3-17-2023 2:00 PM

A Hybrid Continual Machine Learning Model for Efficient Hierarchical Classification of Domain-Specific Text in The Presence of Class Overlap (Case Study: IT Support Tickets)

Yasmen M. Wahba, *The University of Western Ontario*

Supervisor: Madhavji, Nazim H., *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Computer Science

Abstract

In today's world, support ticketing systems are employed by a wide range of businesses. The ticketing system facilitates the interaction between customers and the support teams when the customer faces an issue with a product or a service. For large-scale IT companies with a large number of clients and a great volume of communications, the task of automating the classification of incoming tickets is important for customer relationships and ensuring business growth.

Although the problem of text classification has been widely studied in the literature, the majority of the proposed approaches revolve around state-of-the-art deep learning models. This thesis addresses the following research questions: What are the reasons behind employing black box models (i.e., deep learning models) for text classification tasks? What is the level of polysemy (i.e., the coexistence of many possible meanings for a word or phrase) in a technical (i.e., specialized) text? How do static word embeddings like Word2vec fare against traditional TFIDF vectorization? How do dynamic word embeddings (e.g., PLMs) compare against a linear classifier such as Support Vector Machine (SVM) for classifying a domain-specific text?

This integrated article thesis aims to investigate the aforementioned issues through five empirical studies that were conducted over the past four years. The observation of our studies is an emerging theory that demonstrates why traditional ML models offer a more efficient solution to domain-specific text classification compared to state-of-the-art DL language models (i.e., PLMs).

Based on extensive experiments on a real-world dataset, we propose a novel Hybrid Online Offline Model (HOOM) that can efficiently classify IT Support Tickets in a real-time (i.e., dynamic) environment. Our classification model is anticipated to build trust and confidence when deployed into production as the model is interpretable, efficient, and can detect concept drifts in the data.

Keywords: Customer Support Tickets, Static Word Embeddings, Hierarchical Text Classification, Pre-trained Language Models, Machine Learning, Domain-Specific Datasets, Natural Language Processing, Overlapping Classes, Rule-Based Learning.

Summary for Lay Audience

According to a recent study, 96% of unhappy customers don't complain, and 91% of those will simply leave and never come back. In the IT business, when customers have issues with the systems they are using, they submit a 'support ticket'. A 'Support Ticketing System' is the term used to describe the way customers interact with the support agents to get their issues resolved. For large IT firms, support agents deal with a tremendous volume of support tickets daily. Handling these tickets manually is almost impossible, so the need to automate the process of organizing these tickets into different categories becomes crucial. This is called Text Classification (TC), which is one of several Natural Language Processing (NLP) tasks.

Due to the complexity of the unstructured nature of human language, TC is challenging. Recently, a suite of deep learning models called Pre-trained Language Models (PLMs) have been used extensively for all NLP tasks, including TC. These PLMs have achieved striking success in the NLP field where they are trained on an enormous amount of text (e.g., books, Wikipedia, etc), which enables these models to better understand the language. However, despite their impressive performance, we argue against the need to employ PLMs for TC tasks, especially when the text is domain-specific (i.e., related to a specialized domain such as IT).

Based on this, we pose the key research question: Are PLMs the most cost-efficient solution for domain-specific TC tasks?. The findings of our study suggest that the problem of classifying domain-specific can be addressed efficiently using old traditional classifiers such as SVM and a vectorization technique such as TFIDF that do not involve the complexity found in neural network models such as PLMs.

This thesis proposes a novel hybrid approach to classify IT Support Tickets using a non-deep learning approach that combines a static ML model trained in an offline setting with an online ML model trained in a dynamic (real-time) environment. Our classification model is anticipated to build trust and confidence when deployed into production as the model is efficient and can detect data changes that occur over time.

To my loving dad, Essam,
to whom I owe everything.

Acknowledgements

"One day I'll be writing my thesis acknowledgments with tears of pride", that's what I've been telling myself whenever I feel overwhelmed with being a full-time mother and a full-time Ph.D. student. Here I am writing the acknowledgments and feeling so proud of myself. Despite how tough the journey was, I do not regret any moment of this journey, I honor every struggle and every tear.

First and foremost, I must admit that I was lucky to have Professor Nazim Madhavji as my research supervisor. He fully respected the fact that I am a mother of two young children and showed empathy whenever one of them gets sick. He understood that I have a life outside of school. The amount of effort he puts into reviewing anything being sent to him was incredible. His comments and feedback would make anyone a great technical writer. All this is embedded in an ever-smiling personality. I could not have dreamt of a better supervisor. I will forever be grateful to you Professor.

Dr. Darlan Arruda, I am grateful for your immediate responses to whatever comes to my mind as a fresh PhD candidate with lots of doubts. Thank you for always supporting me whenever I feel down. I would also like to thank my close and faithful friend Priyanka. Our long chats and your comforting words during stressful times would not be forgotten. Thanks for being my photographer during the first year and helping me record some precious moments of my journey.

A special thanks to my friend and lab partner, Marios-Stavros Grigoriou. Your kind and cheerful character did make the lab a positive working environment for everyone. Thanks for wasting your time to solve my silly Python errors while having your own errors to worry about. Thanks for offering your medical advice whenever I start melting down because my son is sick.

Last but not least, I would like to thank my husband Mohsen. Without you, this journey would be impossible. I would also like to thank my mother Maha for making a yearly trip to help me with the kids. Thank you for being such a lifesaver during overwhelming times for me and Mohsen.

Table of Contents

List of Figures

List of Tables

Abbreviations

TC	Text Classification
IT	Information Technology
DL	Deep Learning
LR	Logistic Regression
ML	Machine Learning
CL	Continual Learning
LL	Lifelong Learning
CF	Catastrophic Forgetting
NLP	Natural Language Processing
PLM	Pre-trained Language Model
SOTA	State-of-the-art
SVM	Support Vector Machines
TFIDF	Term Frequency Inverse Document Frequency
HTC	Hierarchical Text Classification
EDA	Exploratory Data Analysis
PAC	Passive Aggressive Classifier

Chapter 1: Introduction

When confronted with two or more competing theories that are supposed to explain the phenomena, one should favor the simplest approach. – William of Ockham.

As the volume of information available on the Internet increases, there is a growing interest in developing tools to rapidly find, filter, and better manage these electronic resources. Text classification (a task of classifying text, e.g., tweets, news, and customer reviews) into different categories (also referred to as *tags*) is a crucial aspect of information organization and management.

With the ubiquity and volume of available data, the need to fully automate text classification methods becomes vital; otherwise, the data soon becomes unmanageable. In IT service management (the broad context of this thesis), text classification can be applied for many purposes, one of which is classifying IT support tickets into different categories organized in a hierarchy. A support ticket describes an issue faced by the customer that is submitted as a bug report to the IT support team. Support agents then spend a significant amount of time manually classifying incoming tickets; there is no reference to best practices based on historical data.

In today's world, support ticketing systems are employed by a wide range of businesses. The ticketing system facilitates the interaction between customers and the support teams when the customer faces an issue with a product or a service. However, for large-scale IT corpora with hundreds of classes organized in a hierarchy, the task of accurately classifying the classes at the higher levels in the hierarchies is critical for preventing the propagation of errors down to the lower levels of the hierarchy. Besides, as the number of classes increases, the possibility of overlapping between the classes also increases. Overlapping classes may occur when an incoming ticket appears as a valid classification for more than one class.

A current trend in the Natural Language Processing (NLP) community is towards employing huge deep learning pre-trained language models (PLMs) for almost any kind

of NLP task [Yoon et al., 2020; Zhou et al., 2020; Nguyen et al., 2020]. These models are also known as transformer-based models (e.g., BERT). Examples of NLP tasks are: question answering, sentiment analysis, and text classification.

This trend is stimulated by the prevalence of ‘Leaderboards’. A leaderboard is the main component of machine learning competitions that are hosted by large companies such as Netflix or popular online platforms such as Kaggle [Blum and Hardt, 2015]. The ‘Leaderboard’ ranks the best submissions for the participating teams by their accuracy scores (i.e., classifier’s performance). Recently, NLP leaderboards are dominated by PLMs which achieve state-of-the-art (SOTA) results on several benchmarks such as GLUE [Wang et al., 2018] or individual datasets such as SQuAD [Rajpurkar et al., 2016].

