. IBCC outperforms majority vote in all experiments.
2. IBCC provides extra information about base classifiers.
3. IBCC does not require extra data or context about the task.
4. IBCC takes less computational time while obtaining a comparable result.
5. IBCC is resilient to poor prompts.

These successes display the benefits of using an IBCC combination for zero-shot text classification. Using such a combination removes requirements for data while providing a good accuracy. Furthermore, it is not influenced poorly by bad base classifiers. Together, these successes decrease the time it takes for text classification, while adding little overhead. Prompts can be designed and used quickly due to the model's robustness, and the feedback it can give about the prompts used can be used to improve them if required. All in all, the model provides a notable improvement in performance with minor extra requirements, and performs comparably with other cloze-phrase work and off-the-shelf zero-shot classification language models. My experiments have shown that it is almost always better than using a majority vote.

4.2 Project Evaluation

With the nature of a project aiming to explore the viability of a certain method, it is impossible to explore all possibilities and potential scenarios that the combination can be put in. When experiments are performed, the results often open up more loose ends than they close. The results and their consequences were assessed continuously as the experiments took place, so this section will serve to evaluate the choice of routes I explored, while mentioning some interesting potential pathways that I did not get a chance to.

4.2.1 Research

Given the length of the project, the amount of research conducted was certainly sufficient to develop my understanding and apply it to the language-based classification task. Having an understanding of a few prior developments for similar research questions was practical and allowed me to take inspiration from them. Understanding previously developed language models and their inner workings and functionality was important and my research into Hugging Face [14] was thorough to reflect this. Their introduction course for NLP problems was useful and a good use of time in order to utilise their library successfully. Further, the use of pre-trained language models from the Hugging Face library was an obvious and successful choice. Using these allowed me to focus on the ensemble aspect of the classification, without the requirement to create or train language models myself. Research about few-shot text classification led to the Schick et al. paper [27], which served as an inspiration to use cloze-phrases for classification. This was a successful route to take for classification, as well as being interesting and serving as a connection to human thought.

4.2.2 Execution

The first obvious shortfall in the execution of this project was that experiments were only performed on a single data set. Experiments on multiple data sets with different subjects to classify on would be needed to come up with a conclusive result regarding whether IBCC is a suitable method for text classification of this form. Since the base classifiers have no specific connection to this data set, I believe the results are still valid as the data set serves as a general task for classification. While executing the project, I began to test IBCC in a similar way with somewhat similar prompts on a Yahoo questions data set [4]. This data set contains a question and answer pair posted to a forum, along with a label corresponding to which category the question was posted to. This is another commonly used data set and is also used in the work of Schick et al [27]. However, when generating some classifications the model would fail because some inputs were too long. Deleting these entries causes misalignment with class labels, and with the project coming to a close, developing a robust solution to handle this was not possible. This means that the results for the AG News data set are the only ones presentable as findings. Results for other data sets would make my conclusions more solid and robust, however not having this does not take away validity from my findings.

Since generating predictions from multiple base classifiers is the most computationally heavy part of the execution, I chose to generate all predictions in one go, and then save them to use later. This was an informed initial decision and made my project more efficient and made experiments a lot smoother. Being able to change parameters for an IBCC classification without generating more predictions is extremely useful and having predictions already available facilitated that. However, as my initial focus was on results, I first analysed the accuracy and then the confusion matrices which explain what the combination has learned about the base classifiers. These matrices are useful - for example, they showed that the use of the word “news” in a prompt improves performance for the “world” category. However, the nature of creating prompt classifications once made the project less malleable to the extra information gained from each combination. This made it harder to act upon the information gained to change the prompts, because it would require reclassification for each experiment. There are no easy solutions to this problem, as classifying all data points will always take a long time and ideally would be performed as few times as possible. One potential remedy would be to create predictions on a prompt-by-prompt basis. That is, design one prompt and generate an entire list of its predictions and save those. This would make selecting a subset of all of the prompts easy, as well as removing one from a combination while including an improved version. This would still require the creation of N predictions for each prompt, but would split it up into smaller chunks; one extra prompt's predictions would not take too long to add in, especially if previous experiments justified a modification in this manner. I would recommend this approach for any similar ensemble-based task in the future.

Despite these, the route I took through the project was justified and successful. The execution of this project tells a meaningful story. It makes sense to begin with a single prompt in order to establish a footing in the problem and build from there. Generating and designing prompts was given adequate attention. With the knowledge that the eventual solution, IBCC, would learn about the base classifiers, prompts did not need to be designed meticulously, however, I gave enough attention to them to ensure a good classification accuracy to begin with, as shown in Table 3.1. Using the Hugging Face transformers library was a positive choice, as it is accessible and allowed me to harmonise my skills in other Python libraries with it, which led to efficient development and ensured the combination could receive adequate attention. The design of prompts which were expected to perform worse was also a good decision, as it gives the opportunity to display the strengths of IBCC when comparing against a majority vote combination. When combining prompts, I started with a simpler combination to illustrate the benefits of a combination, and then moved to IBCC to display its strengths. The majority vote performs as expected for a basic ensemble - it is hampered by bad prompts but improves performance on individual good prompts. This indicates that it was implemented correctly, as well as playing its part well as a comparison for an IBCC. Further, when assessing the performance of each method, I have provided meaningful benchmarks which contextualise comparisons to other work in the field. None of the results subvert initial expectations based on research, which indicates that the project was performed correctly, and illustrates the strengths of the explored method along with an analysis to back this up.

