Uncertainty Estimation for Black-Box Classification Models: A Use Case for Sentiment Analysis

José Mena[1,3(✉)], Axel Brando[2,3], Oriol Pujol[3], and Jordi Vitrià[3]

[1] Eurecat, Centre Tecnològic de Catalunya, Barcelona, Spain
jose.mena@eurecat.org
[2] BBVA Analytics Data & Analytics, Madrid, Spain
axel.brando@bbvadata.com
[3] Universitat de Barcelona, Barcelona, Spain
{axelbrando,oriol_pujol,jordi.vitria}@ub.edu

Abstract. With the advent of new pre-trained word embedding models like ELMO, GPT or BERT, that leverage transfer-learning to deliver high-quality prediction systems, natural language processing (NLP) methods are reaching or even overtaking human baselines in some applications. The basic principle of these successful models is to train a model to solve a given NLP task, mainly Language Modelling, using significant volumes of data like the whole Wikipedia. The model is then fine-tuned to solve another NLP task, requiring fewer domain-specific data to achieve state-of-the-art accuracies. The method proposed in the present work assists the practitioner in evaluating the quality of the transferred classification models when applied to new data domains. In this case, we consider the original model as a black box. No matter how complex the original model may be, the method only requires access to the output layer to train a measure of the uncertainty associated with the predictions of the original model. This measure of uncertainty is a measure of how well the black-box model accommodates to the new data. Later on, we show how a rejection system can use this uncertainty to improve its accuracy, effectively enabling the practitioner to find the best trade-off between the quality of the model and the number of rejected cases.

Keywords: Sentiment analysis · Transfer learning · Uncertainty estimation · Natural Language Processing

1 Introduction

The application of Natural Language Processing (NLP) methods to real-world problems is gaining momentum nowadays thanks to significant advances resulting from the field of Machine Learning, and Deep Learning (DL) in particular. Recently, the appearance of new word embedding models such as BERT [16], ELMO [17] or GPT [18] has taken the quality of prediction models for tasks

A. Morales et al. (Eds.): IbPRIA 2019, LNCS 11867, pp. 29–40, 2019.
https://doi.org/10.1007/978-3-030-31332-6_3

such as Sentiment Analysis, Text Entailment or Question Answering, to a new level, reaching or even overtaking human baselines in some cases.

All these models share the concept of transfer learning to train models that solve a Language Modelling task to learn the fundamental structure of a given language by using vast volumes of data, like the whole Wikipedia, or a collection of News with more than 100 billion words. Once trained, they apply these models to other NLP tasks like Sentiment Analysis, fine-tuning them using a domain-specific dataset and obtaining superior accuracy metrics for the new task, thanks to the fact that the pre-trained model has already incorporated the necessary knowledge about the given language.

The success of these models is fostering the proliferation of new prediction services in the form of application programmable interfaces (API). In this scenario, one of the aforementioned generic models is trained in a given domain, e.g. movie reviews, and offered as a prediction service, e.g. as a sentiment analysis API. Take now a practitioner that wants to apply this API service in a new domain, i.e. restaurant reviews. To which extent is the API going to work in the new domain? Can the user trust the predictions of a model trained for movies? Will it be necessary to fine-tune the model for predicting reviews of restaurants?

One can foresee that fine-tuning the API for each new domain is not always possible. In the case of a third party API, one even might not have access to the model internals for such a task. In this scenario, the practitioner would most probably need some metric that evaluates the success performance, which may be directly related to the classification accuracy in the new domain. However, because domains may not be directly comparable, accuracy may not be a sufficient evaluation metric, demanding additional measures to evaluate the quality of the predictions when applying this API in the new domain.

To address this issue, probabilistic models are a natural option for evaluating confidence or uncertainty in the prediction. Given a classification problem, the output of these probabilistic models is the probability distribution of the labels given the input pattern. The analysis of this output distribution may suffice to derive prediction confidence in many cases. For example, the analysis of the entropy of the distribution or the spread of the distribution is an evident indicator of confidence. While sharper low-spread distributions suggest that the prediction has high confidence, flat distributions advocate for the opposite.