Another reason why deep-learning (DL) models are favorable in the NLP community is that they do not require feature engineering (e.g., pre-processing) as this step is integrated into the model fitting process. Figure 1-1 [Alzubaidi et al., 2021] shows the difference between traditional Machine Learning (ML) models and DL models for a text classification task.

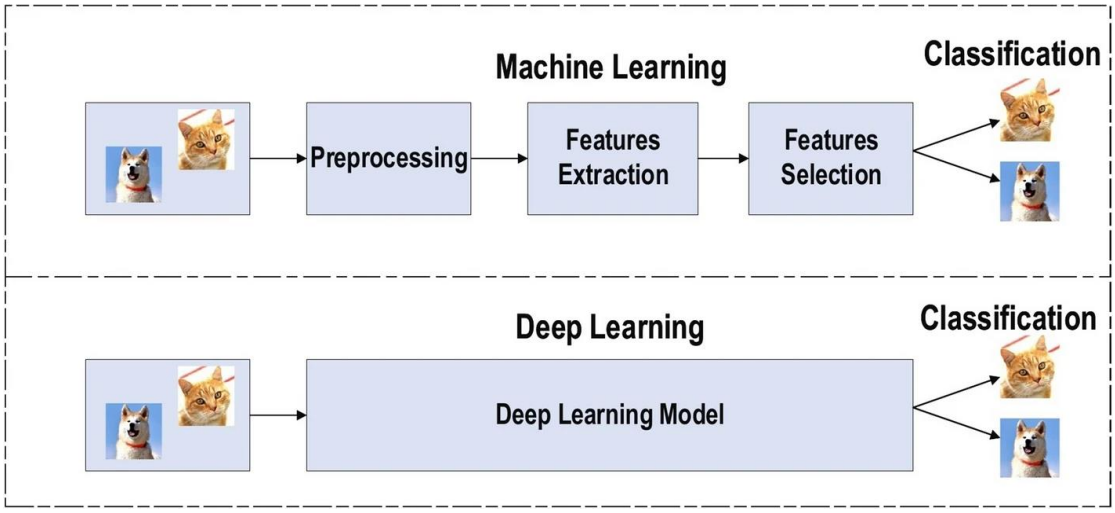


Figure 1-1: The difference between deep learning and traditional machine learning (Alzubaidi et al., 2021, p.7)

Despite the widespread use of PLMs and their impressive performance in a broad

range of NLP tasks, there is a lack of a clear and well-justified need to as why these models are being employed for domain-specific text classification tasks [Chalkidis et al., 2020; Blinov et al., 2020; Zhao et al., 2021], given the following:

- Most text classification problems are linearly separable [Joachims, 1998; Tong and Koller, 2001].
- The large gap between the pre-training cloze-style formulation and objectives (e.g., predict target words) and the downstream objectives (e.g., classification) limit the ability to fully utilize the knowledge encoded in PLMs [Han et al., 2021].
- The level of polysemy in domain-specific (i.e., specialized) text is low because scientific terms need a precise meaning in order to function and be easily recognized [Wielgosz, 2017], defeating the purpose of contextualized embeddings that aim to capture word polysemy and provide more than one embedding for a single word.
- Domain-specific terms are challenging for PLMs since there are few statistical clues in the underlying training corpora [Bollegala et al., 2015; Pilehvar and Collier, 2016].

Our work with IT support agents for a large industrial IT partner to classify customer support tickets has shed light on two main real-world concerns these large corporations face with DL-based models. The first concern is *reproducibility* which creates trust and credibility with the ML model. A recent literature survey [Pham et al., 2020] reveals that the reproducibility of DL models remains a major concern. Due to the randomness of the hyperparameters and weights used in the training stage for DL models and non-determinism in the hardware (i.e., computing resources like GPUs), it is challenging to reproduce these models [Chen et al., 2022; Pham et al., 2020].

The second concern is the *interpretability* of the results. While the field of eXplainable Artificial Intelligence (XAI) has regained the attention of researchers over the past few years [Lundberg and Lee, 2017; Ribeiro et al., 2016; Fong and Vedaldi, 2019], the explanations they provide are not accurate (i.e., low fidelity) [Rudin, 2019]. Cynthia Rudin [Rudin, 2019] argues that if the explanations were completely faithful to what the original model computes, we would not need the original model in the first place and the explanations should suffice.

The proposed approach and models used in this thesis are towards satisfying the needs of the support agents and business stakeholders by providing them with an interpretable, high-accuracy, and less-expensive model that could be easily reproduced.

Following Occam's razor, we propose a novel Hybrid Online Offline Model (HOOM) to classify hierarchical domain-specific text with overlapping classes. The hybrid model combines a static ML model trained in an offline setting with an online ML model trained in a dynamic (real-time) environment.

The offline model is based on a linear SVM classifier and a rule-based algorithm that relies on a set of handcrafted rules based on external knowledge. External knowledge incorporates the most important features (i.e., words) that contribute to the learning process. That knowledge is based on: (1) domain expertise from the support agents and (2) a Python library that highlights important features based on the chosen ML classifier(s).

The online ML model is based on a Passive Aggressive Classifier (PAC), first proposed by Crammer [Crammer et al., 2006]. This classifier belongs to a family of margin-based online learning algorithms, that can handle large datasets.

For our work to be reproducible, we provide the code and the computational environment. However, for the datasets, we only provide the three generic datasets used in this study. Due to a confidentiality agreement with our industrial partner, we are not able to provide their dataset of support tickets.

1.1 Thesis Architecture

This thesis is documented in the "integrated-article" format[1]. This format reports each discrete study (i.e., research paper) in a separate chapter (Chapters 2 to 6). Following these chapters is a chapter (Chapter 7) reflecting on the previous chapters.

The key outcome of this reflection chapter is an emerging theory as a singleton contribution of this thesis to the body of knowledge. Lastly, Chapter 8 concludes the thesis and describes future work. The following diagram (Figure 1-2) shows the thesis

[1] https://grad.uwo.ca/resources/regulations/8.html#8321

architecture. The upper (non-leaf) layers of the architecture depict conceptual layers to give context to the leaf layer that represents concrete chapters of the thesis.