4.3 Extensions

Work for the future should be focused on alleviating the effect of the limitations mentioned in Section 4.1.1, as well as exploring some alternatives offered in Section 4.2.2. The main limitation of IBCC listed is that it requires independent base classifiers. It turned out that even with different prompts, the underlying language model made the classifications dependent on each other. This problem opens a lot of routes with potential remedies to this problem. Firstly, the paper that introduces the IBCC [18] also introduces a dependent model for Bayesian classifier combination. This model incorporates the dependence into the underlying mathematical combination function, and would potentially be better suited to scenarios similar to this one - where due to the nature of the task it is unlikely to be able to ensure independence of base classifiers. Another approach to this problem could be a change to the base classifiers themselves. As performed in Schick et al. [27], given a small amount of correct data, the base classifiers could be fine-tuned on their respective prompts. This may make the classifiers less dependent on each other by lowering the influence that the underlying model has on text generation. Finally, with a small amount of correct data, it can actually be passed into the IBCC itself, showing the combination for certain how the base classifiers act on a few data points. This will give it a better understanding of the errors that each classifier makes and could have the potential to provide a more accurate combination prediction. Any sort of change or development to the prompts could be done using the confusion matrices provided by the model to inform new designs or alterations in prompts. Therefore, as much independence as possible could be created and the performance of the IBCC might improve. All of these methods would be interesting to explore and compare accuracy measures to the results already found in this project.

Another extension is to apply this method to other data sets. There are other available data sets, and now initial experiments have shown that this method can perform well, there is motivation to test it on different data. This will bring another dimension of comparison to use with the method and will give any results found more strength to assert conclusions about IBCC's performance in text classification. With this in mind, further work should remedy the requirement for a lot of classifications and the restriction this brings by being completed in one go at the beginning by designing iterative experiments which use information given from the confusion matrices to proceed. This will allow analysis of how well prompts perform individually in other tasks, and if tested on enough different data sets, might provide a template for what provides a good prompt and what is a bad one. This could guide research into the matter and a more generalist approach, which would break the method away from assessing its performance on only the AG News data set, and assert it as a transferable model.

Chapter 5

Conclusion

5.1 Contributions

This project has successfully experimented with IBCC to create an ensemble to be used to improve the performance of zero-shot text analysis. Starting with a single member of the ensemble, I used a pre-trained language model which is specialised at filling gaps in sentences, in combination with classification sentences about the input text which contain a mask token that the model filled. Each individual classifier in the ensemble was created by using the same language model to fill gaps in different appended sentences, called prompts. By restricting the predictions for the missing word to the available classes, this functioned as a natural language classifier which was used to classify data points in the AG News data set [29], which contains pairs of news headlines and body text. The individual pre-trained language model was used in a zero-shot manner, meaning it was given no extra context about the task it was completing, which was used to save a requirement for training data and extra computation. The first individual classifier performed well on its own, gaining an accuracy of 58.8%. After this, I created further prompts with the aim of combining them, and the extra prompts all performed to a similar degree. With the future in mind, I designed some prompts that would design worse, including one which randomly chose a prediction, irrespective of the text itself. These prompts served the purpose of creating an experiment where not all members of the combination can be trusted, in order to demonstrate the downfalls of simpler combination functions.

When combining the base classifiers, I used two different combination functions, a majority vote and an independent Bayesian classification combination. A majority vote takes the most common answer given by each of the base classifiers and uses them, and an IBCC uses a more complex Bayesian technique to find the class with the highest likelihood when considering all answers. When combining five prompts that perform well, the majority vote demonstrates the benefits of an ensemble model by scoring 63.7%, which is an improvement on all individual models. The IBCC combination exceeds this score, with 64.3%. The highlight of the project comes when including these prompts as well as the five designed to perform poorly. In this scenario the majority vote is hampered, having its score drop to 63.0% accuracy, whereas the IBCC combination utilises the strengths of each of the base classifiers while accounting for their weaknesses, and achieves a score of 72.4%. This result demonstrates how IBCC is a better combination function in most scenarios. In all experiments, IBCC outperforms majority voting, and IBCC is resilient to base classifiers that perform worse.

Further experiments showed that when more prompts are designed which are not intended to perform badly, and combined in a similar way, the IBCC outperforms majority voting but does have a drop in performance to 63.1% accuracy, compared to the majority vote's 62.1% in the same experiment. Since the majority vote's score also dropped, it is likely the classifications were worse in this generation. However, the drop in IBCC performance is large and is likely because of the requirement for independence of base classifiers when using an IBCC. Making the classifiers predict based on different prompts does make answers less dependent on each other, but the underlying pre-trained language model means that the classifications make correlated errors, which hampers the IBCC's ability to learn how well each prompt performs for each class.

Overall, this report successfully illustrates the effectiveness of an IBCC to boost the performance of a classifier on a zero-shot task, where it cannot gain any extra context about the task it is being used on. Its score outperforms other combinations as well as the equivalent zero shot experiment in 2.4. The combination performs comparatively with an off-the-shelf zero-shot model designed for classification in tasks with less context, however, the zero-shot model requires more training and takes longer to classify. This balances out its slight edge over the IBCC combination and illustrates that the IBCC has a place in zero-shot classification, which is a common task in real-world scenarios

5.2 Project Status

All aims given in chapter 1 were achieved in the duration of the project. Objective 1 was to gain an understanding of existing work in the area. I have achieved this and have laid out important and relevant existing work in chapter 2. This includes research into zero-shot classification models given on Hugging Face and an evaluation of the usage of cloze-phrases for classification in Schick et al. [27]. These illustrate my motivation which route was taken to begin the project as well as supplying a basis to build on throughout. During my research, I also have gained a thorough understanding of Bayesian combinations, which was given in Objective 2. Research towards this is presented in Section 2.6 and demonstrates full technical knowledge of the mechanism of an IBCC combination. After this, Section 3.2 illustrates the design and implementation of cloze-phrase prompts, as well as the generation of individual base classifier predictions based on these. When adapted to each data point, the prompts perform well, indicating a successful implementation and sufficient design. The classifiers' performances are analysed in this section also. Therefore, Objectives 3 and 4 are completed to a good standard. In Section 3.3, a thorough guide is given through the process of creating an ensemble out of base classifiers, along with the next steps motivated by previous results, which concludes with a complete comparison of IBCC's performance against majority vote as well as other existing classifiers. Objective 6 refers to exemplifying the effectiveness and resilience of IBCC as a classification tool. I have produced results that illustrate this. The project moved on to testing on alternate data sets, however, this experiment encountered issues and was not run to a conclusion. Therefore Objective 6 is completed, but it is hard to assert that IBCC is beneficial as a transferable concept without successful experiments on other data sets.

