




Improving Usual Naive Bayes Classifier Performances with Neural Naive Bayes based Models

Elie Azeraf^{1,2}^a, Emmanuel Monfrini²^b and Wojciech Pieczynski²^c

¹Watson Department, IBM GBS, Avenue de l'Europe, Bois-Colombes, France

²SAMOVAR, CNRS, Telecom SudParis, Institut Polytechnique de Paris, Evry, France

Keywords: Naive Bayes, Bayes Classifier, Neural Naive Bayes, Pooled Markov Chain, Neural Pooled Markov Chain.

Abstract: Naive Bayes is a popular probabilistic model appreciated for its simplicity and interpretability. However, the usual form of the related classifier suffers from two significant problems. First, as caring about the observations' law, it cannot consider complex features. Moreover, it considers the conditional independence of the observations given the hidden variable. This paper introduces the original Neural Naive Bayes, modeling the classifier's parameters induced from the Naive Bayes with neural network functions. This method allows for correcting the first default. We also introduce new Neural Pooled Markov Chain models, alleviating the conditional independence assumption. We empirically study the benefits of these models for Sentiment Analysis, dividing the error rate of the usual classifier by 4.5 on the IMDB dataset with the FastText embedding, and achieving an equivalent F_1 as RoBERTa on TweetEval emotion dataset, while being more than a thousand times faster for inference.

1 INTRODUCTION


We consider the hidden random variable X , taking its values in the discrete finite set $\Lambda_X = \{\lambda_1, \dots, \lambda_N\}$, and the observed random variables $Y_{1:T} = (Y_1, \dots, Y_T), \forall t, Y_t \in \Omega_Y$. The Naive Bayes is a probabilistic model considering these variables. This probabilistic model is defined with the following joint law:


$$p(X, Y_{1:T}) = p(X) \prod_{t=1}^T p(Y_t|X). \quad (1)$$


It can be represented in Figure 1. It is especially appreciated for its simplicity, its interpretability, and it is one of the most famous probabilistic graphical models (Koller and Friedman, 2009; Wainwright and Jordan, 2008).

The Bayes classifier (Devroye et al., 2013; Duda et al., 2006; Fukunaga, 2013) of the Maximum A Posteriori (MAP) can be written as follows, with the realization $y_{1:T}$ of $Y_{1:T}$ (we use the shortcut notation $p(X = x) = p(x)$):

$$\phi(y_{1:T}) = \arg \max_{\lambda_i \in \Lambda_X} p(X = \lambda_i|y_{1:T}) \quad (2)$$

^a <https://orcid.org/0000-0003-3595-0826>

^b <https://orcid.org/0000-0002-7648-2515>

^c <https://orcid.org/0000-0002-1371-2627>

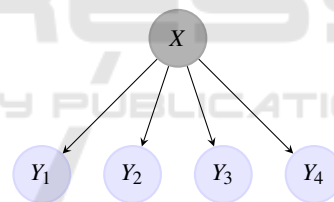


Figure 1: Probabilistic oriented graph of the Naive Bayes.

Therefore, the MAP classifier induced from the Naive Bayes is based on the posterior law, $\forall \lambda_i \in \Lambda_X, p(X = \lambda_i|y_{1:T})$, usually written as follows:

$$\begin{aligned} p(X = \lambda_i|y_{1:T}) &= \frac{p(X = \lambda_i, y_{1:T})}{\sum_{\lambda_j \in \Lambda_X} p(X = \lambda_j, y_{1:T})} \\ &= \frac{p(X = \lambda_i) \prod_{t=1}^T p(Y_t|X = \lambda_i)}{\sum_{\lambda_j \in \Lambda_X} p(X = \lambda_j) \prod_{t=1}^T p(Y_t|X = \lambda_j)} \\ &= \frac{\pi(i) \prod_{t=1}^T b_i^{(t)}(y_t)}{\sum_{\lambda_j \in \Lambda_X} \pi(j) \prod_{t=1}^T b_j^{(t)}(y_t)} \end{aligned}$$

with, $\forall \lambda_i \in \Lambda_X, y \in \Omega_Y, t \in \{1, \dots, T\}$:

- $\pi(i) = p(X = \lambda_i)$;
- $b_i^{(t)}(y) = p(Y_t = y | X = \lambda_i)$.