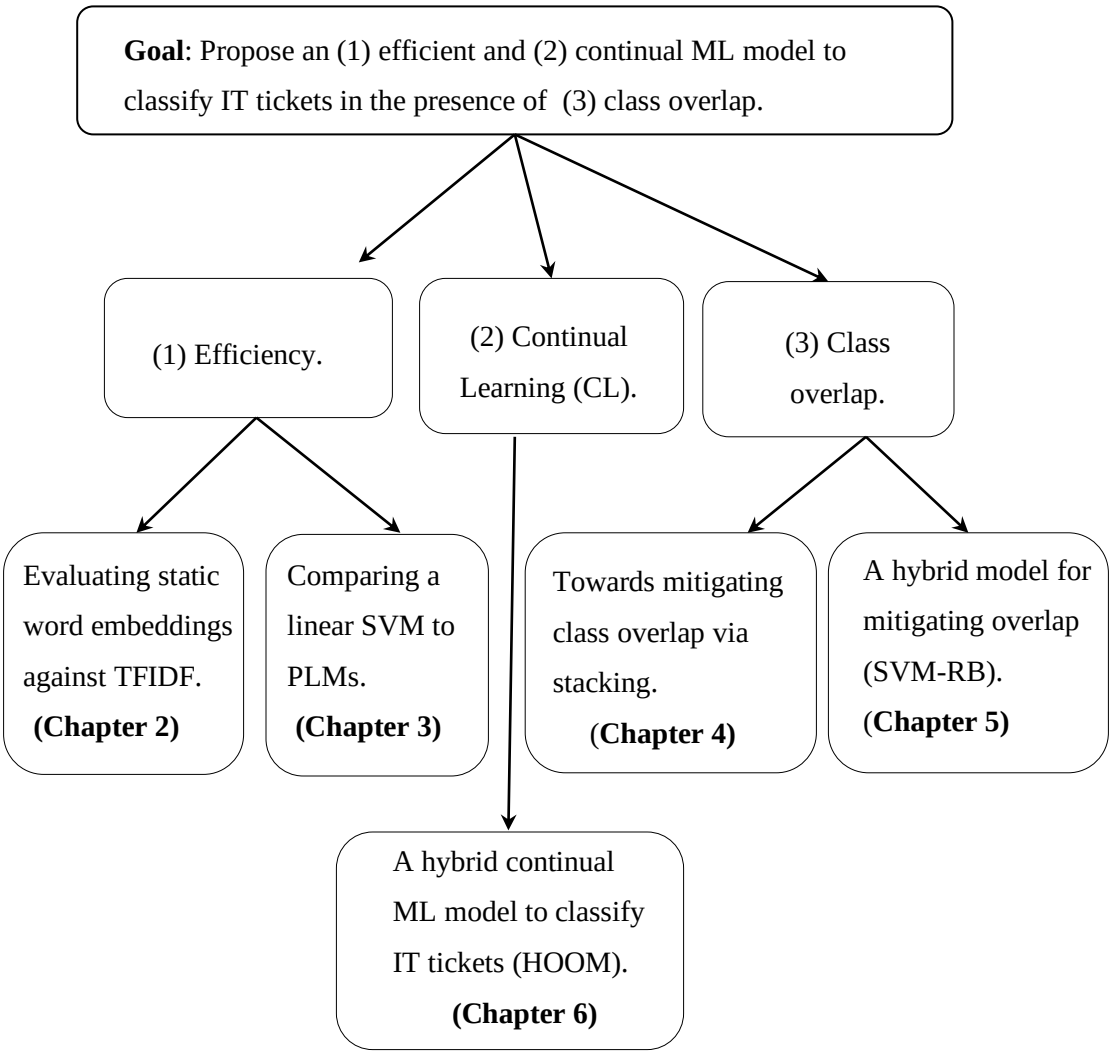


Figure 1-2: Thesis architecture

The thesis architecture (Figure 1-2) defines the research goal (numbered 1 to 3: Efficiency, Continual learning, and Class overlap, represented by the root node). The root node is decomposed into sub-goals that translates to individual chapters.

In Chapter 2, we start by exploring an efficient way of vectorizing specialized text with domain-specific words. Thus, in Chapter 2 we compare static word embeddings (e.g., Word2vec) against traditional bag-of-words models such as TFIDF for domain-

specific text classification. Chapter 3 compares state-of-the-art dynamic word embeddings (e.g., BERT) against TFIDF. Chapters 2 and 3 address the problem of classifying support tickets using efficient (i.e., low computational complexity) and simple methods.

After text vectorization (i.e., the first step in text classification), we explore a challenging issue, that of overlapping classes. An attempt to mitigate the problem of overlap is proposed in Chapter 4. This method shows a significant improvement in terms of accuracy and reducing misclassification errors. However, when testing the model with an IT support agent from the collaborating organization, it did not meet the interpretability and efficiency criteria. Hence, in Chapter 5, we successfully tackled the problem of overlap using a cheap, interpretable, and high-accuracy ML model based on a hybrid rule-based algorithm [Wahba et al., 2022] (also Chapter 5).

The next step was to deploy the proposed model into production. However, during testing on a recent dump of incoming support tickets, we observed that the taxonomy (i.e., class hierarchy) had changed from what had been agreed upon the with support agents.

In essence, for project reasons, they had introduced new classes, which resulted in a poor performance for our model. This problem is widely known as ‘concept drift’ (i.e., the problem of changing the data distribution over time) [Widmer and Kubat, 1993] and that was the motivation behind Chapter 6. Thus, in Chapter 6 we propose a hybrid model (HOOM) that can learn in a real-time environment (i.e., Continuous Learning) and can detect data drifts that evolve over time.

Together, Chapters 2-6 address the goal of the thesis: to efficiently classify domain-specific text (e.g., IT Support tickets) in the presence of class overlap in a real-time environment.

1.2 Thesis Contributions

This thesis aims to provide IT support agents with an interpretable, efficient, and reproducible ML model. Our model is anticipated to classify customer support tickets with high accuracy in a real-world scenario. The contribution of this thesis is a

combination of five empirical studies that were conducted over the last four years. The thesis contributions are as follows:

1. *Feature Engineering Phase* (Chapter 2): we studied the effectiveness of static word embedding models (e.g., GloVe) to classify IT support tickets against a traditional vectorization technique (i.e., TFIDF). The findings of this study show that traditional TFIDF provides comparable performance to static word embeddings with a low computational cost and fast training time.
2. *Feature Engineering & Model Building Phase* (Chapter 4): we propose an approach for reducing the misclassification caused by the class overlapping problem in multi-class text classification scenarios. This approach leverages the power of stacking different ML models that are trained on different pre-chosen feature subsets (i.e., feature selection). The findings of this study show that stacking can be used to tackle the problem of overlapping classes as well as increase the overall accuracy.
3. *Model Building Phase* (Chapters 3): we studied the performance of SOTA PLMs (e.g., BERT, XLM) against that of a linear SVM classifier. The findings of this study show that PLMs do not provide significant gains over the linear SVM and indicate a comparable performance for both models on text classification tasks.
4. *Model Building and Evaluation Phase* (Chapter 5): we propose a hybrid ML model (HSVM-RB) based on a set of N hand-crafted rules and a linear SVM classifier that supports hierarchical classification structures. The findings of this study show that our proposed hybrid model provides a cheap, interpretable, and efficient solution to the problem of classifying IT support tickets in the presence of class(es) overlap.
5. *Model Building and Deployment Phase* (Chapter 6): we propose a Hybrid Online Offline Model (HOOM) that combines a static ML model trained in an offline setting with an online ML model trained in a dynamic (real-time) environment. Finally, we deployed our model by building a web application using Flask[2] and Google Colab [3]for the support agents to test and validate our proposed ML model.

[2] A web application framework written in Python.

References

[Alzubaidi et al., 2021] Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M. and Farhan, L., 2021. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. Journal of big Data, 8(1), pp.1-74.

[Blinov et al., 2020] Blinov, P., Avetisian, M., Kokh, V., Umerenkov, D. and Tuzhilin, A., 2020, August. Predicting clinical diagnosis from patients electronic health records using BERT-based neural networks. In International Conference on Artificial Intelligence in Medicine (pp. 111-121). Springer, Cham.

[Blum and Hardt, 2015] Blum, A. and Hardt, M., 2015, June. The ladder: A reliable leaderboard for machine learning competitions. In International Conference on Machine Learning (pp. 1006-1014).

[Bollegala et al., 2015] Bollegala, D., Maehara, T., Yoshida, Y. and Kawarabayashi, K.I., 2015, February. Learning word representations from relational graphs. In Twenty-Ninth AAAI Conference on Artificial Intelligence (pp. 730–740).

[Chalkidis et al., 2020] Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N. and Androutsopoulos, I., 2020, November. LEGAL-BERT: The Muppets straight out of Law School. In Findings of the Association for Computational Linguistics: EMNLP 2020 (pp. 2898-2904).

---

[3] A product from Google Research that allows the execution of python code through the browser and provides access free of charge to computing resources including GPUs and TPUs.

[Chen et al., 2022] Chen, B., Wen, M., Shi, Y., Lin, D., Rajbahadur, G.K. and Jiang, Z.M., 2022, May. Towards training reproducible deep learning models. In Proceedings of the 44th International Conference on Software Engineering (pp. 2202-2214).

[Dong and Liu, 2018] Dong, G. and Liu, H. eds., 2018. Feature engineering for machine learning and data analytics. CRC Press.

[Fong and Vedaldi, 2019] Fong, R. and Vedaldi, A., 2019. Explanations for attributing deep neural network predictions. In Explainable AI: Interpreting, explaining and visualizing deep learning (pp. 149-167). Springer, Cham.

[Han et al., 2021] Han, X., Zhao, W., Ding, N., Liu, Z. and Sun, M., 2021. Ptr: Prompt tuning with rules for text classification. arXiv preprint arXiv:2105.11259.

[Joachims, 1998] Joachims, T., 1998, April. Text categorization with support vector machines: Learning with many relevant features. In European conference on machine learning (pp. 137-142). Springer, Berlin, Heidelberg.

[Lundberg and Lee, 2017] Lundberg, S.M. and Lee, S.I., 2017, December. A unified approach to interpreting model predictions. In Proceedings of the 31st International Conference on Neural Information Processing Systems (pp. 4768-4777).

