Drift Detection in Models Applied to the Recognition of Intentions in Short Sentences Using Convolutional Neural Networks for Classification

Jairo R. Junior and Leandro A Silva

Graduate Program in Electrical and Computer Engineering.
Presbyterian Mackenzie University
São Paulo, SP – Brazil

Abstract. Significant advancements have been achieved in natural language processing models for text classification with the emergence of pre-trained transformers and deep learning. Despite promising results, deploying these models in production environments still faces challenges. Classification models are continuously evolving, adapting to new data and predictions. However, changes in data distribution over time can lead to a decline in performance, indicating that the model is outdated. This article aims to analyze the lifecycle of a natural language processing model by employing multivariate statistical methods capable of detecting model drift over time. These methods can be integrated into the training and workflow management of machine learning models. Preliminary results show that the statistical method *Maximum Mean Discrepancy* performs better in detecting drift in models trained with data from multiple domains through high-dimensional vector spaces after being subjected to an untrained auto-encoder. The classifier model achieved an accuracy rate of 93% in predicting intentions, using accuracy as the evaluation metric.

Keywords: Intent recognition, Drift detection, Data drift, MLOps (Machine Learning Operations)

1 Introduction

The research on Natural Language Processing (NLP) originated in the 1950s, emerging as an intersection between artificial intelligence and linguistics [1]. This landscape has provided avenues for investigations, where algorithms are employed to analyze, comprehend, and extract information from human language, enabling the creation of machine learning models. These models have demonstrated remarkable abilities, such as automatic summarization, text translation, entity recognition, sentiment analysis, speech, and intent recognition [1][2].

Intent recognition, also known as natural language understanding, is a subfield of NLP aimed at comprehending and recognizing the contextual meanings of human communication, using examples of sentences and intentions classified by machine learning models [2]. In the context of NLP, intent recognition aims to classify user statements into pre-defined intention categories, according to specific domains [2][3], becoming an essential technique for building intelligent dialogue systems [4]. The process of intent recognition faces several challenges, such as the lack of corpora with annotated intentions [5], the detection of multiple intentions in a single expression, and the irregularity of human expressions, which characterizes issues of data drift [6][7].

The concept of data drift refers to the variation in the data used to train and validate a model compared to the data encountered at the time of its implementation, causing changes in data distribution. This discrepancy can lead to model degradation, resulting in drift over time [8].

David C. Wyld et al. (Eds): AI&FL, SCM, NLPTT, DSCC -2023
pp. 23-36, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.131404

Despite significant advances in drift detection for machine learning models on structured data, few methodologies have addressed drift detection in unstructured data, such as text and images. Data drift can occur in both the characteristics of sentences (virtual drift) and the distributions of intentions (real drift) [9]. Virtual drift may be caused by models trained with insufficient data for all instances or by encountering new words, grammatical constructions, and changes in writing style, leading to an incorrect prior probability assumption. On the other hand, real drift can be caused by seasonal events of misinformation [10], sexism [11], clickbait [12], among others.

Efforts are made by machine learning engineers to use machine learning model performance metrics to monitor drift and consequently the model's performance after it has been deployed and made available for use in real-world applications. However, this task faces a significant challenge related to the availability of correct classes during predictions. While the classes are present in the training data, they may not always be available after the model is deployed [8]. This gap between the known classes during training and the unknown classes during the operational phase can impact the model's performance and accuracy, necessitating approaches to deal with these data drift cases.

The challenges faced when dealing with drift in an intent recognition model can vary depending on the nature, extent, and types of deviations present. In certain cases, managing drift is possible through model retraining; however, in more complex scenarios, developing a new model with different approaches and parameters may be necessary to achieve the desired performance. Therefore, it becomes crucial to employ processes that enable the identification of drift, defining acceptable thresholds to configure proactive alerts for taking action. With this purpose, the objective of this work is to compare drift detectors originally designed for low-dimensional vector spaces and apply them to detect drift in texts, using advanced techniques of natural language processing and dimensionality reduction [13]. This approach has the potential to provide higher accuracy and effectiveness in detecting and handling drift in intent recognition models, contributing to the robustness and reliability of these systems in real-world applications.

In addition to the introduction that presents the context and objective of the work, the article is organized in: Section 2, a review of the works related to the theme of this article is presented; Section 3 brings specific information about the data used in the study and we present the methodology adopted for the analysis of the collected data; the results found are detailed in Section 4; finally, in Section 5, it is presented the conclusion and future work.

2 Related Works

To automate drif detection and notify machine learning engineers about the need to update the model,automatic drift detectors have been proposed [14][15][16]. However, state-of-the-art drift detectors face limitations when applied to more advanced natural language models, as they assume that the input data is in manually-designed, low-dimensional vector spaces. On the other hand, modern applications employ complex language models such as transformers and high-dimensional word-embedding layers. Our research builds upon the studies of [9], which introduced data drift and concepts, including patterns of change over time, which are part of our drift detection experiments, such as the injection of out-of-context words. Furthermore, [17] conducted an analysis using unstructured data with methods like Kolmogorov-Smirnov (KS) and Least-Squares Density Difference, which also integrate into our comparative experiments between detection methods.