However, the former reasoning may fail in some cases when applying the black-box prediction to a new data domain. Changes in the vocabulary or the presence of new constructs (e.g. maybe the new text includes words that are not present in movie reviews but are representative of the restaurant domain, such as the words “yummy” or “tasty”) may mislead the interpretation of the results. For example, giving high confidence to a wrong classification because of the lack of data support for that case. In this situation, it is important “to know what the model does not know”. Fortunately, this concept is captured by the notion of uncertainty.

Previous works have analysed the role of uncertainty in deep learning, including those related to Bayesian neural networks [4], by considering the weights of

the network as random variables, thus obtaining not a single model but an ensemble of them which enables the analysis of the variance of the resulting predictions. Alternatively, approximations like Variational Dropout [2] do not require to modify the model to analyse the associated uncertainty. Having a way to measure the uncertainty when applying deep learning methods to natural language processing tasks achieves three goals, namely, to increase the explainability and transparency of the models, to measure the level of confidence of the predictions, and to improve the accuracy of the models [1].

In this article, we address the problem of estimating uncertainty from a black-box model and apply it to the problem of sentiment analysis to improve its accuracy by leveraging the uncertainty in the predictions. The task of sentiment analysis is one of the most popular applications in the field of NLP. The analysis of the sentiment of textual reviews is critical to have a better understanding of customers in fields such as e-commerce [19] or tourism [20], and like other NLP tasks, Sentiment Analysis is having a sweet moment these days thanks to the advances produced in deep learning. Beyond the text feature engineering used in the past, with the advent of deep learning, embedding models like word2vec or Glove, or more recently transfer learning based models like ELMO [17], GPT [18] or BERT [16] are obtaining state of the art results that improve previous approaches by far.

The main contributions of the present work include:

- The analysis of aleatoric heteroscedastic uncertainty in deep learning classification methods, with special emphasis in NLP classification tasks.
- The development of a wrapper methodology for computing uncertainty from black-box classification models.
- The application of the uncertainty measure in a rejection framework for improving the quality of classification in a sentiment analysis domain.

The structure of the rest of the paper is as follows: Sect. 2 analyses the related work, with a particular focus in uncertainty in deep learning, sentiment analysis and rejection methods. Section 3 exposes the method proposed in the present work and Sect. 4 shows the results obtained after applying the proposed method in a practical situation. Finally, Sect. 5 sums up the work presented and points to future research directions.

2 Related Work

2.1 Uncertainty in Deep Learning

When referring to uncertainty we usually have to distinguish among the following types of uncertainty:

- **Epistemic uncertainty** corresponds to the uncertainty originated by the model. It can be explained as to which extent is our model able to describe the distribution that generated the data. In this case, we can talk of two different types of uncertainties caused by whether the model has been trained with

enough data, so it has been able to learn the full distribution of the data, or whether the expressiveness of the model can capture the complexity of the distribution. When using an expressive enough model, this type of uncertainty can be reduced by showing more samples during the training phase.

– **Aleatoric uncertainty** refers to the inherent uncertainty coming from the data generation process, i.e. due to measurement noise or inherent ambiguity of the data. Adding more data to the training process will not reduce this kind of uncertainty. There are two types of aleatoric uncertainty according to the following assumptions:
  - **Homocedastic uncertainty** measures the level of noise derived from the measurement process. This uncertainty remains constant for all the data.
  - **Heteroscedastic uncertainty** measures the level of uncertainty caused by the data. For example, in the case of NLP, this can be explained by the ambiguity of some words or sentences.

Adopting a Bayesian framework, we can formalise the notion of uncertainty as follows. Let us have a training dataset, D, composed by pairs of data points and labels, $D = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$. The goal in inference consists of estimating the label probability y^* for a new data point x^* given D, i.e. $p(y^*|x^*, D)$. By marginalizing this last quantity[1] with respect to model parameters $\boldsymbol{w}$ we obtain,

$$p(y^*|x^*, D) = \int_w p(y^*|w, x^*)p(w|D)dw \tag{1}$$

In Eq. 1, one can see that the distribution of the output depends on two terms. The first one depends on the application of the model to the input query data. The second term measures how the model may vary depending on the training data. From this definition, we can derive that the first term is modelling aleatoric uncertainty, as it measures how the output is affected by the input data given a model, and the second term is modelling the epistemic uncertainty as it measures the uncertainty induced by the parameters of the model.