[Nguyen et al., 2020] Nguyen, D.Q., Vu, T. and Nguyen, A.T., 2020, October. BERTweet: A pre-trained language model for English Tweets. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (pp. 9-14).

[Pham et al., 2020] Pham, H.V., Qian, S., Wang, J., Lutellier, T., Rosenthal, J., Tan, L., Yu, Y. and Nagappan, N., 2020, December. Problems and opportunities in training deep learning software systems: An analysis of variance. In Proceedings of the 35th IEEE/ACM international conference on automated software engineering (pp. 771-783).

[Pilehvar and Collier, 2016] Pilehvar, M.T. and Collier, N., 2016, August. Improved Semantic Representation for Domain-Specific Entities. In Proceedings of the 15th Workshop on Biomedical Natural Language Processing. ACL (pp. 12-16).

[Rajpurkar et al., 2016] Rajpurkar, P., Zhang, J., Lopyrev, K. and Liang, P., 2016, November. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (pp. 2383-2392).

[Ribeiro et al., 2016] Ribeiro, M.T., Singh, S. and Guestrin, C., 2016, August. " Why should I trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).

[Rudin, 2019] Rudin, C., 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence, 1(5), pp.206-215.

[Schick and Schütze, 2020] Schick, T. and Schütze, H., 2020, April. Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 05, pp. 8766-8774).

[Tong and Koller, 2001] Tong, S. and Koller, D., 2001. Support vector machine active learning with applications to text classification. Journal of machine learning research, 2(Nov), pp.45-66.

[Wahba et al., 2022] Wahba, Y., Madhavji, N., Steinbacher, J., 2022. A Hybrid Machine Learning Model for Efficient Classification of IT Support Tickets in The Presence of

Class Overlap, In Proceedings of the 32nd Annual International Conference on Computer Science and Software Engineering (CASCON22) (pp. 151–156).

[Wang et al., 2018] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O. and Bowman, S.R., 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In 1st Workshop on BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, co-located with the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018 (pp. 353-355). Association for Computational Linguistics (ACL). PMLR.

[Widmer and Kubat, 1993] Widmer, G. and Kubat, M., 1993, April. Effective learning in dynamic environments by explicit context tracking. In European Conference on Machine Learning (pp. 227-243). Springer, Berlin, Heidelberg.

[Yoon et al., 2020] Yoon, W., Lee, J., Kim, D., Jeong, M. and Kang, J., 2020. Pre-trained language model for biomedical question answering. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 727-740). Springer, Cham.

[Zhao et al., 2021] Zhao, Z., Zhang, Z. and Hopfgartner, F., 2021, April. A comparative study of using pre-trained language models for toxic comment classification. In Companion Proceedings of the Web Conference 2021 (pp. 500-507).

Abstract: Recently, various state-of-the-art machine learning and deep learning methods have been applied to automate the process of text classification. Because the quality of these methods highly depends on the quality of the associated "features", in this paper we focus on the "feature engineering" step in the classification process. In particular, we evaluate the effectiveness of using different static word embeddings on the accuracy of classifying IT support tickets.

Chapter 2: Evaluating the Effectiveness of Static Word Embeddings on the Classification of IT Support Tickets[4]

2.1 Introduction

Support tickets are service requests, initiated by a system's end-users when they encounter issues with their system. With a wide user-base and system issues, there will be an ongoing influx of generated support tickets. Service agents spend a large amount of time manually classifying the incoming tickets. Unfortunately, this process is complicated, and the support agents have no reference to best practices based on historical data. With the massive growth of data, incorrect routing and delays in the resolution of the issues are frequent and hence, the need to automate ticket classification becomes crucial. Based on the ticket description, the support agents determine the category of the problem and triage the ticket to the appropriate team for resolving the issue.

A typical ticket description is unstructured and hence this makes it challenging for natural language processing. Also, the ticket may contain typos as well as abbreviations which adds to the complexity. Ticket classification is an important process that ensures that tickets get routed to the right support agent. Otherwise, there can be delays,

[4] A version of this chapter has been published in (Wahba, Y., Madhavji, N.H. and Steinbacher, J., 2020, November. Evaluating the effectiveness of static word embeddings on the classification of IT support tickets. In Proceedings of the 30th Annual International Conference on Computer Science and Software Engineering (CASCON), pp. 198-206).

customer dissatisfaction, escalation to management, and reactionary fixes at high costs [Sheng et al., 2014].

Recently, neural networks and deep learning models have surpassed traditional machine learning approaches by delivering state-of-the-art results in several natural language processing (NLP) tasks, including spam filtering, sentiment analysis, and question answering. Hence, these models have become a favorable choice for any text classification or clustering task. However, this comes with the cost of increased computational complexity and therefore increased model training time [Fu and Menzies, 2017].

Word embeddings are one of the popular uses of neural networks for handling natural language text. These embeddings are able to place words in a vector space that contains semantic information about the words. Thus, similar words will be placed close to each other. Capturing word semantics in different contexts is what differentiates between a static and a dynamic word embedding.

Because the quality of these methods highly depends on the quality of the associated "features", in this paper we focus on the "feature engineering" step in the classification process. In particular, we evaluate the effectiveness of using different static word embeddings on the accuracy of classifying IT support tickets. To our knowledge, no work has compared the performance of these word embeddings against old methods like bag-of-words. Thus, the key question being addressed in this paper is: How effective is using static word embeddings in the task of IT ticket classification?

The experimental results show that the traditional Term Frequency Inverse Document Frequency (TFIDF) bag-of-words along with Support Vector Machines (SVM) provides competitive results and sometimes outperforms static word embedding models such as word2vec while maintaining low computational cost. Overall, the findings of this study suggest that the problem of classifying IT support tickets can be addressed efficiently using old traditional methods such as TFIDF bag-of-words that do not involve the complexity found in neural network models.

The rest of the paper is organized as follows. Section 2.2 describes the background. Section 2.3 describes related work. Section 2.4 describes our project context. Section 2.5

describes the methodology and Section 2.6 presents the research results. Section 2.7 concludes the paper.

2.2 Background

The quality of ML or DL model comes from extensive feature engineering than from the learning technique itself, as the quality of these methods highly depends on the quality of available features [Dong and Liu, 2018]. To apply machine learning algorithms, human text must be converted to numeric form through what is known as vector representation [Orsenigo et al., 2018].

Handling vector representations is of the challenges of natural text processing. This is because the same set of words can convey different meanings in different contexts or if given in a different order. This is known as polysemy, which is the association of one word with two or more distinct meanings [Sennet, 2014]. This level of sophistication in understanding text and coming up with the best vector representation for words is why word embeddings emerged in this research direction as an alternative to the bag-of-words (BOW) vector model [Harris, 1954].

The core idea behind word embeddings is that words that are used in similar contexts will be given similar representations, thus capturing word semantics. Two of the popular word embeddings that attracted many researchers are Word2Vec [Mikolov et al., 2013] trained on Google News, and Glove [Pennington et al., 2014] which is trained on Wikipedia. These methods generate word vectors by training the word embedding algorithm against a huge corpus of text. However, these embeddings are referred to as ‘static’, in the sense that each word is represented by only one vector regardless of the context. Thus, the word bank in “I went to the bank to withdraw money before going fishing at the riverbank “will have the same embedding. To mitigate this problem, dynamic representations or so-called contextualized embeddings emerged as a replacement for static word embeddings and improved many NLP tasks [Liu et al., 2019; Devlin et al., 2018; Yao et al., 2018]. These embeddings aim to capture word semantics in different contexts to address the issue of the context-dependent nature of words.

2.3 Related Work

Since our work aims to investigate the effectiveness of using word embeddings to classify IT support tickets, we first give an overview of the existing literature studies on the problem of ticket classification, and then we examine some of the studies on the effectiveness of domain-specific word embeddings.

Diao [Diao et al., 2009] leveraged large expert communities with domain knowledge to develop a rule-based approach, where experts author the classification rules to classify problem tickets. Paramesh's [Paramesh et al., 2018] followed the ensemble approach to improve the accuracy of their ticket classifier system, by combining the predictions of Bagging, Boosting, and Voting ensemble on four base classifiers. Similarly, the work in [Xu et al., 2016] tackled the problem using an ensemble of SVM classifiers and re-sampling techniques to handle the problem of data imbalance.