Although there are works that compare drift detectors [18][17], our proposed study aims to perform a comprehensive comparison of these detectors on complex natural language

processing models, particularly in the context of intent recognition in texts. Additionally, we investigate the creation of a model using convolutional neural networks for natural language processing and the applicability of the detector on imbalanced data, which was introduced into the sentences used to train the model, simulating a real-world usage scenario.

Compared to the existing general approach, our study fills the gap by using unstructured data from multiple domains through high-dimensional vector spaces after being subjected to an untrained autoencoder. This promising approach enables a more comprehensive and improved analysis of data drift in natural language models, providing a better understanding of the challenges faced in this context and enhancing the application of drift detectors in practical scenarios.

3 Preliminaries

3.1 Datasets and Embedding Models

The process of creating the database used in this work was carried out by accessing services provided by the social media platform Twitter, using a developer platform that allows real-time access to messages, known as tweets, posted by its users, for research and analysis purposes.

The data was extracted from February 3, 2023, to March 2, 2023, using intent-related filters for five categories: *music*, *politics*, *movies*, *sports*, and *technology*. The collection was conducted on tweets in the English language, resulting in a total of 67,000 collected tweets. The distribution of this data is shown in Figure 1.

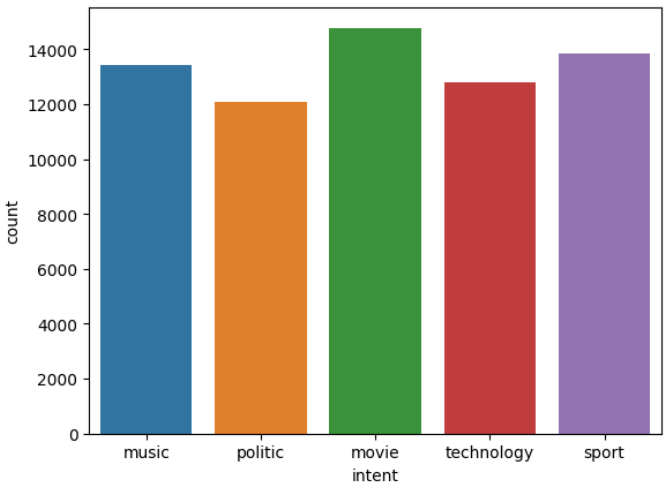

Fig. 1. Number of sentences extracted by intention

Data from social media present peculiarities that are not found in other unstructured text databases. Some of these peculiarities include the presence of special characters, icons, external links, line breaks, and specific terms used on the platform, such as the word "RT," which indicates a reply to another published tweet, as illustrated in Table 1. These unique characteristics demand a careful approach in processing and analyzing the data collected from social media to ensure the accuracy and effectiveness of the applied methods.

tweet	intent
RT @coot271: She's in your home.\n\nFULL UNCUT, HD movie; thanks to @MethadosA for being the inspiration	movie
@gothamfreak_ yea it was wild, shes amazing in the movie too.	movie
Interesting results that happened at the box o...	movie
RT @MarvelStudios: Check out these photos from the red carpet at the Australian	movie
RT @nicktiffany95: Take a listen and learn why #TheBatman found itself in my Top 10 Films of 2022!	movie

Table 1. Example of sentences extracted before the cleaning step

With the established database, the process of data preparation for model training began, aiming to use the pre-trained transformer BERT as the input layer for the deep neural network convolutional (CNN).

The BERT (Bidirectional Encoder Representations from Transformers) model is a machine learning approach based on deep neural networks known as Transformers. Unlike traditional natural language processing (NLP) models that read text in a single direction, BERT is bidirectional, meaning that it considers the context of each word from the words that precede and follow it [19].

The training process of BERT involves a prior pre-training step, where the model is exposed to large amounts of unlabeled text to learn the contextual representation of words in a general task of predicting masked words. Subsequently, the model is fine-tuned in a specific task, known as fine-tuning, where it is provided with a labeled dataset for a particular task, such as intent classification in an intelligent dialogue.

The combination of BERT with a Convolutional Neural Network (CNN) involves using the output of BERT, which is a contextual representation of words in a text, as input for the CNN. The CNN is responsible for processing and extracting relevant features from the word representations to perform classification [20].

The CNN is a neural network architecture particularly suitable for feature extraction in high-dimensional data, such as text. It uses convolutional filters to scan through the text representation and to identify important patterns and features. These filters can detect different characteristics, such as keywords or syntactic patterns, which can be useful for the classification task.

In the context of text classification, the CNN uses convolutional filters to detect patterns and features in the word representations provided by BERT. Then, these features are passed through additional layers of the CNN to learn more complex relationships between words. Finally, the output of the CNN is used to perform classification into desired categories or intentions.

This approach of combining BERT with a CNN is powerful because BERT provides rich contextual representations of words, capturing important contextual information, while the CNN extracts relevant features from these representations for the classification task. This combination results in a highly effective text classification model capable of handling nuances and complexities present in natural language data [21].