5.3 Future Work

The following outlines potential routes for future work to further explore the viability of this method for combinations. More detail for each option can be found in Section 4.3.

1. Conduct comparable experiments on a different data set. I began doing a similar experiment, however, this experiment is incomplete. Results on another data set would open the door to further justification for using IBCC for zero-shot classification and illustrate more thoroughly its uses. This is justified because of the good performance of IBCC on the AG News data set. Good results on a different data set would cement the place of an IBCC for improving performance in zero-shot classification.

2. Perform experiments to decorrelate the base classifiers. Since my results in Section 3.3.5 displayed a drop in combination quality, further experimentation into this would be interesting. In order to decorrelate the classifiers, the confusion matrices could be used to learn about which have the most correlated errors.

3. Use a dependent Bayesian combination. Although this method is more mathematically involved and beyond the reach of this project, a dependent Bayesian combination would be able to account for biases in underlying language models better and could provide a better classification score.

Bibliography

[1] Bart-large-MNLI. https://huggingface.co/facebook/bart-large-mnli. Accessed 24/04/23.

[2] BookCorpus. https://yknzhu.wixsite.com/mbweb. Accessed 24/04/23.

[3] English Wikipedia. https://en.wikipedia.org/wiki/English_Wikipedia. Accessed 24/04/23.

[4] Yahoo questions data set. https://huggingface.co/datasets/yahoo_answers_topics. Accessed 10/04/23.

[5] Zooniverse. https://www.zooniverse.org/. Accessed 19/04/23.

[6] Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online, November 2020. Association for Computational Linguistics. URL: https://aclanthology.org/2020.findings-emnlp.148, doi:10.18653/v1/2020.findings-emnlp.148.

[7] Christopher M Bishop. Pattern recognition and machine learning, ch. 14. 2006. URL: https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf.

[8] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28, 1979. URL: http://www.jstor.org/stable/2346806.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL: http://arxiv.org/abs/1810.04805, arXiv:1810.04805.

[10] Harris et al. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi:10.1038/s41586-020-2649-2.

[11] Simpson et al. Dynamic bayesian combination of multiple imperfect classifiers. URL: https://www.robots.ox.ac.uk/~sjrob/Pubs/galaxyZooSN_simpson_etal.pdf.

[12] Waskom et al. mwaskom/seaborn: v0.8.1, 2017. Accessed 30/04/23. doi:10.5281/zenodo.883859.

[13] Hugging Face. AG News dataset. https://huggingface.co/datasets/ag_news. Accessed 13/02/23.

[14] Hugging Face. Hugging Face models. https://huggingface.co/models. Accessed 24/01/23.

[15] Hugging Face. NLP Course. https://huggingface.co/learn/nlp-course/chapter1/1. Accessed 28/01/23.

[16] Hugging Face. Transformers Library. https://huggingface.co/docs/transformers/index. Accessed 24/04/23.

[17] Francis Galton. Vox populi. 1907. URL: https://doi.org/10.1038/075450a0.

[18] Ghahramani and Kim. Bayesian classifier combination. *Gatsby Computational Neuroscience Unit Technical Report*, 2003. URL: https://mlg.eng.cam.ac.uk/zoubin/papers/GhaKim03.pdf.

[19] Dilek Küçük and Fazli Can. A tutorial on stance detection. https://dl.acm.org/doi/pdf/10.1145/3488560.3501391, 2022.

[20] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461, 2019. URL: http://arxiv.org/abs/1910.13461, arXiv:1910.13461.

[21] Sasha Luccioni, Victor Schmidt, Alexandre Lacoste, and Thomas Dandres. Quantifying the carbon emissions of machine learning. In *NeurIPS 2019 Workshop on Tackling Climate Change with Machine Learning*, 2019. URL: https://www.climatechange.ai/papers/neurips2019/22.

[22] Wes at al. McKinney. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[23] Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. SemEval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, San Diego, California, June 2016. Association for Computational Linguistics. URL: https://aclanthology.org/S16-1003, doi:10.18653/v1/S16-1003.

[24] Josiah Ober. An aristotelian middle way between deliberation and independent-guess aggregation. 2009.

[25] OpenAI. Gpt-4. https://openai.com/research/gpt-4, 2023. Accessed on 18/04/2023.

[26] Raul Puri and Bryan Catanzaro. Zero-shot text classification with generative language models. *CoRR*, abs/1912.10165, 2019. URL: http://arxiv.org/abs/1912.10165, arXiv:1912.10165.

[27] Timo Schick and Hinrich Schütze. Exploiting cloze questions for few shot text classification and natural language inference, 2021. arXiv:2001.07676.

[28] Wenpeng Yin, Jamaal Hay, and Dan Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. *CoRR*, abs/1909.00161, 2019. URL: http://arxiv.org/abs/1909.00161, arXiv:1909.00161.

[29] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.

Appendix A

Heatmaps

This appendix contains all heatmaps from both combination experiments given in Section 3.3.2 and Section 3.3.5.