Thus, the classifier is:

$$\phi^{NB}(y_{1:T}) = \arg \max_{\lambda_i \in \Lambda_X} \left(\frac{\pi(i) \prod_{t=1}^T b_i^{(t)}(y_t)}{\sum_{\lambda_j \in \Lambda_X} \pi(j) \prod_{t=1}^T b_j^{(t)}(y_t)} \right) \quad (3)$$

We consider for all this paper the stationary case for all models, i.e., the different parameters are not depending on time t . Thus, for the Naive Bayes, we set $b_i^{(t)}(y) = b_i(y)$.

The classifier induced from the Naive Bayes for classification with supervised learning always uses the form (3), depending on the parameters π and b (Jurafsky and Martin, 2009; Ng and Jordan, 2002; Metsis et al., 2006; Liu et al., 2013). It is applied in many applications, such as *Sentiment Analysis* or *Text Classification* (Jurafsky and Martin, 2009; Kim et al., 2006; McCallum et al., 1998).

However, it is mainly criticized for two major drawbacks (Ng and Jordan, 2002; Sutton and McCallum, 2006). First, through the parameter b , it cares about the observations' law. It implies that one cannot consider complex features with the usual Naive Bayes classifier. Indeed, assuming that $Y_t \in \mathbb{R}^d$, and modeling b by a Gaussian law, the number of parameters to learn is equal to:

$$\underbrace{d}_{\text{for the mean}} + \underbrace{\frac{d(d+1)}{2}}_{\text{for the covariance matrix}}$$

If it is tractable for relatively small values of d , it quickly becomes intractable when d increases. For example, with Natural Language Processing (NLP), it is common to convert words to numerical vectors of size 300 (Pennington et al., 2014), 784 (Devlin et al., 2019), or even 4096 (Akbik et al., 2018), which becomes impossible to estimate. This process is called *word embedding* and is mandatory to achieve relevant results. One can suppose the independence between the different components, also called features, of Y_t , resulting in estimating only d parameters for the covariance matrix, but this process achieves poor results. We select the Gaussian law as an example, but this problem happens for every law. This "feature problem" is even more significant when the features belong to a discrete space, and approximation methods remain limited (Brants, 2000; Azeraf et al., 2021b).

Another main drawback of the Naive Bayes concerns the conditional independence assumption of the

observed random variables. It implies not considering the order of the different observations with the classifier induced from the stationary Naive Bayes model.

In this paper, inspired by the writing of the classifier in (Azeraf et al., 2021a) and the Hidden Neural Markov Chain in (Azeraf et al., 2021c), we propose the Neural Naive Bayes. This model consists in defining the classifier induced from the Naive Bayes written in a discriminative manner with neural networks functions (LeCun et al., 2015; Ian Goodfellow and Courville, 2016). This neural model corrects the first drawback of the usual Naive Bayes classifier, as it can consider complex features of observations. Moreover, we propose the Neural Pooled Markov Chains, which are neural models assuming a conditional dependence of the observations given the hidden random variables, modeling them as a Markov chain. Therefore, they also correct the second drawback of the usual Naive Bayes classifier. Finally, we empirically study these different innovations' contributions to the *Sentiment Analysis* task.

This paper is organized as follows. In the next section, we present the classifier of the Naive Bayes written discriminatively, i.e., which does not use the observations' law, the Pooled Markov Chain (Pooled MC) and Pooled Markov Chain of order 2 (Pooled MC2) models and their classifiers. The third section presents a different way to compute the classifiers, estimating parameters with neural network functions. Then, we empirically evaluate the contributions of our neural models applied to *Sentiment Analysis*. Conclusion and perspectives lie at the end of the paper.

To summarize our contributions, we present (i) three original neural models based on probabilistic models, and (ii) we show their efficiency compared with the usual form of the Naive Bayes classifier for *Sentiment Analysis*.

2 NAIVE BAYES AND POOLED MARKOV CHAINS

2.1 The Classifier Induced from the Naive Bayes Written Discriminatively

A classifier is considered "written generatively" if its form uses some $p(Y_A | X_B)$, with A, B non-empty subsets of the observed and hidden variable sets. If it is not written generatively, a classifier is written discriminatively. These notions rejoin the usual ones about generative and discriminative classifiers (Ng and Jordan, 2002; Jebara, 2012). For example, (3)

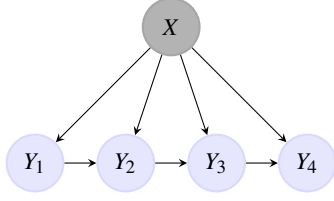


Figure 2: Probabilistic oriented graph of the Pooled Markov Chain.

is the classifier induced from the Naive Bayes written generatively, as its form uses $b_i(y_t) = p(Y_t = y_t | X_t = \lambda_i)$.