To ensure the coherence of the model input with the proposed architecture [21], the following steps were performed, namely:

– Adapting the vector space created through word tokenization, limiting the number of tokens per sentence between 50 and 150, based on the average analysis of sentence lengths, as illustrated in the figure 2;

– Tokenizing each word of the tweet sentences, generating token sequences for each sentence;
– Converting the token sequences into corresponding numeric IDs for each token, using the vocabulary of the BERT model;
– Padding the sequences to ensure they all have the same length;

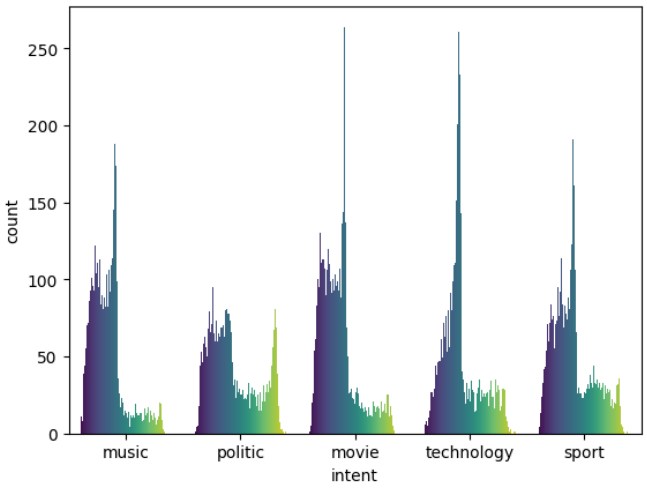

Fig. 2. Number of tokens by sentences and intent

The tokenization step involves breaking down the text into small token units using a vocabulary of words, facilitating the correct separation of sentence tokens. For the tokenization process, the WordPiece model extracted from the pre-trained BERT model with 12 layers and 768 dimensions was used to generate the tokenizer. Before tokenization, some tasks were performed, such as normalizing uppercase letters to lowercase and including the tokens "[CLS]", used as a token to indicate the start of a new sentence, and the token "[SEP]", which represents the end of a sentence and the start of a new one, both necessary as input for the classification model.

In addition to tokenization and transforming words into their respective IDs, the classifier model requires a fixed sentence length as input, due to the varying number of tokens in each sentence. To meet this requirement, the model uses the token "[PAD]" to distinguish and handle the convolution process in the neural network.

3.2 Intention recognition

With the aim of recognizing and classifying each tweet sentence into an intent, we adopted a deep learning model called Convolutional Neural Network (CNN) [21], as illustrated in Figure 3.

In the CNN architecture, the first layer of the neural network was built using the pre-trained BERT model. This layer is responsible for transforming each input generated in the pre-processing step into a matrix representation of word embeddings, allowing calculations in the convolutional layers. In this process, each sentence is converted into a matrix representation, where the number of rows in the matrix corresponds to the number of words in the sentence. This word embeddings matrix enables efficient information processing in the convolutional neural network, contributing to the accurate recognition of intents in tweet sentences.

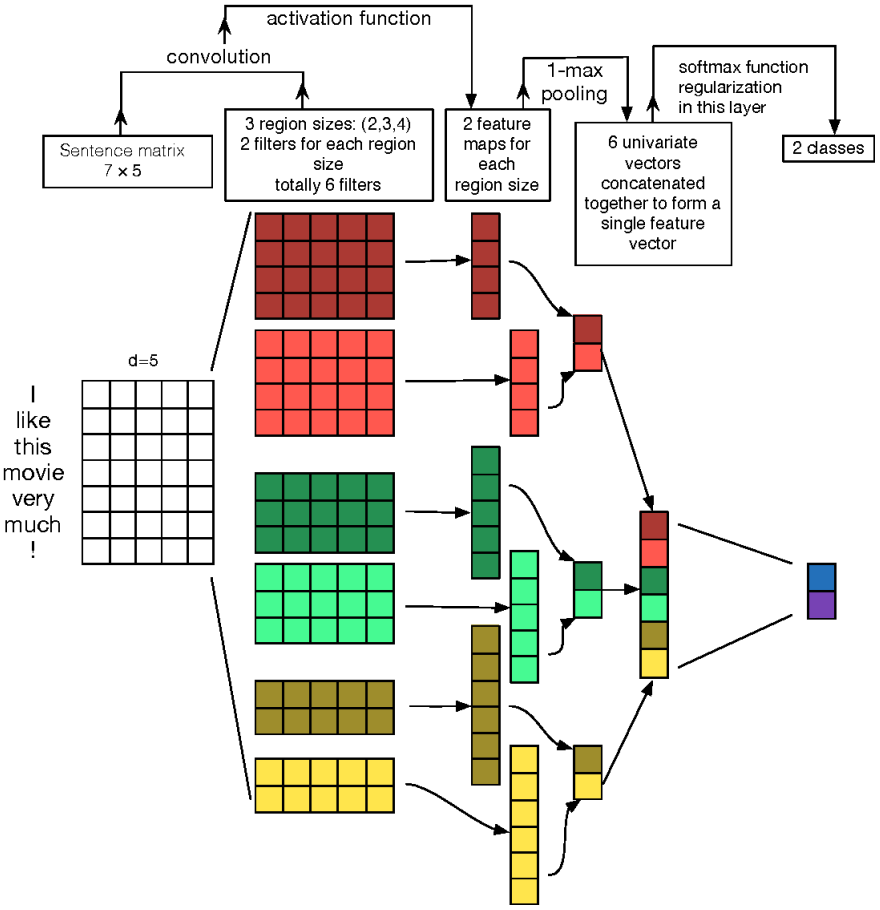

Fig. 3. Inspired architecture for text classification using convolutional neural networks [21]