Aleatoric Heteroscedastic Uncertainty. Because we are assuming a black-box model, in this work, we are only concerned with the estimation of heteroscedastic uncertainty. Thus, we consider that we have a fixed deterministic black-box model $f^w(x)$ with undisclosed non-trainable parameters w. Our goal is to compute the variability of the term $p(y^*|w, x^*)$ in Eq. 1. For the sake of simplicity, let us assume that this conditional distribution follows a normal distribution, i.e. $y^*|w, x^* \sim \mathcal{N}(f^w(x^*), \sigma^2(x^*))$, where $f^w(x^*)$ is the black-box model evaluated at the data point x^*, and $\sigma^2(x^*)$ is a function of the input data that models the variance for that data point. In regression tasks, applying this

[1] We additionally assume an inductive learning approach to modelling where $p(y^*|w, x^*, D) = p(y^*|w, x^*)$ and the model parameters are independent from the test data, i.e. $p(w|D, x^*) = p(w|D)$.

approximation to the log-likelihood adds a term to the loss that depends on $\sigma(x)$ [3]. However, in classification tasks, this approximation is not as straightforward as in regression.

We consider the scenario where $f^w(x^*)$ is implemented by a deep neural network. In general, the computation of aleatoric heteroscedastic uncertainty for fully trainable networks considers the introduction of a stochastic layer to represent the output logits space. In the case that the output logits, u, of the classification model are modelled as normal distributed variables with a diagonal variance term they can be written as follows,

$$u \sim \mathcal{N}(f^w(x^*), diag(\sigma^2(x^*))) \tag{2}$$

$$p = softmax(u) \tag{3}$$

$$y \sim Categorical(p) \tag{4}$$

by reparameterizing the logits, u, we obtain,

$$u = f^{\boldsymbol{w}}(x^*) + \sqrt{diag(\sigma^2(x^*))} \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1) \tag{5}$$

In its simple approach, working with this stochastic layer requires of its sampling. In general, we would need to compute the expected value by applying Monte Carlo sampling, obtaining,

$$\mathbb{E}[p] = \frac{1}{M} \sum_{m=1}^{M} softmax(u_m) \tag{6}$$

When applied to a cross-entropy loss allows us to obtain the loss we will use for the wrapper, i.e.

$$\mathcal{L}(W) = \frac{1}{N} \sum_{i=1}^{N} -\frac{1}{C} \sum_{c=1}^{C} y_{i,c} \log\left(p_{i,c}\right) = \tag{7}$$

$$\frac{1}{N} \sum_{i=1}^{N} -\frac{1}{C} \sum_{c=1}^{C} y_{i,c} \log \frac{1}{M} \sum_{m=1}^{M} softmax(u_m) \tag{8}$$

Where N is the number of examples, C is the number of classes and M is the number of Monte Carlo samples.

2.2 Rejection Methods

As outlined in the introduction, the purpose of this work is to take advantage of the uncertainty associated with classification methods, especially when using pre-trained models in new domains, to improve the quality of the resulting predictions. By using this uncertainty as a rejection metric, one can choose whether to trust or not a prediction obtained.

In the literature, we can find many approaches for classification with rejection: from the initial work presented by [21] where they minimise the classification risk by setting a threshold for rejection to more recent works where

they embed the rejection option in the classifier [22]. The problem with these approaches is the fact that they need to modify the original classifier to include the rejection, which goes against the requirement of the present work of having a frozen classifier.

Moreover, many metrics used for rejection has shown some limitations. Metrics like accuracy or F1-score are used for obtaining accuracy-rejection curves (ARC) [23] or 3D ROC (receiver operating characteristic) [24]. They compare different classifiers through an analysis of the behaviour of the respective curves. The problem with this approach is that it is not able to look for the optimal rejection rate by comparing the performance of the classifiers working with different rejection rates.