(Azeraf et al., 2021a) presents how to write the classifier induced from the Naive Bayes written discriminatively:

$$\phi^{NB}(y_{1:T}) = \arg \max_{\lambda_i \in \Lambda_X} \text{norm} \left(\pi(i)^{1-T} \prod_{t=1}^T L_{y_t}(i) \right) \quad (4)$$

with, $\forall \lambda_i \in \Lambda_X, t \in \{1, \dots, T\}, y \in \Omega_Y$:

$$L_y(i) = p(X = \lambda_i | Y_t = y),$$

and the norm function defined as, $\forall x \in \mathbb{R}^N$:

$$\text{norm} : \mathbb{R} \rightarrow \mathbb{R}, x_i \rightarrow \frac{x_i}{\sum_{j=1}^N x_j}.$$

2.2 Pooled Markov Chain

We introduce the Pooled Markov Chain model, considering the same random variables as the Naive Bayes. It is defined with the following joint law:

$$p(X, Y_{1:T}) = p(X)p(Y_1|X) \prod_{t=1}^{T-1} p(Y_{t+1}|X, Y_t) \quad (5)$$

This model generalizes the Naive Bayes, insofar as the latter is a special case of it, supposing that the observed random variables follow a Markov chain given the hidden variable, while the Naive Bayes supposes the conditional independence. It is represented in Figure 2.

The classifier induced from the Pooled MC written discriminatively is defined as:

$$\phi^{MC}(y_{1:T}) = \arg \max_{\lambda_i \in \Lambda_X} \text{norm} \left(L_{y_1}^{MC,1}(i) \prod_{t=1}^{T-1} \frac{L_{y_t, y_{t+1}}^{MC,2}(i)}{L_{y_t}^{MC,1}(i)} \right). \quad (6)$$

with the following parameter of the stationary Pooled MC, $\forall \lambda_i \in \Lambda_X, y_1, y_2 \in \Omega_Y$:

- $L_{y_1}^{MC,1}(i) = p(X = \lambda_i | Y_t = y_1)$;
- $L_{y_1, y_2}^{MC,2}(i) = p(X = \lambda_i | Y_t = y_1, Y_{t+1} = y_2)$.

The proof is given in the appendix.

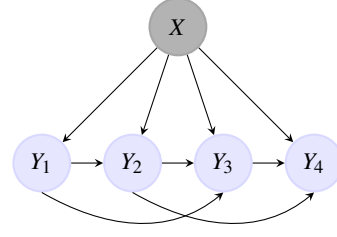


Figure 3: Probabilistic oriented graph of the Pooled Markov Chain of order 2.

2.3 Pooled Markov Chain of Order 2

The Pooled Markov Chain of order 2, represented in Figure 3 is defined with the joint law:

$$p(X, Y_{1:T}) = p(X)p(Y_1|X)p(Y_2|X, Y_1) \prod_{t=1}^{T-2} p(Y_{t+2}|X, Y_t, Y_{t+1}) \quad (7)$$

This probabilistic model generalizes the two others, considering the observations follow a Markov chain of order 2 given the hidden variable.

The classifier induced from the Pooled MC2 written discriminatively is defined as:

$$\phi^{MC2}(y_{1:T}) = \arg \max_{\lambda_i \in \Lambda_X} \text{norm} \left(L_{y_1, y_2}^{MC2,1}(i) \prod_{t=1}^{T-1} \frac{L_{y_t, y_{t+1}, y_{t+2}}^{MC2,2}(i)}{L_{y_t, y_{t+1}}^{MC2,1}(i)} \right). \quad (8)$$

with the following parameter of the stationary Pooled MC2, $\forall \lambda_i \in \Lambda_X, y_1, y_2, y_3 \in \Omega_Y$:

- $L_{y_1, y_2}^{MC2,1}(i) = p(X = \lambda_i | Y_t = y_1, Y_{t+1} = y_2)$;
- $L_{y_1, y_2, y_3}^{MC2,2}(i) = p(X = \lambda_i | Y_t = y_1, Y_{t+1} = y_2, Y_{t+2} = y_3)$.

The proof is also given in the appendix.