In addition to the embedding layer, the model was composed of other features, namely:

– **Convolution layers**: The model consists of 3 one-dimensional convolution layers with 2, 3, and 4 kernels, respectively. Each convolution layer was trained with 100 filters using the ReLU activation function defined by equation 1

$$ReLU(x) = max\{0, x\} \qquad ReLU'(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{c.c.} \end{cases} \tag{1}$$

– **Max pooling layers**: One-dimensional layer responsible for retrieving the value of each filter generated by the convolution layers.
– **Dense layer**: Layer composed of 256 neurons that receive the concatenated results from the max-pooling layer and apply the ReLU activation function. 1;
– **Dropout layer**: Layer to prevent overfitting;
– **Dense layer result**: Output layer of the model consisting of 5 neurons that use the softmax activation function 2 returning a probability for each of the classes;

$$softmax(y)_i = \frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_i}} \tag{2}$$

In Equation 2, y is the vector subjected to the softmax function, and e^{y_i} is the element at position i of the vector. The normalization term is given by $\sum_{j=1}^{e^{y_i}\ n}$, which ensures

a probability distribution that satisfies $0 \geq softmax(y)_i \leq 1$. The number of classes, or possible outputs, is given by n.

After the cleaning and preprocessing step, the database was left with approximately 48,000 records. For the training phase, the database was shuffled and divided into batches of 32 records each, which were fed to the model for 10 epochs.

3.3 Evaluation of the classification model

From the database formed after the cleaning and preprocessing process, 80% of the data was allocated for model training, and 20% for validation. Categorical cross-entropy was employed as the loss function, expressed by Equation 3

$$\text{Categoric crossentropy}(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \tag{3}$$

where $\hat{y}_i$ is the i-th scalar value of the network output, y_i is the desired output, and N is the number of network outputs.

The Adam optimizer [22] was used to update the weight parameters of the convolutional network, with the aim of minimizing the loss function with a learning rate of 0.0001. Accuracy, calculated through Equation 4, was the chosen metric to evaluate the neural network during the training and validation steps. Using the sentences classified as true positive (TP), false positive (FP), true negative (TN), and false negative (FN) from a confusion matrix.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

3.4 Drift detectors

For the detection of classifier model drift, three detectors were used, including one univariate statistical detector:: *Kolmogorov-Smirnov* (KS) and two multivariate statistical detectors: *Maximum Mean Discrepancy* (MMD) and *Least-Squares Density Difference* (LSDD), The goal was to identify deviations in the relationship between the data used in training and the data generated by subsequent predictions after the model was trained, as follows:

***Kolmogorov-Smirnov* (KS)**: Statistical test of agreement between two probability distributions using the maximum absolute difference between the distributions.

***Maximum Mean Discrepancy* (MDD)**: Kernel-based static independence test for multivariate 2-sample tests through the equation

$$MMD(F, p, q) = ||\mu_p - \mu_q||_F^2 \tag{5}$$

MDD is a distance-based measure between two distributions p and q based on the mean embeddings $\mu_p - \mu_q$ in a reproducing *Kernel Hilbert space* space F.

***Last-squares density difference detector* (LSDD)**: Method for multivariate 2-sample tests. The least-squares density difference between two distributions is found through p and q, where X is defined by the equation

$$LSDD(p, q) = \int_{\mathcal{X}} (p(x) - q(x))^2 \, dx. \tag{6}$$

Given two samples, it is possible to calculate an estimate of the least-squares density between the two underlying distributions and use them as a test statistic to determine whether a sample belongs or not to a reference set.

4 Our proposal

4.1 Comparison of drift detectors

For the comparison step, the following processes were performed: splitting the training data for reference and validation of the detector, tokenization of sentences, generating word embeddings of the generated tokens, reducing the dimension of each embedding through an autoencoder, calibrating each chosen detection method in the work, and finally, a hypothesis test to conclude the effectiveness of the detector, as illustrated in Figure 4.

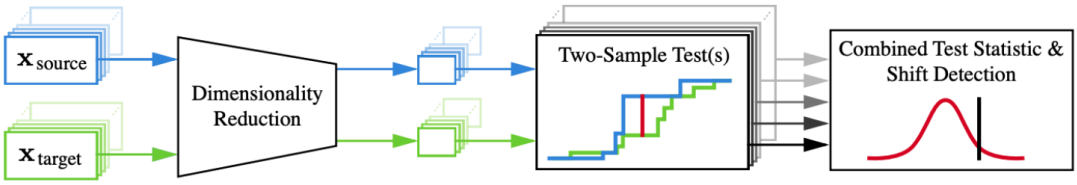

Fig. 4. Illustration of the process to detect data set change. Source and destination data are fed through a dimensionality reduction process and subsequently analyzed through statistical hypothesis testing [16]

The data used for the calibration of the detectors were the same as those used for training the classifier model, which served as reference data for the detection methods. The reference data is divided into 10 batches, each containing 1,000 sentences. The first batch is used for the hypothesis test (H0) with the aim of evaluating whether the detector generates false positives when a new accuracy, not different from the average of the reference samples, is characterized as a deviation, potentially indicating drift in the already deployed model. The other batches were divided and organized according to Table 2.