A.1 First Experiment

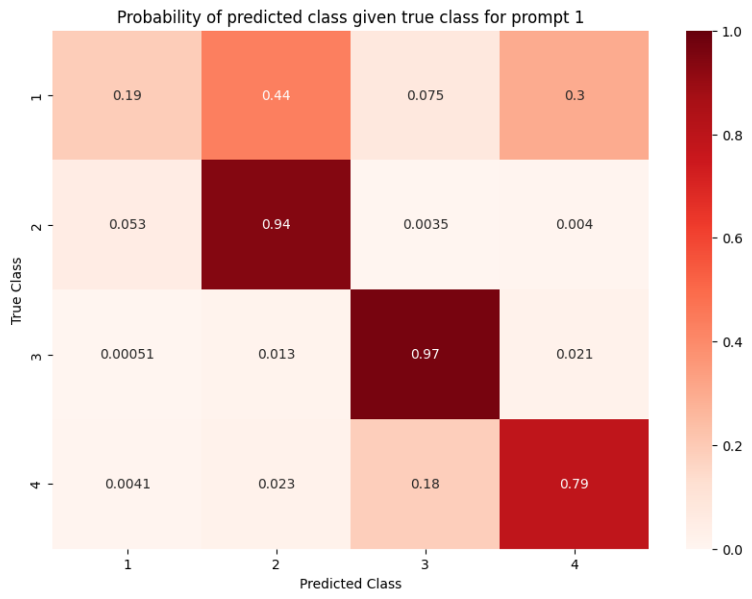

Figure A.1: Heatmap for the confusion matrix of prompt 1

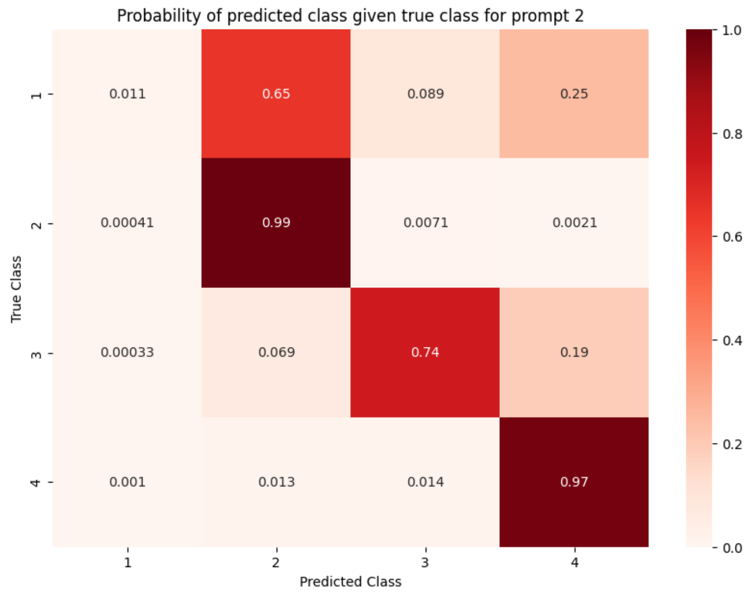

Figure A.2: Heatmap for the confusion matrix of prompt 2

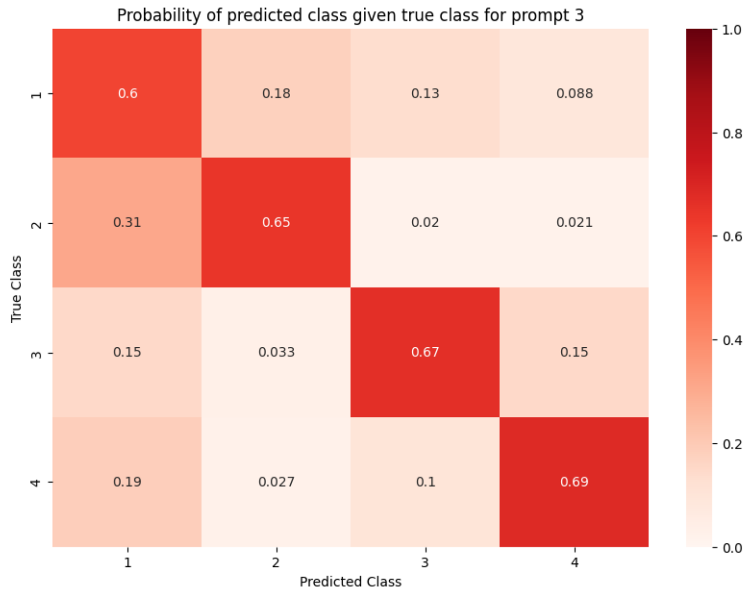

Figure A.3: Heatmap for the confusion matrix of prompt 3

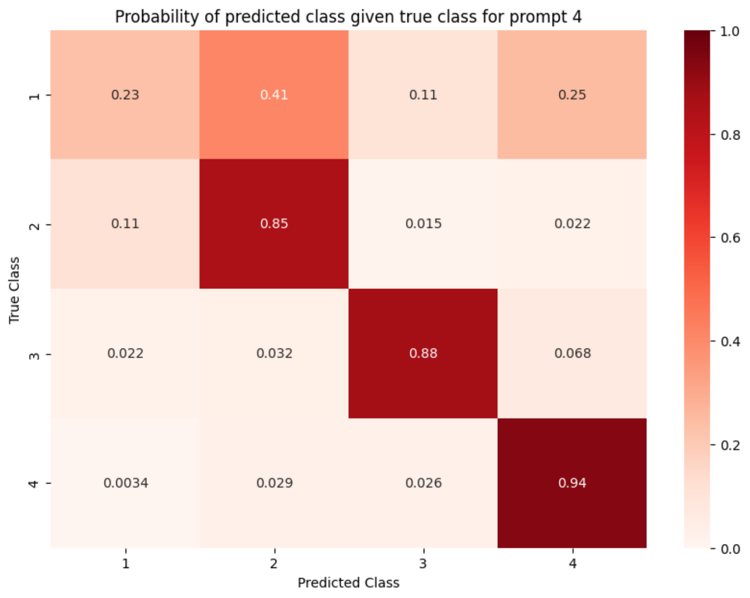

Figure A.4: Heatmap for the confusion matrix of prompt 4