Authors in [Paramesh and Shreedhara, 2019] investigated different classification algorithms to classify incident tickets, SVM was reported to perform well on all data samples. However, [Son et al., 2014] reported Multinomial Naive Bayes (MNB) to outperform Softmax Regression Neural Network (SNN) for classifying help desk tickets.

In contrast to 'Flat' text classification, hierarchal classification has also been addressed. Authors in [Cai and Hofmann, 2004] proposed a novel architecture for hierarchical classification that extends the strengths of SVM classifiers to leverage prior knowledge about class relationships. While authors in [Zeng et al., 2017] investigated hierarchal multi-label classification of incident tickets by leveraging the known hierarchical relationship between categories using a novel greedy algorithm 'GLabel' to label the predicting ticket. Adding to the previous work, authors in [Zeng et al., 2014] proposed an algorithm to utilize the knowledge from domain experts. Note that all these papers focus on the final stage of the text classification pipeline, which is model building and machine learning algorithms.

With the introduction of static word embeddings in 2013 by Mikolov [Mikolov, Chen et al., 2013] that leveraged neural networks, Natural Language Processing (NLP) tasks have changed dramatically. Accordingly, text classification methods were classified into those which use neural networks and the ones that do not. Authors in [Lyubinets et al., 2018] reported that recurrent neural networks (RNNs) using word embeddings data

outperform the classic solutions for the task of classifying data from customer service systems and task trackers. Similarly, the work in [Han and Akbari, 2018] leveraged deep networks, where a convolutional neural network (CNN) was reported to achieve the best performance for the task of classifying IT tickets without much feature engineering. However, authors in [Lilleberg et al., 2015] achieved an improved classification accuracy using a linear support vector machine (SVM) along with the term weighted Word2Vec model. In contrast to using pre-trained word vectors, authors in [Rabut et al., 2019] provided additional semantic information by enriching the vectors with Part-of-Speech (POS) tags.

Despite the success of general domain word embeddings like Word2Vec in many NLP tasks, domain-specific terms always represent a challenge, since these embeddings are trained over general corpora like books or Wikipedia. Some researchers suggested fusing domain-specific data with general data for better performance [Yen et al., 2017; Wu et al., 2017]. While the work in [Efstathiou et al., 2018] introduced 'SO_Word2Vec', a domain-specific word embedding that is trained over 15GB of textual data from Stack Overflow posts. Similarly, authors in [Roy et al., 2019] presented Annotation Word Embedding (AWE) which incorporates different kinds of domain knowledge. The model's performance outperformed state-of-the-art baselines on two cybersecurity applications. The work in [Risch and Krestel, 2018] reported an increase of 17 percent in accuracy compared to state-of-the-art methods when using a domain-specific word embedding to classify patent applications.

Upon critical analysis of the literature, we note that it is not clear at all how effective static word embeddings are in solving the task of IT support ticket classification. This problem has a caveat that it contains IT-related terminologies (e.g., mongoDB, kubernetes, and logdna) and unique fragments of text (e.g., HTML code, IP addresses, XML code) and specific abbreviations (e.g., paas, vlan, and iam). We note the current trend of using neural and deep learning architectures for solving text classification problems. This imposes us to think about whether it is worth using sophisticated and computationally expensive neural or deep learning architectures for the task of classifying support tickets.

This was thus a motivation for us to investigate the usefulness of word embeddings over the simple TFIDF in solving the IT support ticket classification problem.

2.4 Project Context

This section describes the nature of our dataset. This is followed by an analysis of the problem context in Section 2.4.2

2.4.1 Dataset

Our dataset is considered a large-scale dataset containing over 1.6 million support tickets classified into 32 different ticket categories. For customers to submit a new ticket, they have to give a short and full description of their issue. We noted that predominantly only the short description field is used (as shown in Table 2-1). This problem is handled in the pre-processing stage by concatenating both fields into a new one. Also, the description entered by the customer is unstructured containing non-English characters, dates, and typos.

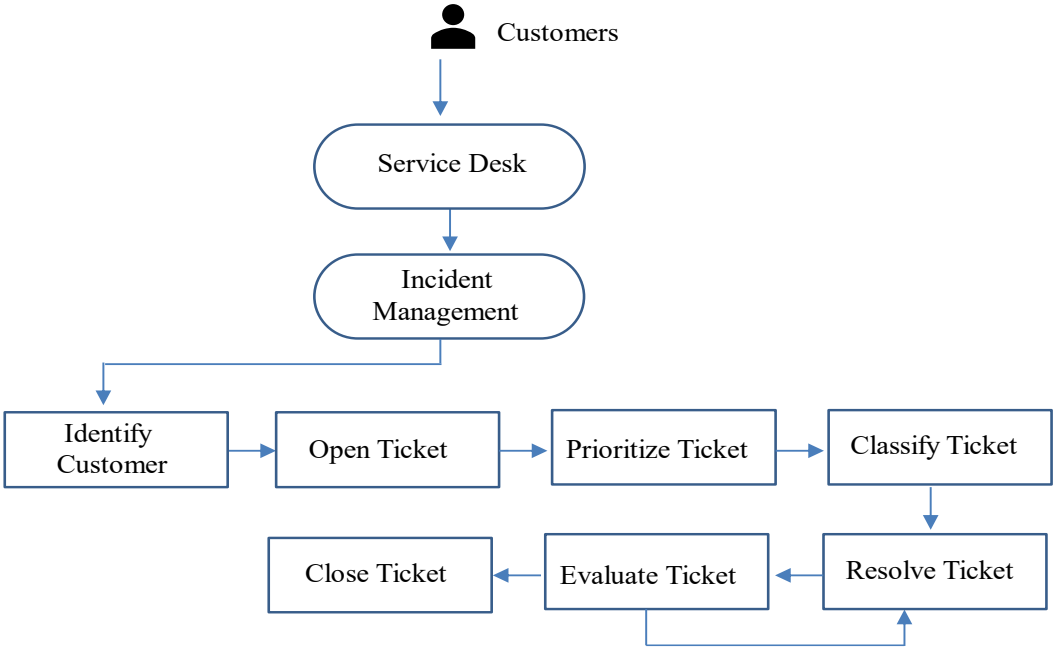
Table 2-1: Example snapshot of the dataset

DESCRIPTION	CATEGORY	SHORT_DESCRIPTION	DV_CATEGORY
	2	Virtual Server Cancellatio	Infrastructure
	4	Mass Data Migration - Po	Project Office
	2	Virtual Server Cancellatio	Infrastructure
	2	[ABNS] ë°ì´í„° ì „ì†¡ë Ÿ‰ ì	Infrastructure
	2	Payment is late	Infrastructure
	2	Test	Infrastructure
	2	Server Cancellation - 08/1	Infrastructure
	2	ì‹ ìš©ì¹´ë“œ ë§Œê¸°	Infrastructure
	2	Virtual Server Cancellatio	Infrastructure

2.4.2 Problem Analysis

In a typical IT organization, customers raise an issue (i.e., open a ticket), through the IT service desk. IT service management (ITSM) is responsible for dealing with the resolution of these tickets. Figure 2-1 depicts the standard process of incident

management that starts with ticket generation, which is followed by prioritization, categorization, and then a resolution of the ticket by an IT specialist. If the customer is satisfied, then the ticket is closed.



Legend: Ellipse – entity; Rectangle – task; Arrow – flow.

Figure 2-1: Process flow of IT service management

As can be seen from the processing pipeline, classification plays a substantial role. Wrong manual ticket classification will prevent the tickets from being triaged to the appropriate support team. In turn, this can lead to the problem of time delays in ticket resolution, violation of service-level agreements, and customer dissatisfaction.

Thus, ML-based methods for automation are considered crucial for the overall incident management efficiency. Millions of support tickets can be sorted in a fraction of the time spent manually for this task, thus freeing the agents to focus on more important or other tasks. In addition to reducing the number of escalations triggered by unhappy customers.

Ticket classification is one of the use cases of document classification where the ticket's description submitted by customers represents a document and the ticket category is the document label. Therefore, the steps for classifying a support/issue ticket are the same steps followed in a typical document classification problem.

The following figure (Figure 2-2) shows the main steps for a text classification model.