Batch	Reference data	Unbalanced data
1	1000 tweets	0 tweets
2	900 tweets	100 tweets
3	800 tweets	200 tweets
4	700 tweets	300 tweets
5	600 tweets	400 tweets
6	500 tweets	500 tweets
7	400 tweets	600 tweets
8	300 tweets	700 tweets
9	200 tweets	800 tweets
10	100 tweets	900 tweets

Table 2. Data separation for detector evaluation

To evaluate the sensitivity and specificity of the detectors, batches containing controlled drift data were created by including out-of-context words in each sentence, with the aim of unbalancing each batch, as presented in Table 2. For this task, the 10 most used words in each intention were analyzed, as shown in Figure 5. These words were separated and later injected into sentences different from those that characterize their domain. For example, for sentences related to *music*, words more common in the intentions corpus of *movies*, *politics*, *sports*, and *technology* were injected. This approach allowed the creation of batches of data with controlled drift, for the evaluation of the detectors under different conditions of imbalance and deviation.

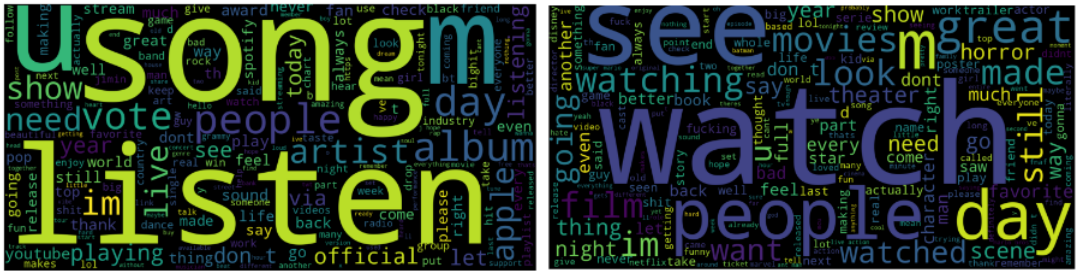

Fig. 5. Word cloud generated with frequent words for music and film intents

The purpose of creating this subset is to assess the speed and reliability with which the drift detectors detect the gradually introduced drift. The injection of out-of-context words was carried out following some criteria, as described below:

- In sentences with more than 10 tokens and less than 20 tokens, two words corresponding to other intentions were imputed, randomly inserted between the middle and the end of the sentence;
- In sentences with more than 20 tokens, three words corresponding to other intentions were imputed, randomly inserted at the beginning, middle, and end of the sentence;
- In cases where the sentence had a predefined intention, the token corresponding to that intention was removed.

Both detectors were initialized using the hypothesis test measure, *p-value*, with a threshold value of 0.05, defined as the threshold for the existence or absence of deviation or difference between the samples. Table 3 assists in this interpretation.

P-value	**Interpretation**
P<0,01	very strong evidence against H0
0,01<= P <0,05	moderate evidence against H0
0,05<= P <0,10	suggestive evidence against H0
0,10<= P	little or no real evidence against H0

Table 3. Reasonable interpretation of p-values in statistical hypothesis tests [23]

4.2 Dimensionality reduction

Detecting drift in raw or tokenized texts is not effective as they do not adequately represent the semantics of the input [17]. Similar to the process of training the classifier model, the data was preprocessed to extract contextual embeddings of the sentences, using the pre-trained BERT model as the tokenizer and *word embedding*.

The embedding space of the BERT model used for sentence embedding consists of a 768-dimensional vector. Due to the size of this space, calibrating the model may require high computational cost to perform statistical calculations and may not capture token similarities effectively after the embedding process. To handle high-dimensional data, it is common to reduce dimensionality before performing hypothesis tests [16]. Therefore, a drift detector can be applied to the lower-dimensional data, capturing token similarities and dissimilarities more effectively among the analyzed samples. The dimensionality reduction process was performed using an Untrained Autoencoder (UAE), where the data was

processed and transformed into a lower-dimensional space, resulting in 32 dimensions per embedded sentence, as opposed to the initial 768 dimensions generated by the pre-trained BERT model. This approach made the detection process more efficient and effective, with minimal loss of relevant information for drift analysis in the data.

4.3 Experimental Results

Table 4 presents the results obtained from different CNN model trainings, varying the number of epochs.

Epoch	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Loss	0.1925	0.1848	0.1845	0.1972	0.2136	0.2558	0.2352	0.2359	0.2448	0.2530
Acc	0.9367	0.9383	0.9384	0.9385	0.9398	0.9381	0.9397	0.9379	0.9397	0.9391

Table 4. Model performance by training epoch

The obtained results were experimentally validated, and it was decided to extrapolate the values up to 10 epochs of training, at which point a reduction in the model's performance was observed. After these 10 epochs, the model achieved an accuracy of 93%. Figure 6 illustrates the optimization process (a) and the performance (b) of the CNN over the 10 epochs of training.