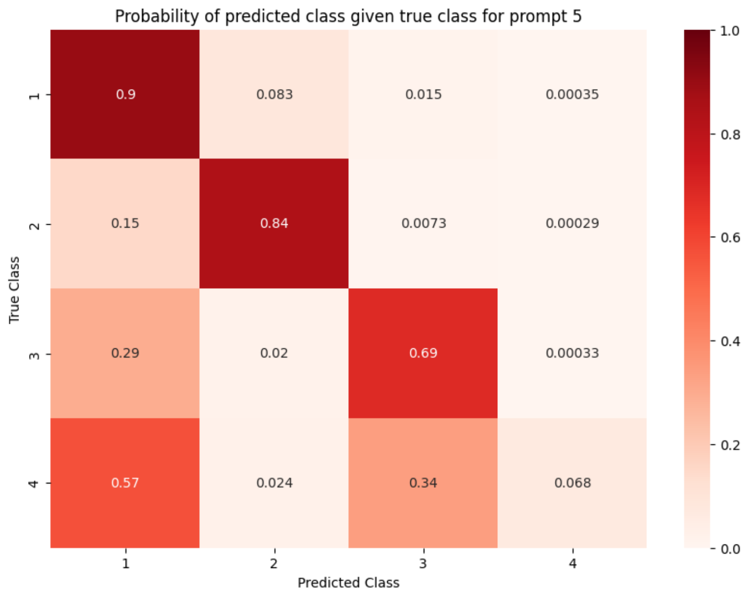

Figure A.5: Heatmap for the confusion matrix of prompt 5

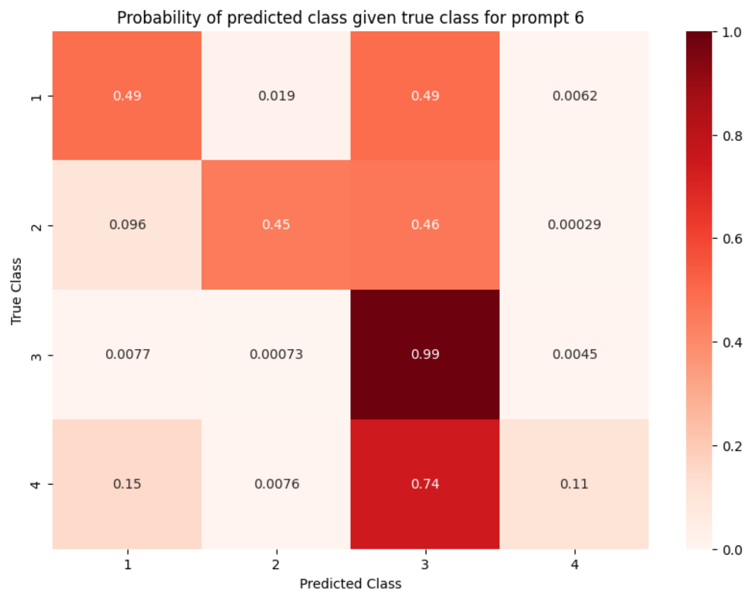

Figure A.6: Heatmap for the confusion matrix of prompt 6

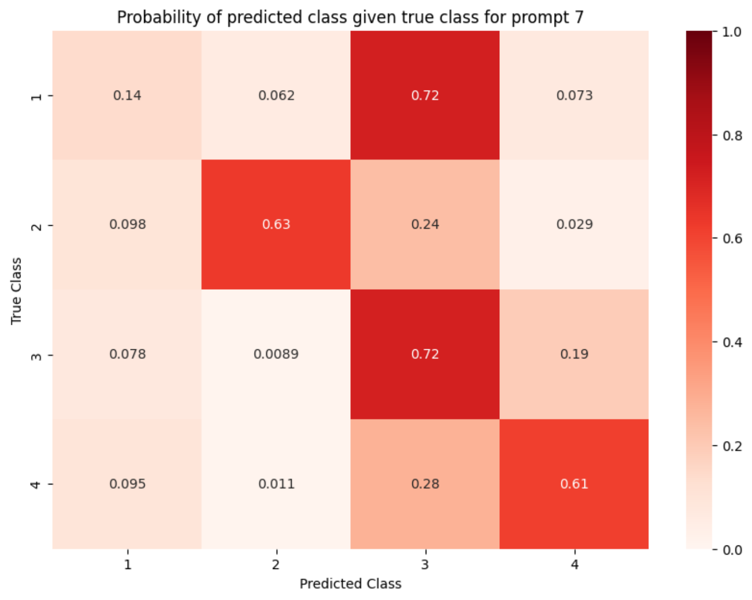

Figure A.7: Heatmap for the confusion matrix of prompt 7

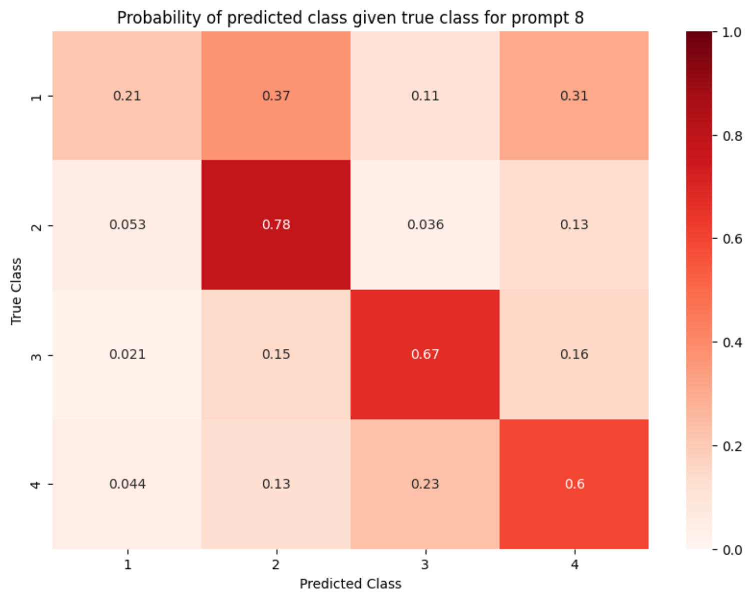

Figure A.8: Heatmap for the confusion matrix of prompt 8