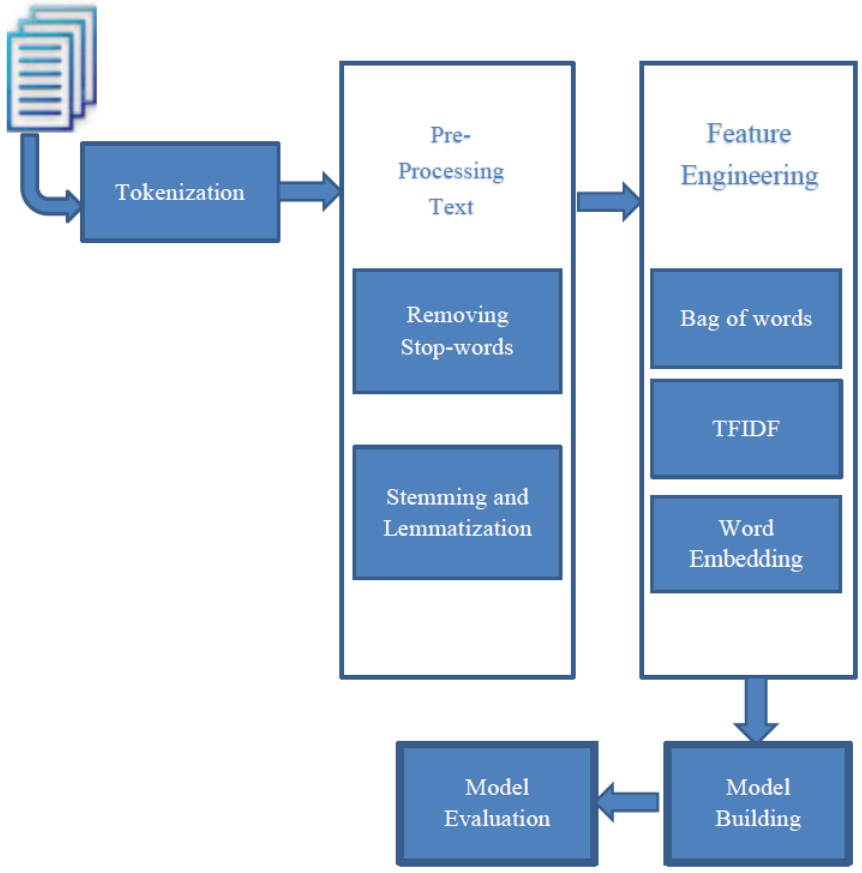


Figure 2-2: Text classification steps

There are a few types of text classification based on the number of classes/categories to predict:

- Binary classification: When the total number of classes is two, any prediction can contain either one of those classes.
- Multi-class classification: Involves classifying instances into more than two classes, where each instance can be classified into one of those classes.
- Multi-label classification: Involves classifying instances into more than two classes, where each instance can be classified into one or more categories at the same time.

Our work is considered a multi-class classification task, where the support tickets are classified into 32 different ticket categories (e.g., Infrastructure, Project Office, Sales, Databases, etc. – please see later in Figure 2-3 for more).

2.5 Methodology

This section gives an overview of the dataset we used in our study and the pre-processing steps performed to clean the data. This is followed by the experimental steps and the word embeddings used in this study.

2.5.1 Dataset Preparation

The first stage in building a text classification model is cleaning the data (the data pre-processing stage). This stage aims to reduce the vocabulary size and remove noise found in the input documents. This is anticipated to help in maximizing the classifier's performance [Krouska et al., 2016; Barushka and Hajek, 2019].

For a natural language text, noise can be spelling errors, abbreviations, character repetitions, missing punctuations, non-standard words, etc. In our work, we applied the regularly used operations in text mining in addition to domain-specific operations that we perform based on the ticket descriptions and domain experience from the support agents of our industrial partner. Given the ticket structure in Table 2-2, we are only interested in the ticket description and its corresponding category; all other fields are thus ignored.

Table 2-2: Typical support ticket data

Ticket number	**Ticket category**	**Ticket priority**	**Ticket state**	**Ticket description**
CS177	Services	Medium	In Progress	IP Billing address missing
CS190	Infrastructure	High	Open	Payment late

The following are the common pre-processing tasks that we carried out in this order:

1. Removing missing data.
2. Removing numbers and special characters.
3. Converting text to lowercase
4. Word tokenization.
5. Removing stop words.
6. Lemmatization.
7. Removing non-English words.

While we applied such operations, we also found the need to employ some domain-specific steps. For example, upon careful examination of our ticket descriptions, we noticed the presence of Chinese characters. Hence, we performed the regular step of non-English words removal. A side-effect was that some important domain-specific words were removed in the process. Thus, we created a list of words that could have an impact on ticket classification and called it the 'to_keep' list. For this purpose, we incorporated domain knowledge from our support agents along with some common knowledge of some IT terminologies. For example, words such as "Watson" and "Vmware" are kept and not removed during the pre-processing step.

Since the focus of this research is more on feature engineering and pre-processing steps. We carefully examined the list of discarded words during the step of non-English words removal, and, to our surprise, we found a huge list of common English words. For example, words such as groups, questions, requests, and chatbot were removed.

There are two reasons behind this. First, the "Words" Corpus from NLTK [Bird et al., 2009a] that we used is a delimited list of dictionary words, hence, words are stored in their singular form. Second, this Corpus is not an exhaustive list of all English words, so some words might be missing [Bird et al., 2009b] (e.g., blog, chatbot). To mitigate the first problem, we performed lemmatization which ensures that words are kept in their dictionary or base form, known as a "lemma". This step is done prior to removing non-English words. While for the second problem, we added the missing words to our

"to_keep" list and used another lexical database for English called "WordNet" which is published by Princeton University [Miller, 1995].

Our dataset described in Section 2.4.1 suffers from a severe imbalance, where the distribution of class samples is uneven by a large amount in the training dataset (e.g., 1:100 or more) as shown in Figure 2-3. This imbalance makes the classification algorithm biased toward the major categories and ignores the minor ones, leading to poor classification for these classes [Akkaradamrongrat et al., 2019; Wang and Zhang, 2018].

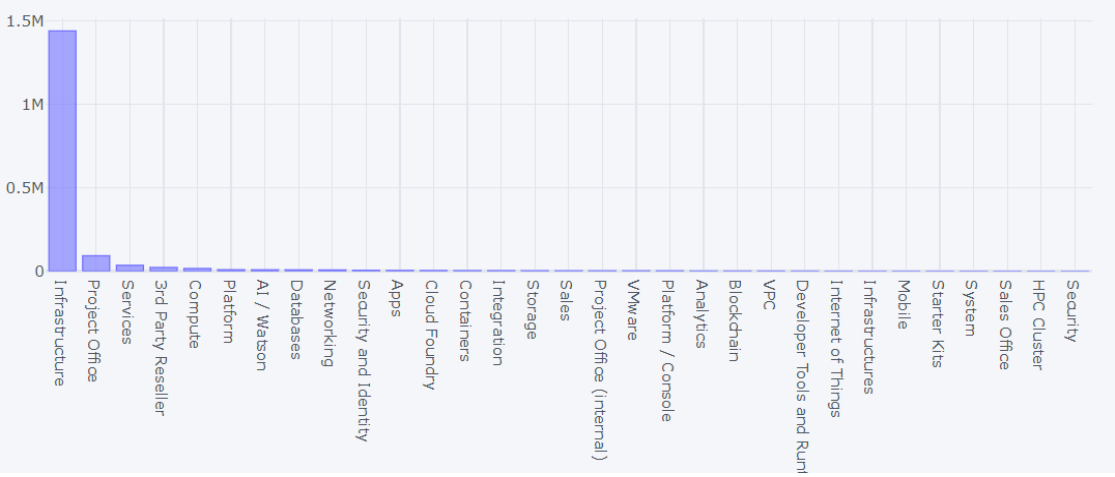


Figure 2-3: Distribution of classes and the severe imbalance

There are several approaches for handling this imbalance, and they can be grouped into four categories [Ma and He, 2013]: (i) algorithm-level, (ii) data-level, (iii) cost-sensitive, and (iv) ensemble learning. Since our work is focused on the pre-processing stage, 'data-level' approaches such as oversampling techniques [Chiamanusorn and Sinapiromsaran, 2017; Zhu et al., 2017] and undersampling [Yap et al., 2014] are more relevant to our purpose. However, these methods have major drawbacks [Ma and He, 2013] and are sometimes reported to be ineffective, and may often cause negative effects on multiclass tasks [Zhou and Liu, 2005]. Hence, we decided to keep the original distribution while in the future we intend to gather more data for the minor classes.