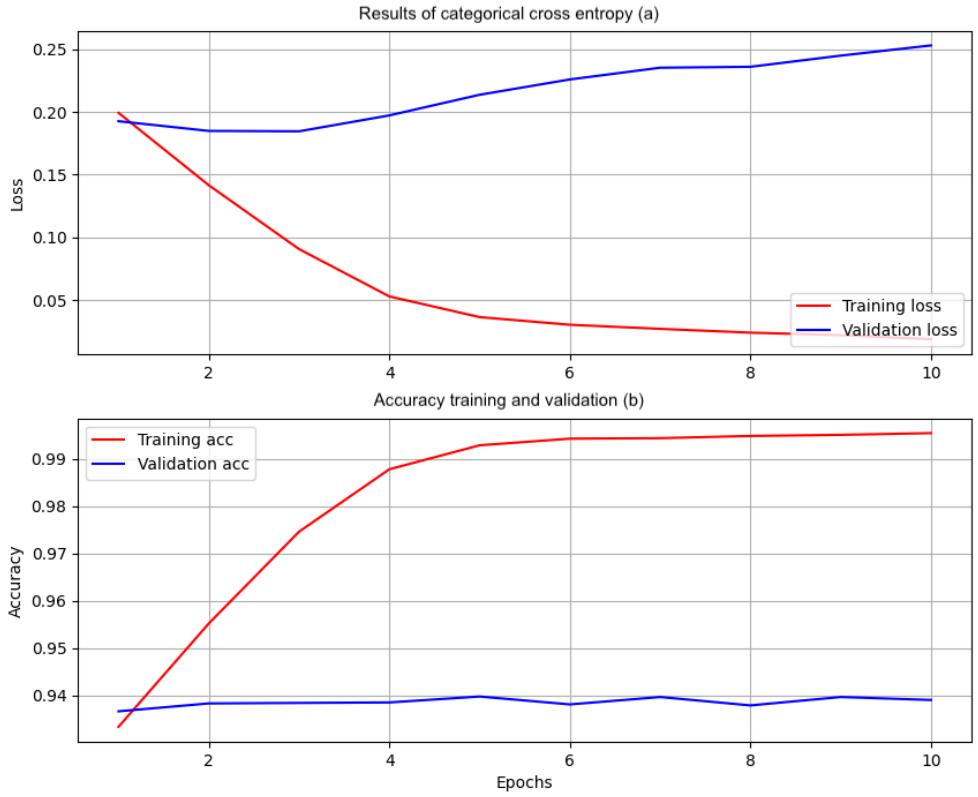

Fig. 6. Performance result of the classifier model

After submitting new data to the trained model, containing imbalanced data, all predictions were grouped into batches and submitted to the created detectors. Overall, all

detection methods were able to distinguish the drift in all batches. Through Figure 7, it can be observed that as the batches were submitted with an increase in incorrectly classified data, as illustrated in Figure 8, the standard deviation between the predicted data and the reference data, used to calibrate the detector, increased appropriately.

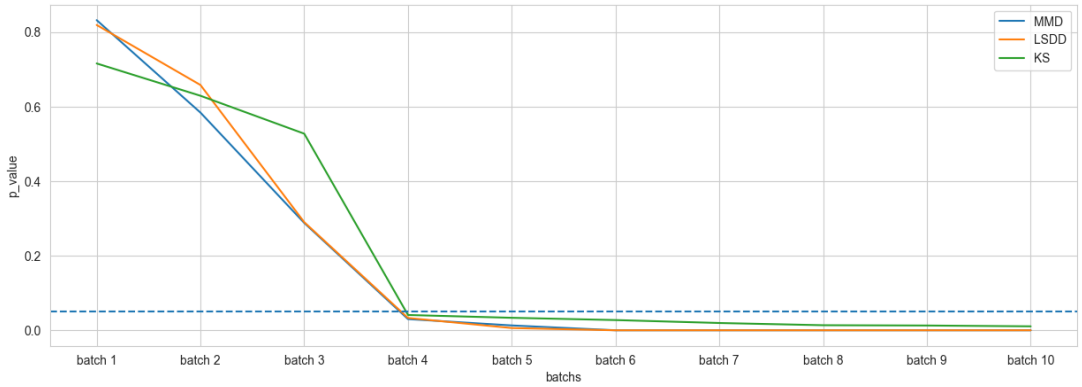

Fig. 7. Statistical p-value analysis based on batches created and methods used for drift detection

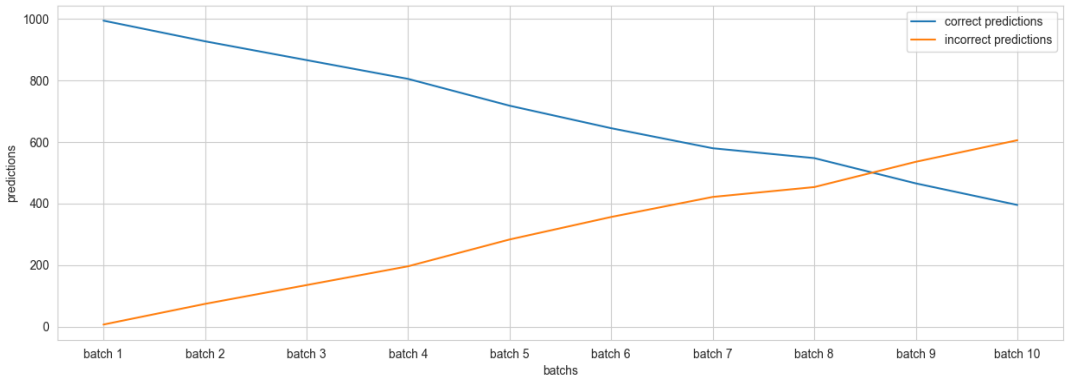

Fig. 8. Result of data classification when submitted to batches with unbalanced data