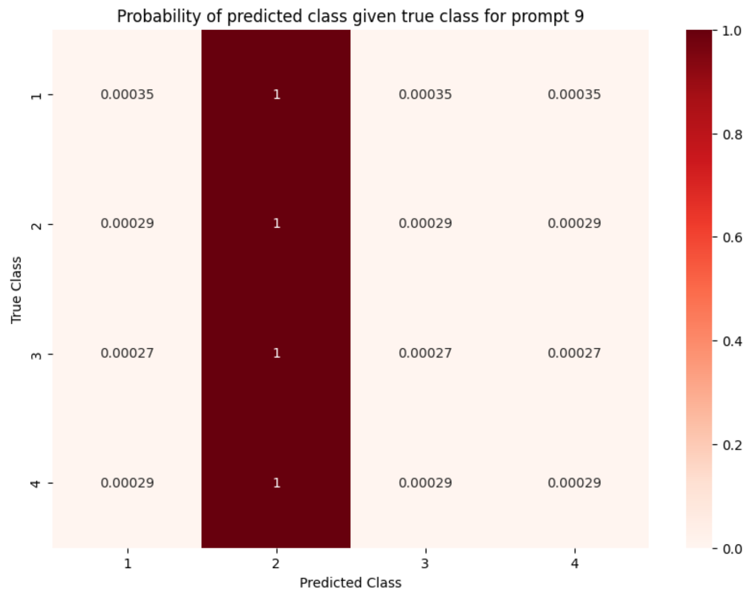

Figure A.9: Heatmap for the confusion matrix of prompt 9

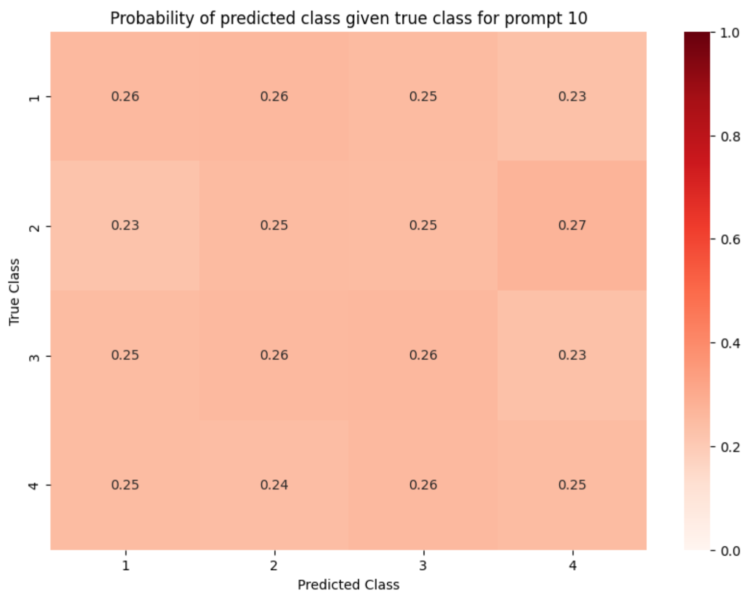

Figure A.10: Heatmap for the confusion matrix of prompt 10

A.2 Second Experiment

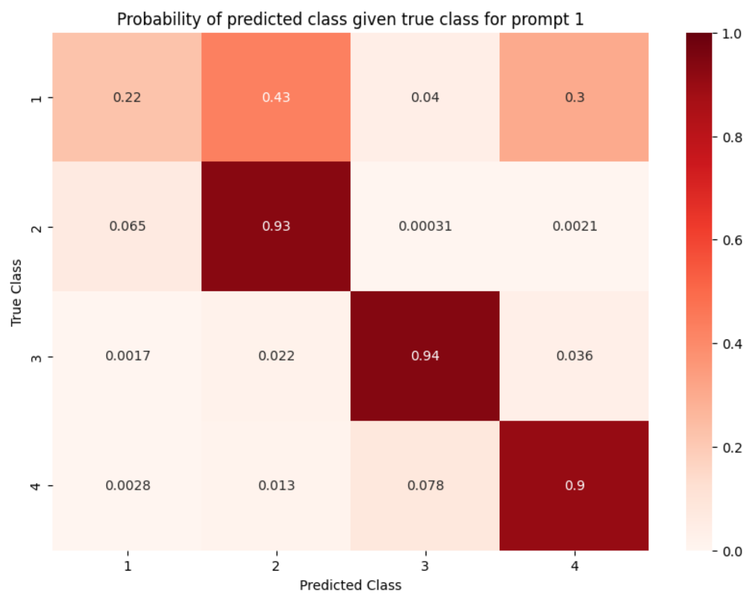

Figure A.11: Heatmap for the confusion matrix of prompt 1

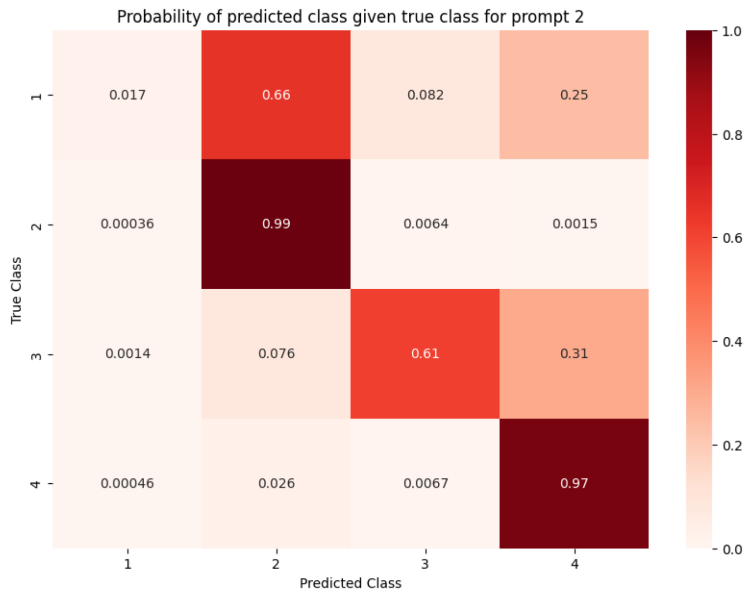

Figure A.12: Heatmap for the confusion matrix of prompt 2