In [5], they describe a set of three performance measures: non-rejected accuracy, classification quality and rejection quality. These measurements allow us to analyse different rejection metrics for a given classifier while considering different aspects of the classification, miss-classification and rejection. In the present work, we take advantage of these performance measures to evaluate the proposed rejection heuristic based on the prediction uncertainty.

3 Uncertainty Measures from Black-Box Models

In this section, we introduce the wrapping technique for obtaining aleatoric uncertainty from a black-box model. Following this, we further introduce a heuristics for measuring uncertainty from the resulting values.

3.1 A Wrapper for Computing Aleatoric Heteroscedastic Uncertainty

Given a black-box model, the goal of the wrapper is to endow this model with an aleatoric heteroscedastic uncertainty layer. For all purposes, the black box can not be modified as it is frozen and we do not have access to the internals of the model. However, we require an entry point that allows connecting the black box to the wrapper. In this work, we consider that the black box gives access to the pre-normalized logits (before the softmax) of the last layer[2]. This is a very mild requirement. It is agnostic to the particular architecture of the classifier since it does not interfere with the model internals and all models display this same structure making the wrapper generic for any architecture.

Figure 1 shows an illustration of the proposed wrapper architecture. The wrapper architecture aims at computing aleatoric uncertainty as expressed in Eq. 5. In that equation we distinguish two components: the original black-box model, shown in blue/gray in Fig. 1, corresponding to $f^w(x)$ and the trainable wrapper architecture, shown in orange/light gray in Fig. 1, that will gives us $\sigma(x)$. The first component corresponds to the logits of the original classifier, what we call μ-logits, and is the result of the last layer when applying the

[2] This is consistent with the former use of the notation of the model, $f^w(x)$.

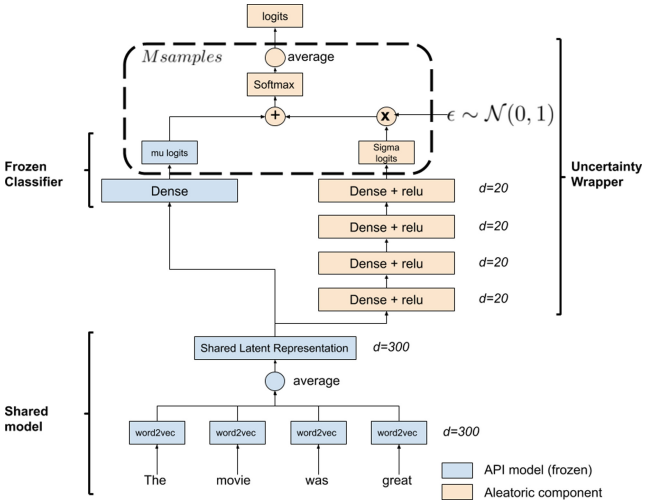

Fig. 1. Architecture of the full aleatoric model. The blue components correspond to the original classifier as exposed by the API. In orange, there is the aleatoric trainable part of the model. (Color figure online)

original classifier to the inputs. The second component, $\sigma(x)$, is the aleatoric part of the equation and will capture the variance of the predictions. We train this component using as input the same latent representation resulting from applying the frozen model to the training examples. The result of this component is what we call the σ-logits. It is worth mentioning that the composition of the μ-logits and σ-logits define a normal random variable layer. The evaluation of the network using random variables requires of its sampling. Note that the same input might generate different instantaneous predictions. Through this sampling process, we use the reparametrization trick, as described in Eq. 5 to be able to propagate gradients through the wrapper layers and infer the output distributions and statistics. In particular, by analysing the variance of these predictions, we may infer the aleatoric uncertainty that can be used as a heuristic for rejecting uncertain predictions, as shown in the next sections.

3.2 Uncertainty Heuristics

Using the former wrapper we have access to the variance of the logits. In this section, we discuss how to compute uncertainty scores based on that measure. In regression systems, it is usual to approximate the output with a Gaussian random variable. In this setting, the uncertainty score can be identified with the associated standard deviation. However, in classification systems, obtaining a single estimate is harder as the random variable is applied to the last layer logits.