3 NEURAL NAIVE BAYES BASED MODELS

3.1 Neural Naive Bayes

We consider the classifier induced from the Naive Bayes written discriminatively (4). We assume the functions π and L , allowing to define these classifiers, are strictly positive, and $\Omega_Y = \mathbb{R}^d, d \in \mathbb{N}^*$. (4) can be written as follows:

$$\phi^{NB}(y_{1:T}) = \arg \max_{\lambda_i \in \Lambda_X} \text{norm} \left(\pi(i)^{1-T} \prod_{t=1}^T L_{y_t}(i) \right)$$

$$\begin{aligned}
 &= \arg \max_{\lambda_i \in \Lambda_X} \text{norm} \left(\pi(i) \prod_{t=1}^T \frac{L_{y_t}(i)}{\pi(i)} \right) \\
 &= \arg \max_{\lambda_i \in \Lambda_X} \text{softmax} \\
 &\quad \left(\log(\pi(i)) + \sum_{t=1}^T \log \left(\frac{L_{y_t}(i)}{\pi(i)} \right) \right). \quad (9)
 \end{aligned}$$

with the softmax function defined as, $\forall x \in \mathbb{R}^N$:

$$\text{softmax} : \mathbb{R} \rightarrow \mathbb{R}, x_i \rightarrow \frac{\exp(x_i)}{\sum_{j=1}^N \exp(x_j)}.$$

Moreover, we suppose, $\forall i \in \{1, \dots, N\}$, that the influence of $\log(\pi(i))$ is negligible in (9), which is true if the classes of X have the same probability, or if T is big enough.

We set NN^{NB} a neural network function with y_t as input and the output of size N . We define it as follows:

$$NN^{NB}(y_t)_i = \log \left(\frac{L_{y_t}(i)}{\pi(i)} \right)$$

with $NN^{NB}(y_t)_i$ the i -th component of the vector $NN^{NB}(y_t)$.

Therefore, (4) is approximated with:

$$\phi^{NB}(y_{1:T}) \approx \arg \max_{\lambda_i \in \Lambda_X} \text{softmax} \left(\sum_{t=1}^T NN^{NB}(y_t)_i \right) \quad (10)$$

As (10) is the classifier induced from the Naive Bayes parametrized with neural networks, we call this application the Neural Naive Bayes.

3.2 Neural Pooled Markov Chains

We consider the Pooled MC model, and we assume the same hypothesis as above about the parameter functions. The classifier of the Pooled MC (6) can be written:

$$\begin{aligned}
 \phi^{MC}(y_{1:T}) &= \arg \max_{\lambda_i \in \Lambda_X} \text{softmax} \\
 &\quad \left(\log(L_{y_1}^{MC,1}(i)) + \sum_{t=1}^{T-1} \log \left(\frac{L_{y_t, y_{t+1}}^{MC,2}(i)}{L_{y_t}^{MC,1}(i)} \right) \right). \quad (11)
 \end{aligned}$$

As above, we suppose that $\log(L_{y_1}^{MC,1}(i))$ is negligible. We define a neural network function, NN^{MC} , with the concatenation of y_t and y_{t+1} as input, and an output of size N . It is used to model:

$$NN^{MC}(y_t, y_{t+1})_i = \log \left(\frac{L_{y_t, y_{t+1}}^{MC,2}(i)}{L_{y_t}^{MC,1}(i)} \right)$$

Therefore, (11) is approximated as follows:

$$\begin{aligned}
 \phi^{MC}(y_{1:T}) &\approx \arg \max_{\lambda_i \in \Lambda_X} \text{softmax} \\
 &\quad \left(\sum_{t=1}^{T-1} NN^{MC}(y_t, y_{t+1})_i \right) \quad (12)
 \end{aligned}$$

We called (12) the Neural Pooled Markov Chain (Neural Pooled MC) function.

In the same idea, the Neural Pooled Markov Chain of order 2 (Neural Pooled MC2) is defined with:

$$\begin{aligned}
 \phi^{MC2}(y_{1:T}) &\approx \arg \max_{\lambda_i \in \Lambda_X} \text{softmax} \\
 &\quad \left(\sum_{t=1}^{T-2} NN^{MC2}(y_t, y_{t+1}, y_{t+2})_i \right) \quad (13)
 \end{aligned}$$

with NN^{MC2} a neural function having the concatenation of y_t, y_{t+1} , and y_{t+2} as input, and an output of size N .

To go further, $\forall k \in \mathbb{N}^*$, we introduce the Neural Pooled Markov Chain of order k (Neural Pooled MC(k)):

$$\begin{aligned}
 \phi^{MCk}(y_{1:T}) &\approx \arg \max_{\lambda_i \in \Lambda_X} \text{softmax} \\
 &\quad \left(\sum_{t=1}^{T-k} NN^{MCk}(y_t, y_{t+1}, \dots, y_{t+k})_i \right) \quad (14)
 \end{aligned}$$

with NN^{MCk} a neural function having the concatenation of y_t to y_{t+k} as input, and an output of size N .