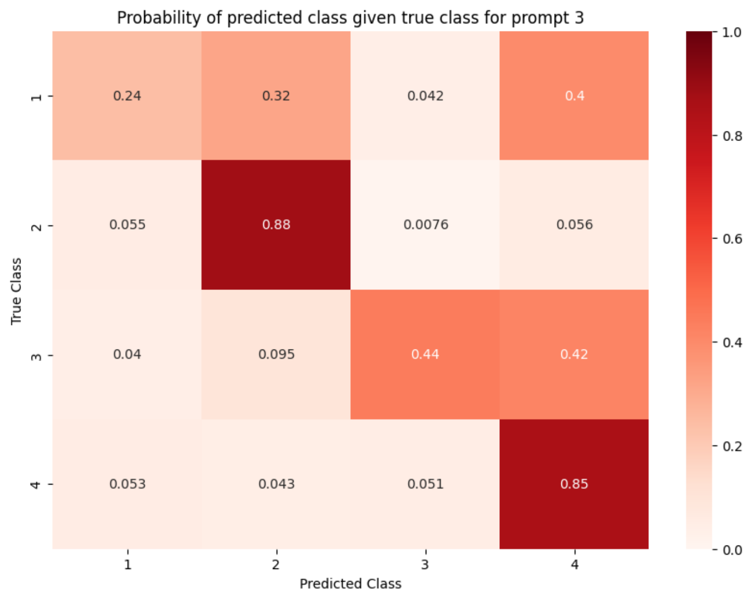

Figure A.13: Heatmap for the confusion matrix of prompt 3

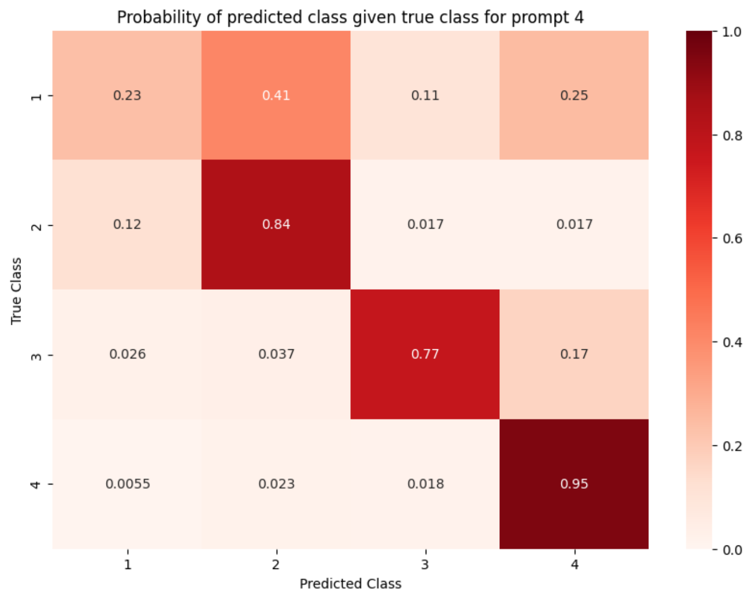

Figure A.14: Heatmap for the confusion matrix of prompt 4

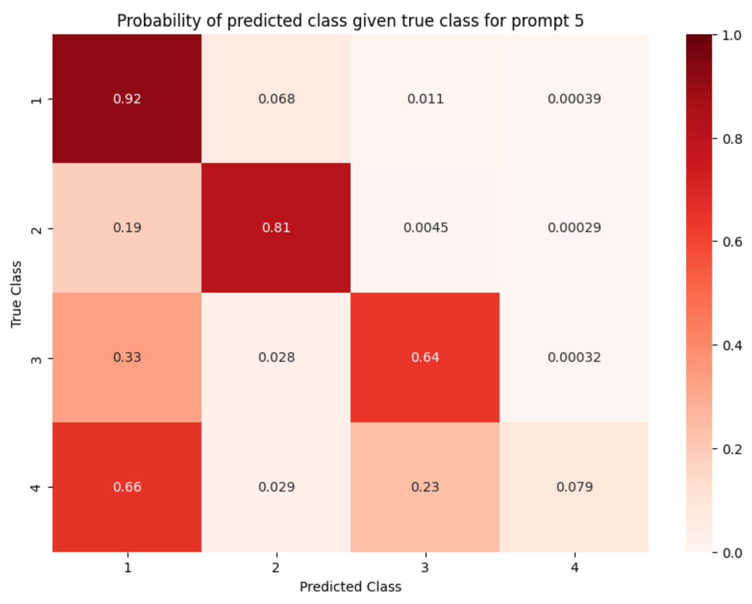

Figure A.15: Heatmap for the confusion matrix of prompt 5

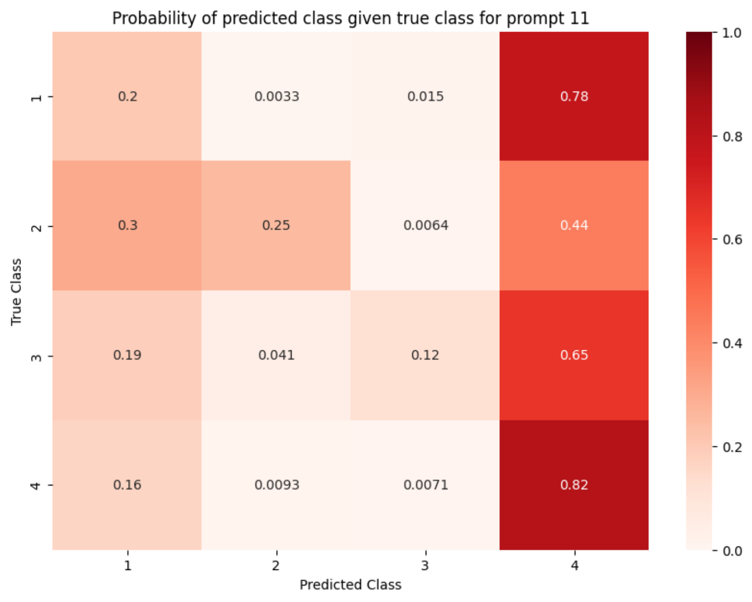

Figure A.16: Heatmap for the confusion matrix of prompt 11