2.6 Empirical Study

In this section, we describe the empirical study that we conducted. In particular, we describe data characteristics, the infrastructure used, the word vectorization models used, and the performance measures.

We collected over 1.6 million tickets from a large cloud-based ticketing system, classified into 32 different categories. Our experimental algorithms are written in Python 3.8.3. The testing machine is Windows 10 with an Intel Core i7 CPU 2.71 GHz and 32GB of RAM.

The following are the different word vectorization models used in this study:

1. GN_Word2Vec [Mihálta, 2016]: This is a neural network–based implementation that is provided by Google and is trained on a part of the Google News dataset (about 100 billion words). The model contains 300-dimensional vectors for 3 million words and phrases.

2. SO_Word2Vec [Efstathiou, 2018b]: This is a domain-specific Word2Vec model that is trained on Stack Overflow posts which is a generic model of Software Engineering knowledge containing 200-dimensional vectors.

3. CO_Word2Vec: This is the Word2Vec algorithm trained on our corpus of support tickets using a size of 100-dimensional vectors. Thus, we call it Corpus Word2Vec (CO_Word2Vec).

4. TFIDF[5]: This is the simplest yet powerful technique for vectorizing text documents [Sarkar, 2016].

An important parameter that we considered when applying the TFIDF vectorizer is N-grams. An 'N-gram' is simply a sequence of N words that predicts the occurrence of a word based on the occurrence of its (N – 1) previous words. Unigrams or single words

[5] TFIDF stands for Term Frequency-Inverse Document Frequency, which is a combination of two metrics:
1. Term frequency (*tf*): a measure of how frequently a term, t, appears in a document, d.
2. Inverse document frequency(*idf*): a measure of how important a term is. It is computed by dividing the total number of documents in our corpus by the document frequency for each term and then applying logarithmic scaling on the result.

are the default setting. In our study, we set ngram_range to (1,3) which means that we included feature vectors consisting of all unigrams, bigrams, and trigrams.

For the machine learning models we chose two popular and simple classification algorithms:

1. Support Vector Machines (SVM): reported as one of the best algorithms for text classification [Joachims, 1998; Telnoni et al., 2019].

We chose the LinearSVC algorithm in the Scikit-learn library [Pedregosa et al., 2011a]. The reason is that this algorithm implements "one-vs-the-rest" or what is known as a one-versus-all (OVA) multi-class strategy, which is suitable for high dimensional data and, has a very low running time [Chauhan et al., 2019].

2. Logistic Regression (LR): a simple linear classifier that uses maximum likelihood for estimation method [Pedregosa et al., 2011b].

To evaluate the performance of the above-mentioned two classification algorithms, we used the standard information retrieval (IR) measures, Precision[6], Recall[7] , and F-measure[8] or F-score. As mentioned before, our task is a multi-classification one, and our data is hugely imbalanced, so, we used the F-score metric which is the harmonic mean value of precision and recall. This measure is suitable for multi-classification tasks. However, the F-score does not take into account the True Negatives (TN) [Powers, 2015]. In our case, we give more importance to classifying rare positives and this is why F-score is a suitable measure.

2.7 Results

The experimental results obtained after experimenting with different static word embeddings are presented in Figure 2-4, which shows the weighted-average F1 score of each word embedding model evaluated using two base classifiers: Linear SVM and Logistic Regression.

---

[6] $Precision = \frac{True\ Positive\ (TP)}{True\ Positive(TP)+False\ Positive\ (FP)}$

[7] $Recall = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP)+False\ Negative\ (FN)}$

[8] $F1 - Score = 2 \times \frac{Precision \times Recall}{Precision+recall}$

Surprisingly, the traditional TFIDF model achieved a competitive accuracy of 92% using the SVM classifier and 91% using Logistic Regression. While the three static word2vec models achieved a close classification accuracy of 89% trained using the Logistic Regression classifier, however, with a high computational cost. Although SVM and LR generally have close performance (i.e., accuracy), the SVM may work better for the highly imbalanced datasets [Musa, 2013].

Also, since our dataset is highly skewed, it was expected that the classification algorithm will be biased towards the major classes, leading to a high classification accuracy for the two major ticket categories (Infrastructure & Project Office), while showing poor accuracies towards the minor ones. This is shown clearly in the detailed classification report presented in Figure 2-5.

The first column in Figure 2-5 represents the class number as given in the dataset we collected. While the last column (i.e., support) represents the number of instances for a given class. For the precision and recall values, note that some classes show zero or very

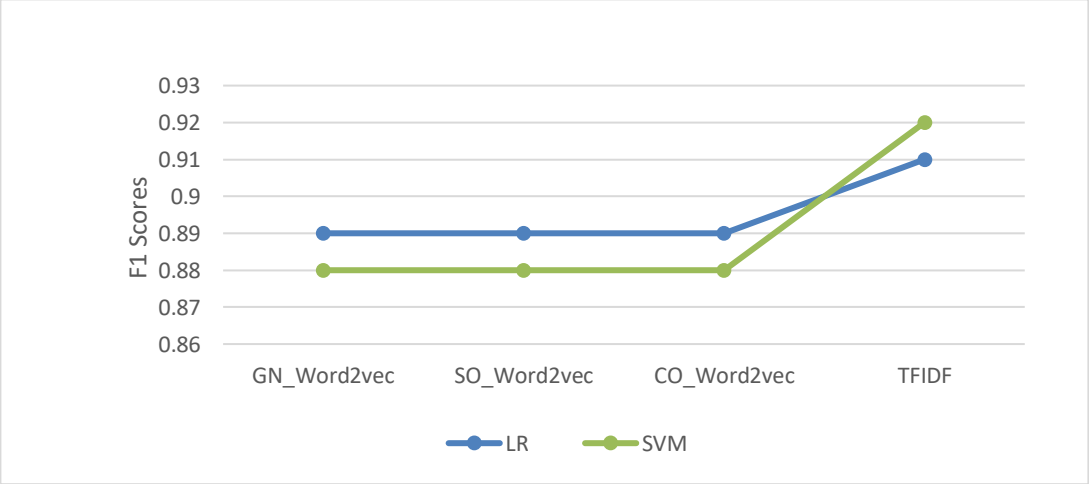


Figure 2-4: Performance of different Word2Vec embeddings versus TFIDF

low F-scores. This is because the number of instances collected for these categories was below 50 records; hence, the classification algorithm failed to classify them.

Accuracy for LinearSVC : 0.9241538025635081

	precision	recall	f1-score	support
0	0.31	0.04	0.07	222
1	0.32	0.15	0.21	2899
2	0.97	0.99	0.98	865043
3	0.44	0.45	0.44	21268
4	0.75	0.74	0.74	55678
5	0.31	0.05	0.08	1430
6	0.00	0.00	0.00	12
7	0.00	0.00	0.00	50
8	0.24	0.04	0.08	179
9	0.39	0.23	0.29	5418
10	0.56	0.56	0.56	5139
20	0.45	0.17	0.24	878
30	0.69	0.50	0.58	803
40	0.42	0.27	0.33	2520
50	0.48	0.52	0.50	9801
60	0.52	0.60	0.55	2143
70	0.60	0.64	0.62	4954
80	0.36	0.14	0.20	587
90	0.56	0.49	0.52	2097
100	0.37	0.07	0.11	223
110	0.55	0.07	0.12	87
120	0.49	0.45	0.47	4533
130	0.24	0.05	0.08	1353
140	0.80	0.53	0.64	3277
150	0.22	0.07	0.10	73
160	0.54	0.42	0.47	1682
170	0.55	0.25	0.35	1414
180	0.44	0.11	0.18	1487
190	1.00	0.05	0.10	19
200	0.68	0.11	0.18	13399
210	1.00	0.04	0.08	25
600	0.67	0.52	0.59	705
accuracy			0.92	1009398
macro avg	0.50	0.29	0.33	1009398
weighted avg	0.91	0.92	0.92	1009398

Figure 2-5: Classification report of TFIDF showing precision and recall of Linear SVM for all 32 classes

It must be noted that, when using the TFIDF model, trying different n-grams is important. In our study, we experimented with different n-grams and recorded the performance for all 32 classes. Results showed that using trigrams (1,3) enhanced the F-score of almost all minor classes.