4 APPLICATION TO SENTIMENT ANALYSIS

Sentiment Analysis is an NLP task consisting of predicting the sentiment of a given text. In this section, we are going to apply the usual classifier induced from the Naive Bayes (3), the Neural Naive Bayes (10), the Neural Pooled MC (12), and the Neural Pooled MC2 (13) for *Sentiment Analysis*. The goal is to observe the improvement brought by our new classifiers compared to the usual one usually.

We use two different embedding methods: FastText (Bojanowski et al., 2017) and ExtVec (Komninos and Manandhar, 2016), allowing to convert each word in a given sentence to a vector of size 300. Therefore, $\Omega_Y = \mathbb{R}^{300}$ in this use-case.

Every parameter of neural models, $NN^{NB}, NN^{MC}, NN^{MC2}$, is a feedforward neural network with the adapted input size, a hidden layer of size 64 followed by a ReLU (Nair and Hinton, 2010) activation function, and an output layer of size N , the latter depending on the dataset.

Table 1: Accuracy errors of the usual classifier induced from the Naive Bayes and our proposed neural models on IMDB dataset with FastText and ExtVec embeddings.

	Usual Naive Bayes	Neural Naive Bayes	Neural Pooled MC	Neural Pooled MC2
FastText	49.49%	12.46% \pm 0.19	11.37% \pm 0.24	11.01% \pm 0.05
ExtVec	46.64%	13.40% \pm 0.34	12.68% \pm 0.17	12.34% \pm 0.14

Table 2: Accuracy errors of the usual classifier induced from the Naive Bayes and our proposed neural models on SST-2 dataset with FastText and ExtVec embeddings.

	Usual Naive Bayes	Neural Naive Bayes	Neural Pooled MC	Neural Pooled MC2
FastText	48.00%	15.32% \pm 0.20	14.50% \pm 0.17	14.10% \pm 0.26
ExtVec	49.80%	17.22% \pm 0.22	16.14% \pm 0.29	15.88% \pm 0.14

Table 3: F_1 scores of the usual classifier induced from the Naive Bayes and our proposed neural models on TweetEval dataset with FastText and ExtVec embeddings.

	Usual Naive Bayes	Neural Naive Bayes	Neural Pooled MC	Neural Pooled MC2
FastText	17.68	71.15 \pm 0.16	71.32 \pm 0.20	72.00 \pm 0.28
ExtVec	15.68	68.17 \pm 0.37	69.07 \pm 0.46	69.28 \pm 0.45

Table 4: F_1 scores of the usual classifier induced from the Naive Bayes and our proposed neural models on Financial Phrasebank dataset with FastText and ExtVec embeddings.

	Usual Naive Bayes	Neural Naive Bayes	Neural Pooled MC	Neural Pooled MC2
FastText	31.83 \pm 0.99	82.87 \pm 0.53	84.61 \pm 0.91	86.02 \pm 0.49
ExtVec	34.48 \pm 1.39	82.60 \pm 0.58	85.01 \pm 0.77	85.14 \pm 1.06

4.1 Dataset Description

We use four datasets for our experiments:

- the IMDB dataset (Maas et al., 2011), composed of movie reviews, with a train set of 25000 texts and a test set of the same size, with $\Lambda_X = \{Positive, Negative\}$.
- the SST-2 dataset (Socher et al., 2013) from GLUE (Wang et al., 2019), also composed of movie reviews, with a train set of 67349 texts, a test set of 1821 texts, and a validation set of 872 texts, with $\Lambda_X = \{Positive, Negative\}$;
- the TweetEval emotion dataset (Barbieri et al., 2020; Mohammad et al., 2018), composed with Twitter data, with a train set of 3257 texts, a test set of 1421 texts, and a validation set of 374 texts, with $\Lambda_X = \{Anger, Joy, Optimism, Sadness\}$;
- the Financial Phrasebank (FPB) dataset (Malo et al., 2014), which consists in English sentences from financial news. We select the "All agree" data of 2264 sentences with $\Lambda_X = \{Positive, Neutral, Negative\}$. We set aside 453 randomly for test set. To avoid biases, we do a different draw at every experiment.

All these datasets are freely available with the datasets library (Lhoest et al., 2021).