For the case of classification, we find in literature [7] different ways for transforming the variance of the logits into an uncertainty score: variation ratios [8],

predictive entropy [9], and mutual information [9]. The first heuristic, variation ratios, evaluates the variability of the predictions made when sampling different predictions using the aleatoric model, sort of a measure of the dispersion of the predictions around their mode. The second heuristic, predictive entropy, is based on the information theory and evaluates the average amount of information contained in the predictive distribution. Those results with lower entropy values will correspond with confident predictions, whereas a high entropy will correspond with high uncertainty. In our case, we evaluate variation ratios and predictive entropy as both can be obtained directly analyzing the output layer of the black-box model, discarding mutual information because of the complexity of its calculation.

Finally, we use the uncertainty score to reject those samples that are more uncertain, increasing thus the accuracy of the classifier for the sentiment analysis task.

4 Use Case and Results

Use Case and Datasets Description: We illustrate the feasibility of the presented method in the following use case: Consider a sentiment analysis API. In our case, this API is trained using one domain from the four used in this article. The resulting model is frozen and the internals not accessible to us except for the seam in the output layer logits for the wrapper injection. We want to apply this API to a new sentiment analysis domain. Transfer learning is not an option since we rely on a black-box non-trainable model. We will use the API in the new problem directly applying the predictions obtained. Using the uncertainty wrapper, we expect to identify where these predictions are not reliable in the product domain and apply a rejection rule in those cases.

The details on the datasets used are the following:

- Stanford Sentiment Treebank [10], SST-2, binary version where the purpose of the tasks is to classify a movie review in two categories: a positive or negative review. The dataset is split in 65,538 test samples, 872 for validation and 1,821 for testing.
- Yelp challenge 2013 [12], the goal is to classify reviews about Yelp venues where their users rated them using 1 to 5 stars. To be able to reuse a classifier trained with the SST-2 problem, we transform the Yelp dataset from a multiclass set to a binary one by grouping the ratings below three as a negative review, and positive otherwise. The dataset is split in 186,189 test samples, 20,691 for validation and 22,991 for testing.
- Amazon Multi-Domain Sentiment dataset contains product reviews taken from Amazon.com from many product types (domains) [25]. As in Yelp, the dataset consists on ratings from 1 to 5 stars that we label as positive for those with values greater or equal to 3, and negative otherwise, split into train, validation and test datasets. We use two of the domains available: music (10,595/993/2,621 examples) and computer and video games (406,035/45,093/112,794 examples).

We consider four scenarios:

- **Scenario 1:** The API is trained on Yelp venues, and applied to movie reviews.
- **Scenario 2:** The API is trained on movie reviews using SST-2, and applied to Yelp venues.
- **Scenario 3:** The API is trained on the Amazon music domain, and applied to computer and video games.
- **Scenario 4:** The API is trained on computer and video games, and applied on music reviews.

Simulating the API: In all cases, we trained sentiment analysis models using word2vec [11] to vectorize the textual reviews, as described in [6]. The idea is to obtain a sentence representation for each review by averaging the word2vec embedding of each word into a 300 summarizing vector. Using this sentence representation, we apply a classifier based on an ANN with a softmax output layer to predict whether the review is positive or negative. We simulate the black-box API by training a classifier with only one softmax layer using the Keras framework.

Uncertainty Wrapper Architecture: The wrapper architecture, shown in Fig. 1, is composed by an input layer that uses the 300 dimension vector, and four hidden layers with 20 units each. Lastly, the output layer uses a softplus activation to ensure that the outputs are positive. The wrapper is trained for 100 epochs using the Adam optimiser with a learning rate of 2e−4.

Pre-processing: In all datasets, we pre-process the textual input that represents the corresponding reviews by tokenising it, removing HTML symbols, numbers and punctuation, also removing English stop words and contractions.

Evaluation Metrics: We apply the performance measures described in [5] to analyse how the rejection metric affects the quality of the resulting classifier, i.e.