Based on the configurations of the generated batches for analysis, we evaluated the confidence of the drift detectors by increasing the incorrect predictions, simulating a real-world scenario for the intention *movie*. An ideal drift detector produces a low p-value when no drift is identified and a low standard deviation with few or no incorrect predictions.

When analyzing the standard deviation, as shown in Figure 9, and the statistical measure *p-value*, illustrated in Figure 7, between the imbalanced data and the reference data, we observed that the *Kolmogorov-Smirnov* (KS) method performed the worst compared to the other methods. On the other hand, the *Maximum Mean Discrepancy* (MMD) method performed the best among the methods used. The MMD method showed good performance in detecting drift in the batch of data separated for the hypothesis test, and its performance remained consistent as the batches were subjected to an increase in imbalanced data.

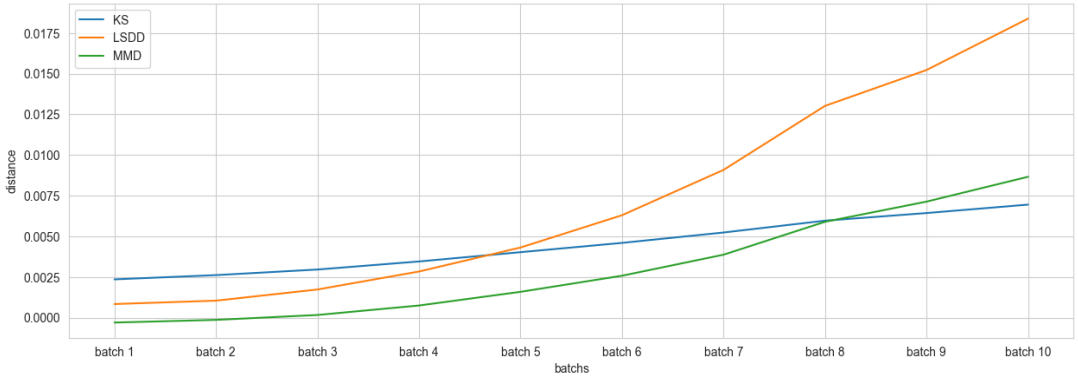

Fig. 9. Standard deviation based on batches created between reference data and unbalanced data

5 Conclusion

In conclusion, intelligent dialogues, addressed in this study through intention recognition methods, have proven to be an increasingly utilized technology in the industry to solve a variety of problems. However, ensuring continuous performance and effectiveness of these models over time has become crucial for their practical application.

The multivariate methods Least-Squares Density Difference (LSDD) and Maximum Mean Discrepancy (MMD) demonstrated the ability to identify degradation in intention recognition models after their deployment, as evidenced by the standard deviation and p-value hypothesis tests between the reference data and the unbalanced data. This early drift detection capability is essential for alerting machine learning engineers to the need for model updates and ensuring their continuous effectiveness.

The convolutional neural network adopted as the deep learning model demonstrated a performance of 93% in predicting intentions, measured by accuracy as an evaluation metric. This proves the effectiveness of this approach for intention recognition in texts, contributing to a more enhanced experience in intelligent dialogues.

Based on the obtained results, it becomes evident the importance of adopting effective methods for detecting drift in intention recognition models, especially in real-world environments where data may undergo significant changes over time. The application of LSDD and MMD methods has shown to be highly promising in maintaining reliability and adequate performance of models in production.

In summary, this study contributed to enhancing the understanding of drift detection in intention recognition models, presenting efficient approaches for monitoring and managing these models in practical scenarios. It is expected that these findings will further drive the utilization of intelligent dialogues in various fields, providing more robust and effective solutions for the industry and society as a whole.

In future work, we would like to further explore the effect of different dimensionality reduction techniques on drift detectors, incorporating new approaches in generating text embeddings, for instance, based on diverse models such as LLAMA and GPT.

References

1. Nadkarni, Prakash M and Ohno-Machado, Lucila and Chapman, Wendy W. Natural language processing: an introduction. Journal of the American Medical Informatics Association, vol. 18, 2011, pp. 544-551.
2. Jiao Liu and Yanling Li and Min Lin. Review of Intent Detection Methods in the Human-Machine Dialogue System. Journal of Physics: Conference Series, vol. 1267, 2019, pp. 012-059.