4.2 Parameter Learning and Implementation Details

On the one hand, the parameter π and b from (3) are learned with maximum likelihood estimation. For π , it consists of estimating it by counting the frequencies of the different patterns:

$$\pi(i) = \frac{N_i}{\sum_j N_j}$$

with N_i the number of times $X = \lambda_i$ in the training set. We model b with a gaussian law. As the estimation of the covariance matrix of size 300×300 for each class is intractable and result in computational problems when used, we select the vector's mean as the observation of each word. It estimates a mean and a variance of size 1 for each class. Therefore, for each $\lambda_i \in \Lambda_X$, $b_i \sim \mathcal{N}(\mu_i, \sigma_i)$, with μ_i and σ_i estimated by maximum likelihood.

On the other hand, parameters NN^{NB} , NN^{MC} , and NN^{MC2} are estimated with stochastic gradient descent algorithm using Adam (Kingma and Ba, 2014) optimizer with back-propagation (Rumelhart et al., 1986; LeCun et al., 1989) and Cosine Annealing gradient method (Loshchilov and Hutter, 2016). We minimize the Cross-Entropy loss function with a learning rate of 0.001 and a mini-batch size of 64 - selected after a K-fold cross-validation step. For the IMDB and

FPB datasets, where the validation set is not directly given, we construct it with 20% of the train set, drawn randomly. For each experiment, we train our model for 5 epochs (except for TwetEval, which requires 15 epochs, and FPB, requiring 25 epochs), and we keep the one achieving the best score on the validation set for testing. As it is a stochastic algorithm, every experiment with a Neural Naive Bayes based model is realized five times, and we report the mean and the 95% confidence interval. Every experiment is realized 10 times for FPB, even for the usual classifier, due to the test set drawing.

About the implementation details, all the codes are written in python. We use the Flair (Akbik et al., 2019) library for the word embedding methods, and PyTorch (Paszke et al., 2019) for neural functions and training. All experiments are realized with a CPU having 16Go RAM.

4.3 Results

We apply the usual classifier induced from Naive Bayes, the Neural Naive Bayes, the Neural Pooled MC, and the Neural Pooled MC2 to the four datasets. The accuracy errors are available in Table 1 and 2 for IMDB and SST-2, and the F_1 scores in Table 3 and 4 for TweetEval and FPB.

As expected, one can observe important improvements with the neural models related to the usual classifier induced from the Naive Bayes. Indeed, for example, with FastText embedding on the IMDB dataset, this error is divided by about 4.5 with the Neural Pooled MC2. It confirms the importance of considering complex features.

As one can observe in Table 1, 2, 3, and 4, increasing the Markov chain's order allows to improve results. It shows the effect of alleviating the conditional independence condition. This observation is expected as the model becomes more complex and extended. Indeed, for each $k \in \mathbb{N}$, the Pooled Markov Chain of order k is a particular case of the Pooled Markov Chain of order $k + 1$. To observe the limits of this case, we also apply the Neural Pooled MC(k), with $k \in \{3, 5, 7, 10\}$. All these models achieve slightly equivalent results as the Neural Pooled MC2, without significant improvements, showing the empirical limits of this method.

Moreover, we can compare our models' results with other popular ones. First, we precise that our models are lights, as training and inference time are very fast: about 5 minutes to train a model and about 1.5 milliseconds for inference. Therefore, we compare our results only with models reputed as "light" and not heavy models as the ones based on Trans-

former (Vaswani et al., 2017), or LSTM-CRF (Akbik et al., 2018). These light models present the benefit of being easy to serve for industrial purposes. Thus, about IMDB, our models are largely better than SVM with TF-IDF, Convolutional Neural Networks (Tang et al., 2015), or even the classic FastText algorithm for text classification (Joulin et al., 2017), which achieve accuracy errors of 59.5%, 62.5%, and 54.8%, respectively. About TweetEval, our models achieves better results than the BiLSTM (66.0), FastText (65.2), SVM (64.7), and even equivalent result as RoBERTa (Barbieri et al., 2020; Liu et al., 2019) (72.0), the latter being an heavy model based on the Transformers, requiring about 2 seconds for inference. We can make the same observation for FPB, with LSTM (0.74), LSTM with ELMO (0.77), and other models available in (Araci, 2019). These results show our proposed models' interest regarding the other popular ones. Therefore, they are a great alternative considering the training and inference time/performances threshold.