- Non-rejected Accuracy: measures the ability of the classifier to classify non-rejected samples accurately.
- Classification Quality: measures the ability of the classifier with rejection to accurately classify non-rejected samples and to reject misclassified samples.
- Rejection Quality: measures the ability to concentrate all misclassified samples onto the set of rejected samples.

By using these performance metrics, we compare the uncertainty derived from the wrapper as a rejection heuristic with the baseline derived from directly using the entropy of the classification output logits. We compute the uncertainty heuristics based on predictive entropy and variation ratios of the aleatoric wrapper.

Experimental Results. Figure 2 displays the three rejection measures in all scenarios comparing the uncertainty derived from the predictive entropy of the output labels with the predictive entropy and variation ratios obtained using the wrapper. Observe that the aleatoric predictive entropy performs better in

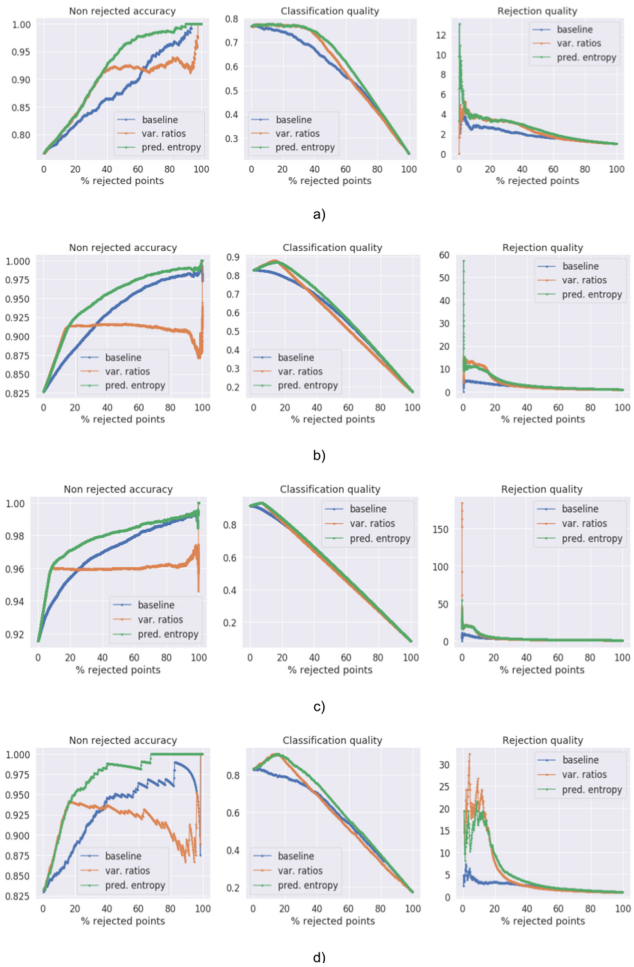

Fig. 2. Comparison of the three rejection measures where we compare the predictive entropy obtained using only the predictions of the original classifier with the predictive entropy and variation ratios obtained with the wrapper for the four experiments: (a) Yelp to SST-2; (b) SST-2 to Yelp; (c) computer reviews to music; (d) music reviews to computers.

the three performance measures. Variation ratios performs well for classification and rejection quality, but it fails to detect misclassified examples as the rejection point increases. Comparing aleatoric and non-aleatoric predictive entropies, both seem to capture wrong predictions, but the aleatoric one always outperforms the baseline in all cases.

Moreover, analyzing the non-rejected accuracy when using the aleatoric predictive entropy, the best performance is usually obtained with rejection ratios between 15% and 35%. Further, Table 1 shows that the proposed method out-

Table 1. Accuracy obtained by training an standalone classifier, applying the API and the proposed wrapper for each domain

	Standalone	API	Wrapper (20%)
Yelp	89.2%	77.1%	**92.2%**
SST-2	82.5%	82.2%	**83.7%**
Computer	85.2%	91.8%	**95.3%**
Music	93.1%	83.1%	**97.2%**

performs the accuracy resulting accuracy from training a standalone classifier using the target dataset directly. It is worth noting that we do not need a large amount of data for getting reliable uncertainty estimations. In particular, the Amazon music domain is composed of only 10k samples. The behaviour when exchanging the domains is not symmetric. We have observed that niche domains tend to display larger values of estimated sigmas when applied to more general domains. Effectively resulting in larger uncertainty estimations.