3. Ravuri, Suman and Stoicke, Andreas. A comparative study of neural network models for lexical intent classification. 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), 2015, pp. 368-374.
4. Mctear, Michael and Seminck, Olga. Conversational AI: Dialogue Systems, Conversational Agents, and Chatbots by Michael McTear. Massachusetts Institute of Technology Press (MIT Press), vol. 13, 2022, pp. 1-4.
5. Stefan Larson and Anish Mahendran and Joseph J. Peper and Christopher Clarke and Andrew Lee and Parker Hill and Jonathan K. Kummerfeld and Kevin Leach and Michael A. Laurenzano and Lingjia Tang and Jason Mars. An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction. CoRR, vol. abs/1909.02027, 2019.
6. Xie, Wenxiu and Gao, Dongfa and Ding, Ruoyao and Hao, Tianyong". A Feature-Enriched Method for User Intent Classification by Leveraging Semantic Tag Expansion. Springer International Publishing, 2018, pp. 224-234
7. Fernández-Martínez, Fernando and Griol, David and Callejas, Zoraida and Luna Jiménez, Cristina. An approach to intent detection and classification based on attentive recurrent neural networks, 2021.
8. Patrick Lindstrom and Sarah Jane Delany and Brian Mac Namee. Handling Concept Drift in a Text Data Stream Constrained by High Labelling Cost. AAAI Press, 2010.
9. Gama, João and Žliobaitundefined, Indrundefined and Bifet, Albert and Pechenizkiy, Mykola and Bouchachia, Abdelhamid. A Survey on Concept Drift Adaptation. Association for Computing Machinery, 2014, vol. 46,pp. 224-234
10. Kumar, Srijan and West, Robert and Leskovec, Jure. Disinformation on the Web: Impact, Characteristics, and Detection of Wikipedia Hoaxes. International World Wide Web Conferences Steering Committee, 2016, pp. 591–602.
11. Ghosh Chowdhury, Arijit and Sawhney, Ramit and Shah, Rajiv Ratn and Mahata, Debanjan.YouToo? Detection of Personal Recollections of Sexual Harassment on Social Media. Association for Computational Linguistics, 2019, pp. 2527-2537.
12. Chen, Yimin and Conroy, Niall J. and Rubin, Victoria L. Misleading Online Content: Recognizing Clickbait as "False News". Association for Computing Machinery, 2015, pp. 15–19.
13. Chen, Yimin and Conroy, Niall J. and Rubin, Victoria L. Misleading Online Content: Recognizing Clickbait as "False News". Association for Computing Machinery, 2015, pp. 15–19.
14. Arthur Gretton and Karsten M. Borgwardt and Malte J. Rasch and Bernhard Schölkopf and Alexander Smola. A Kernel Two-Sample Test. Journal of Machine Learning Research, 2012, vol. 13,pp. 723-773.
15. David Lopez-Paz and Maxime Oquab. Revisiting Classifier Two-Sample Tests, 2017.
16. Rabanser, Stephan and Günnemann, Stephan and Lipton, Zachary. Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift. Inc Curran Associates, vol. 32, 2019.
17. Feldhans, Robert and Wilke, Adrian and Heindorf, Stefan and Shaker, Mohammad Hossein and Hammer, Barbara and Ngonga Ngomo, Axel-Cyrille and Hüllermeier, Eyke. Drift Detection in Text Data with Document Embeddings. Springer-Verlag, 2021, pp. 107-118.
18. Lucas Baier and Niklas Kühl and Gerhard Satzger. How to Cope with Change? - Preserving Validity of Predictive Services over Time, 2019
19. Jacob Devlin and Ming-Wei Chang and Kenton Lee and Kristina N. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018.
20. Yoon Kim. Convolutional Neural Networks for Sentence Classification. CoRR, 2014
21. Ye Zhang and Byron C. Wallace. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. CoRR, vol. abs/1510.03820, 2015.
22. Kingma, Diederik P. and Ba, Jimmy. Adam: A Method for Stochastic Optimization, 2014.
23. H. Arsham. Kuiper's P-value as a measuring tool and decision procedure for the goodness-of-fit test. Journal of Applied Statistics, vol. 15, 1988, pp. 131-135.

Authors

Jairo R Junior Graduated in Information Systems from Mackenzie University, he pursued post-graduation in Data Science at Mackenzie University, and currently, he is enrolled in the Master's program in Electrical Engineering, also at Mackenzie University. Presently,

he work as a software engineer consultant at Telefônica Brasil. As a research focus, he has mainly been involved in areas related to Natural Language Processing, including Artificial Neural Networks, Machine Learning, Natural Language Processing, and Data Mining.

Leandro A Silva Graduated in Computer Engineering, he holds a Master's and Ph.D. from the Polytechnic School of the University of Sao Paulo (USP). Currently, he is a Professor at the Faculty of Computing and Informatics (FCI) and a faculty member of the Academic Stricto-Sensu Graduate Program in Electrical Engineering and Computing (PPGEEC) and the Professional Stricto-Sensu Graduate Program in Applied Computing (PPGCA). Additionally, he serves as the Coordinator of Research Funding (CFP) at the Office of Research and Postgraduate Studies (PRPG) of Mackenzie Presbyterian University (UPM). As research focus, he has primarily worked in areas related to Data Science, including Artificial Neural Networks, Machine Learning, Data Mining, Big Data, and Pattern Recognition.