5 CONCLUSION AND PERSPECTIVES

This paper presents the original Neural Naive Bayes and Neural Pooled Markov Chain models. These models significantly improve the usual Naive Bayes classifier's performances, considering complex observations' features without constraints, modeling their parameters with Neural Network functions, and alleviating the conditional independence hypothesis. They allow to achieve great performances and simplicity, and present an original way to define the classifier induced from probabilistic models with neural network functions.

Moreover, these new neural models can be viewed as original pooling methods for textual data. Indeed, starting from word embedding methods, a usual way for text embedding consists of summing the embedding of the words of the text. Our methods, which benefit from being computationally light, propose a new way to do this document embedding process and can be included in any neural architecture. Therefore, going further in this direction can be a promising perspective.

REFERENCES

- Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., and Vollgraf, R. (2019). FLAIR: An Easy-To-Use Framework for State-Of-The-Art NLP. In *Proceed-*

- ings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), pages 54–59.
- Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual String Embeddings for Sequence Labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.
- Araci, D. (2019). FinBERT: Financial Sentiment Analysis with Pre-Trained Language Models. *arXiv preprint arXiv:1908.10063*.
- Azeraf, E., Monfrini, E., and Pieczynski, W. (2021a). Using the Naive Bayes as a Discriminative Model. In *Proceedings of the 13th International Conference on Machine Learning and Computing*, pages 106–110.
- Azeraf, E., Monfrini, E., Vignon, E., and Pieczynski, W. (2021b). Highly Fast Text Segmentation With Pairwise Markov Chains. In *6th IEEE Congress on Information Science and Technology*, pages 361–366.
- Azeraf, E., Monfrini, E., Vignon, E., and Pieczynski, W. (2021c). Introducing the Hidden Neural Markov Chain Framework. In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 2*, pages 1013–1020.
- Barbieri, F., Camacho-Collados, J., Espinosa-Anke, L., and Neves, L. (2020). TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification. In *Proceedings of Findings of EMNLP*.
- Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Brants, T. (2000). TnT: a Statistical Part-of-Speech Tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, pages 224–231.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.
- Devroye, L., Györfi, L., and Lugosi, G. (2013). *A Probabilistic Theory of Pattern Recognition*, volume 31. Springer Science & Business Media.
- Duda, R. O., Hart, P. E., et al. (2006). *Pattern Classification*. John Wiley & Sons.
- Fukunaga, K. (2013). *Introduction to Statistical Pattern Recognition*. Elsevier.
- Ian Goodfellow, Y. B. and Courville, A. (2016). *Deep Learning*. MIT Press.
- Jebara, T. (2012). *Machine Learning: Discriminative and Generative*. Springer Science & Business Media.
- Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain.
- Jurafsky, D. and Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics, 2nd Edition*. Prentice-Hall.
- Kim, S.-B., Han, K.-S., Rim, H.-C., and Myaeng, S. H. (2006). Some Effective Techniques for Naive Bayes Text Classification. *IEEE Transactions on Knowledge and Data Engineering*, 18(11):1457–1466.
- Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.
- Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT press.
- Komninos, A. and Manandhar, S. (2016). Dependency Based Embeddings for Sentence Classification Tasks. In *Proceedings of the 2016 conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1490–1500.
- LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep Learning. *Nature*, 521(7553):436–444.
- LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural computation*, 1(4):541–551.
- Lhoest, Q., Villanova del Moral, A., Jernite, Y., Thakur, A., von Platen, P., Patil, S., Chaumond, J., Drame, M., Plu, J., Tunstall, L., Davison, J., Šaško, M., Chhablani, G., Malik, B., Brandeis, S., Le Scao, T., Sanh, V., Xu, C., Patry, N., McMillan-Major, A., Schmid, P., Gugger, S., Delangue, C., Matussière, T., Debut, L., Bekman, S., Cistac, P., Goehringer, T., Mustar, V., Lagunas, F., Rush, A., and Wolf, T. (2021). Datasets: A Community Library for Natural Language Processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.
- Liu, B., Blasch, E., Chen, Y., Shen, D., and Chen, G. (2013). Scalable Sentiment Classification for Big Data Analysis using Naive Bayes Classifier. In *IEEE International Conference on Big Data*, pages 99–104.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A Robustly Optimized BERT Pre-training Approach. *arXiv preprint arXiv:1907.11692*.
- Loshchilov, I. and Hutter, F. (2016). SGDR: Stochastic Gradient Descent with Warm Restarts. *arXiv preprint arXiv:1608.03983*.
- Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.
- Malo, P., Sinha, A., Korhonen, P., Wallenius, J., and Takala, P. (2014). Good Debt or Bad Debt: Detecting Semantic Orientations in Economic Texts. *Journal of the Association for Information Science and Technology*, 65(4):782–796.
- McCallum, A., Nigam, K., et al. (1998). A Comparison of Event Models for Naive Bayes Text Classification. In

AAAI-98 Workshop on Learning for Text Categorization, volume 752 (1), pages 41–48.