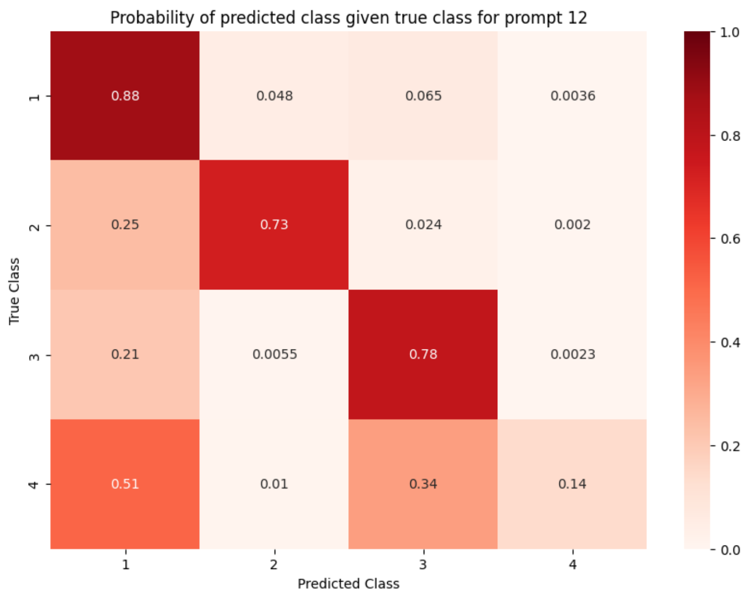

Figure A.17: Heatmap for the confusion matrix of prompt 12

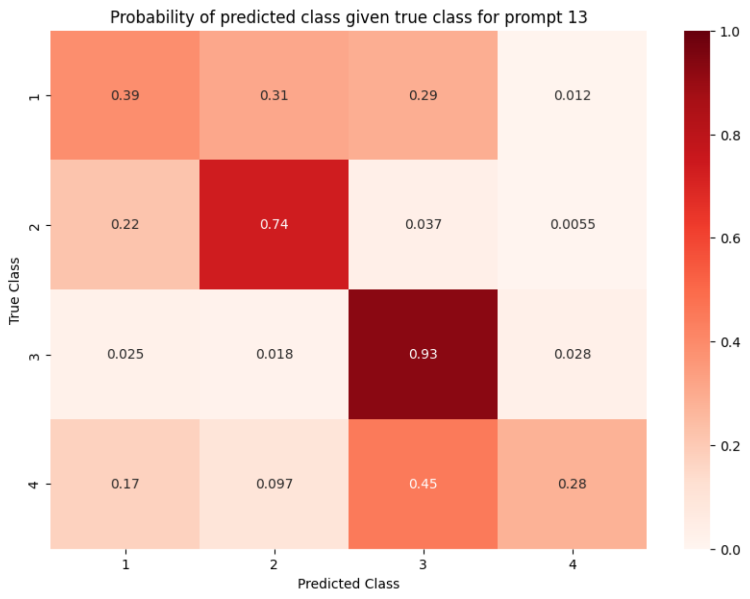

Figure A.18: Heatmap for the confusion matrix of prompt 13

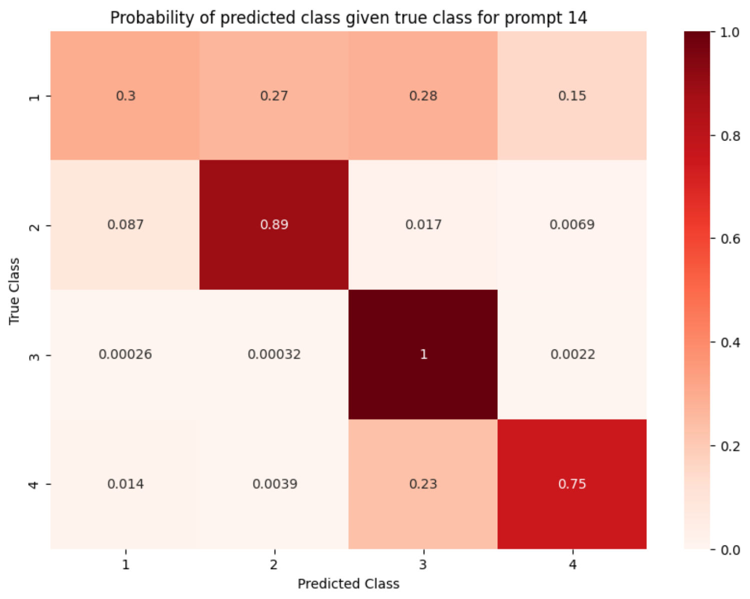

Figure A.19: Heatmap for the confusion matrix of prompt 14

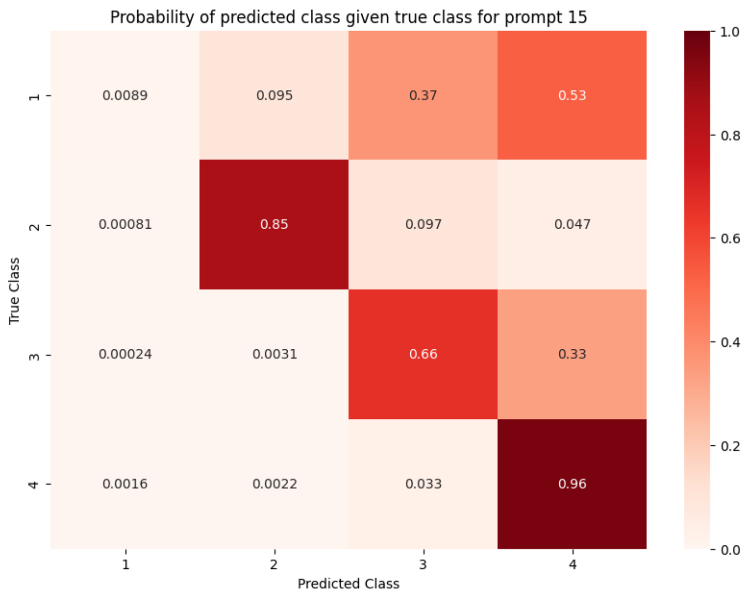

Figure A.20: Heatmap for the confusion matrix of prompt 15