5 Conclusions

We present a method that leverages the aleatoric uncertainty to obtain a rejection metric for pre-trained classification systems. We illustrate the utility of this metric applying it to a sentiment analysis problem in four different scenarios, showing how practitioners can benefit from this method to discard those predictions that present a higher uncertainty, increasing, therefore, the quality of the prediction system.

For future work we plan to include the epistemic uncertainty, applying the method to other tasks, like language models, or work on the explainability of the uncertainty and how to identify which elements, in this case, words, are more relevant to the uncertainty obtained.

Acknowledgements. This work has been partially funded by the Spanish projects TIN2016-74946-P and TIN2015-66951-C2 (MINECO/FEDER, UE), and by AGAUR of the Generalitat de Catalunya through the Industrial PhD grant.

References

1. Xiao, Y., Wang, W.Y.: Quantifying Uncertainties in Natural Language Processing Tasks. arXiv preprint arXiv:1811.07253 (2018)
2. Gal, Y., Ghahramani, Z. Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In: International Conference on Machine Learning (2016)
3. Kendall, A., Gal, Y. What uncertainties do we need in Bayesian deep learning for computer vision? In: Advances in Neural Information Processing Systems (2017)

4. Hernández-Lobato, J.M., Adams, R.: Probabilistic backpropagation for scalable learning of Bayesian neural networks. In: International Conference on Machine Learning (2015)
5. Condessa, F., et al.: Performance measures for classification systems with rejection. Pattern Recognit. **63**, 437–450 (2017)
6. Liu, H.: Sentiment Analysis of Citations Using Word2vec. CoRR abs/1704.00177, July 2017
7. Gal, Y.: Uncertainty in deep learning. Ph.D. thesis, University of Cambridge (2016)
8. Freeman, L.G.: Elementary Applied Statistics. Wiley, Hoboken (1965)
9. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**(3), 379–423 (1948)
10. Socher, R., et al.: Recursive deep models for semantic compositionality over a sentiment Treebank. In: Proceedings of the 2013 EMNLP (2013)
11. Mikolov, T., et al.: Efficient estimation of word representations in vector space. In: Proceedings of Workshop at ICLR (2013)
12. Yelp Dataset Challenge. Yelp dataset challenge (2013)
13. Ranganath, R., Gerrish, S., Blei, D.: Black box variational inference. Artif. Intell. Stat. 814–822 (2014)
14. Naesseth, C.A., et al.: Variational sequential Monte Carlo. arXiv preprint arXiv:1705.11140 (2017)
15. Gal, Y., Ghahramani, Z.: A theoretically grounded application of dropout in recurrent neural networks. In: Advances in Neural Information Processing Systems (2016)
16. Devlin, J., et al.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
17. Peters, M.E., et al.: Deep contextualized word representations. arXiv preprint arXiv:1802.05365 (2018)
18. Radford, A., et al.: Improving language understanding by generative pre-training (2018)
19. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Found. Trends® Inf. Retr. **2**(1–2), 1–35 (2008)
20. Meehan, K., et al.: Context-aware intelligent recommendation system for tourism. In: 2013 IEEE International Conference on PERCOM Workshops. IEEE (2013)
21. Chow, C.K.: On optimum recognition error and reject tradeoff. IEEE Trans. Inf. Theory **16**(1), 41–45 (1970)
22. Yuan, M., et al.: Classification methods with reject option based on convex risk minimization. J. Mach. Learn. Res. **11**, 111–130 (2010)
23. Nadeem, M., et al.: Accuracy-rejection curves (ARCs) for comparing classification methods with a reject option. Mach. Learn. Syst. Biol. **8**, 65–81 (2010)
24. Landgrebe, T., et al.: The interaction between classification and reject performance for distance-based reject-option classifiers. Pattern Recognit. Lett. **27**(8), 908–917 (2006)
25. Blitzer, J., et al.: Biographies, bollywood, boom-boxes and blenders: domain adaptation for sentiment classification. In: ACL (2007)