- Metsis, V., Androutsopoulos, I., and Paliouras, G. (2006). Spam Filtering with Naive Bayes - which Naive Bayes? In *CEAS*, volume 17, pages 28–69.
- Mohammad, S., Bravo-Marquez, F., Salameh, M., and Kiritchenko, S. (2018). Semeval-2018 task 1: Affect in tweets. In *Proceedings of the 12th international workshop on semantic evaluation*, pages 1–17.
- Nair, V. and Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the International Conference on Machine Learning*.
- Ng, A. Y. and Jordan, M. I. (2002). On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. In *Advances in Neural Information Processing Systems*, pages 841–848.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32:8026–8037.
- Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning Representations by Back-Propagating Errors. *nature*, 323(6088):533–536.
- Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013). Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.
- Sutton, C. and McCallum, A. (2006). An Introduction to Conditional Random Fields for Relational Learning. *Introduction to Statistical Relational Learning*, 2:93–128.
- Tang, D., Qin, B., and Liu, T. (2015). Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Wainwright, M. J. and Jordan, M. I. (2008). *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc.
- Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *In the Proceedings of International Conference on Learning Representations*.

APPENDIX

Proof of the Classifier Induced from the Pooled MC Written Discriminatively

The joint law of the Pooled MC is given with (5). For all $\lambda_i \in \Lambda_X$ and the realization $Y_{1:T} = y_{1:T}$, it can be written:

$$\begin{aligned} p(X = \lambda_i, y_{1:T}) &= p(X = \lambda_i) p(y_1 | X = \lambda_i) \prod_{t=1}^{T-1} p(y_{t+1} | X = \lambda_i, y_t) \\ &= p(X = \lambda_i) \frac{p(X = \lambda_i, y_1)}{p(X = \lambda_i)} \prod_{t=1}^{T-1} \frac{p(X = \lambda_i, y_t, y_{t+1})}{p(X = \lambda_i, y_t)} \\ &= p(y_1) \prod_{t=1}^{T-1} p(y_{t+1} | y_t) L_{y_1}^{MC,1}(i) \prod_{t=1}^{T-1} \frac{L_{y_t, y_{t+1}}^{MC,2}(i)}{L_{y_t}^{MC,1}(i)} \end{aligned}$$

Therefore, the posterior law of the Pooled MC can be written using the Bayes rule, allowing to define the Bayes classifier with the MAP criterion of the Pooled MC as (6).

Proof of the Classifier Induced from the Pooled MC2 Written Discriminatively

The joint law of the Pooled MC2 is given with (7). For all $\lambda_i \in \Lambda_X$ and the realization $Y_{1:T} = y_{1:T}$, it can be written:

$$\begin{aligned} p(X = \lambda_i, y_{1:T}) &= p(X = \lambda_i) p(y_1 | X = \lambda_i) p(y_2 | X = \lambda_i, y_1) \\ &\quad \times \prod_{t=1}^{T-2} p(y_{t+2} | X = \lambda_i, y_t, y_{t+1}) \\ &= p(X = \lambda_i) \frac{p(X = \lambda_i, y_1)}{p(X = \lambda_i)} \frac{p(X = \lambda_i, y_1, y_2)}{p(X = \lambda_i, y_1)} \\ &\quad \times \prod_{t=1}^{T-2} \frac{p(X = \lambda_i, y_t, y_{t+1}, y_{t+2})}{p(X = \lambda_i, y_t, y_{t+1})} \\ &= p(y_1, y_2) \prod_{t=1}^{T-2} p(y_{t+2} | y_t, y_{t+1}) \\ &\quad \times L_{y_1, y_2}^{MC2,1}(i) \prod_{t=1}^{T-2} \frac{L_{y_t, y_{t+1}, y_{t+2}}^{MC2,2}(i)}{L_{y_t, y_{t+1}}^{MC,1}(i)} \end{aligned}$$

Therefore, the posterior law of the Pooled MC2 can be written using the Bayes rule, allowing to define the Bayes classifier with the MAP criterion of the Pooled MC2 as (8).