In Figure 2-6, we describe the performance of the four vectorization models used in the study to classify each of the classes. It is clear that the imbalance problem is affecting the classifier's performance to recognize the minor classes. However, the performance of the traditional TFIDF to classify the minor classes outperformed that of the three static word embeddings. This is demonstrated in Figure 2-6(d). The performance of the static word embeddings (Figures 2-6(a), 2-6(b), and 2-6(c)) to classify the minor classes is almost the same with neglectable differences. Both classification algorithms (SVM & LR) showed very similar results, for the sake of space for this paper, we included only the results for SVM classifier.

This is demonstrated in Figure 2-6(d). The performance of the static word embeddings (Figures 2-6(a), 2-6(b), and 2-6(c)) to classify the minor classes is almost the same with neglectable differences. Both classification algorithms (SVM & LR) showed very similar results, for the sake of space for this paper, we included only the results for SVM classifier.

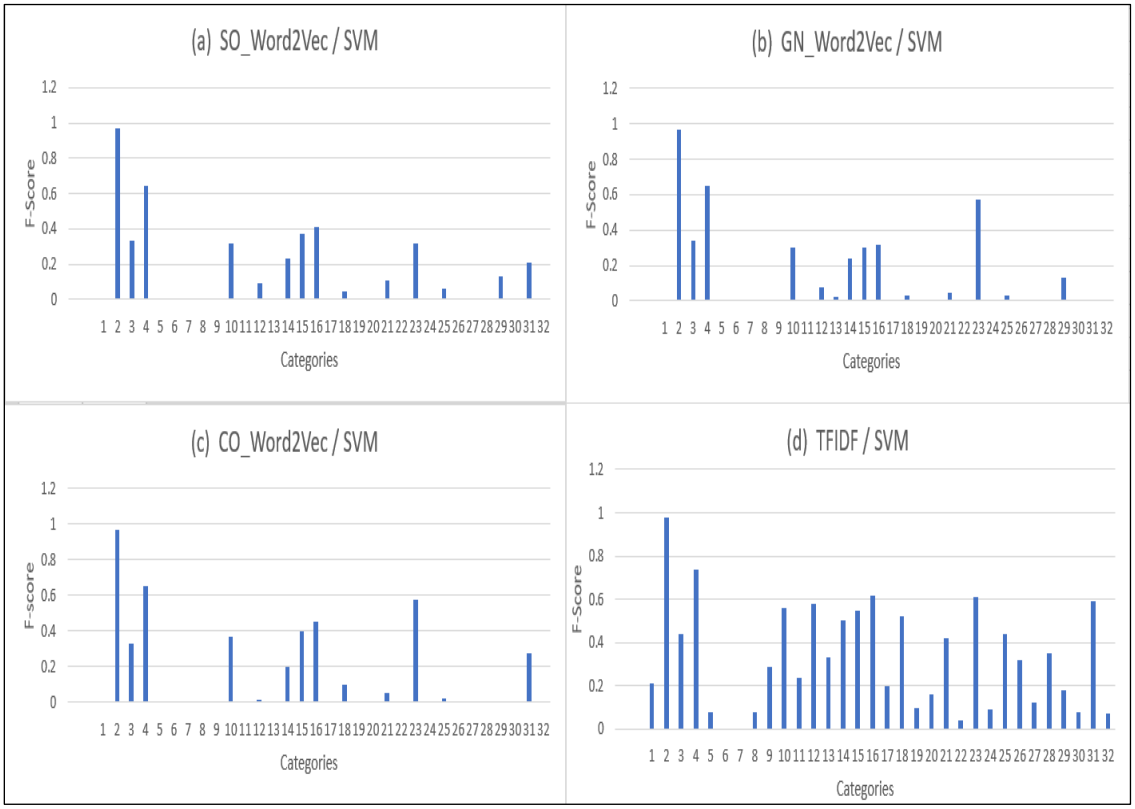


Figure 2-6: Classification accuracy of SVM using different embedding models for all 32 Classes

However, in an effort to examine the representational power of the domain-specific word embedding SO_Word2Vec versus the general word embedding GN_Word2Vec in capturing some of the ticket-specific keywords, we retrieved the top 5 similar words for some of the four frequently appearing keywords that we noticed while pre-processing our support tickets. These results are shown in Table 2-3. As can be seen from the table, the domain-specific word embedding trained on a software engineering domain (Stack Overflow), was able to capture semantically related words better than the general pre-

trained model on Google news. It's also clear that they are very effective in identifying domain-specific ambiguities.

Table 2-3: Examples of top 5 related words in SO Word2Vec and Google News model

Keyword	Most similar in SO Word2Vec	Most similar in GN Word2Vec
cloud	clound, cloud-based, azure, gcp, iaas	clouds, cloud, cloud_computing Abu_Risha_assassination
fetch	retrieve, fetching, fetched, fetches, retrieved	fetchesd, fetches, fetching, Sotheby_auction, presale_estimate
watson	nlc, nlu, stt, speech-to-text, luis	thompson, walsh, bennett, armstrong, crawford
abort	aborts, aborting, aborted, interrupted, terminate	aborting, aborted, aborts, abort, abort_fetus

One important observation to note here is that the task of classification of support tickets can be automated using simple traditional methods such as TFIDF with a high classification accuracy and a very low computational power compared to complex algorithms that are often hard to interpret. While the problem of poor accuracies for minor classes can be mitigated efficiently by collecting more data for the minor categories, or by using a closed feedback loop between the support agents and the ML algorithm, which continuously improves the model by adding new ticket information for minor classes.

2.8 Conclusion and Future Work

Classifying support tickets plays an important role in any help disk system. Automation of the tickets' classification should improve the resolution time significantly and minimize errors in the escalation process. In this paper, we describe the effectiveness of different static word embeddings including a domain-specific word embedding for the

software engineering domain (SO_Word2Vec) on the task of classifying IT support tickets of a real-world dataset.

Results showed that, unlike general document classification, IT support tickets do not benefit much from using static word embeddings. This is due to the domain-specific words that are considered as Out of Vocab (OOV) words for pre-trained embeddings. Also, the level of polysemy (i.e., the coexistence of many possible meanings for a word or phrase) in IT technical text is very low which is the reason why the traditional TFIDF bag-of-words provided comparable performance and sometimes outperformed static word embeddings with a low computational cost and fast training time. For future work, we plan to apply contextual word embeddings (e.g., BERT, ELMO) and investigate their effectiveness in improving the accuracy of our minor classes. Also, we intend to address the hierarchical classification problem of support tickets.

References

[Akkaradamrongrat et al., 2019] Akkaradamrongrat, S., Kachamas, P. and Sinthupinyo, S., 2019, July. Text generation for imbalanced text classification. In 2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE) (pp. 181-186). IEEE.

[Barushka and Hajek, 2019] Barushka, A. and Hajek, P., 2019, November. The effect of text preprocessing strategies on detecting fake consumer reviews. In Proceedings of the 2019 3rd international conference on e-business and internet (pp. 13-17).

[Bird et al., 2009a] 2. Accessing Text Corpora and Lexical Resources. 2013. Nltkorg. https://www.nltk.org/book/ch02.html. (last accessed Oct. 16, 2022).

[Bird et al., 2009b] Bird, S., Klein, E. and Loper, E., 2009. Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc."

[Cai and Hofmann, 2004] Cai, L. and Hofmann, T., 2004, November. Hierarchical document categorization with support vector machines. In Proceedings of the thirteenth ACM international conference on Information and knowledge management (pp. 78-87).

[Chauhan et al., 2019] Chauhan, V.K., Dahiya, K. and Sharma, A., 2019. Problem formulations and solvers in linear SVM: a review. Artificial Intelligence Review, 52(2), pp.803-855.

[Chiamanusorn and Sinapiromsaran, 2017] Chiamanusorn, C. and Sinapiromsaran, K., 2017, December. Extreme anomalous oversampling technique for class imbalance. In Proceedings of the 2017 International Conference on Information Technology (pp. 341-345).

[Devlin et al., 2018] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[Diao et al., 2009] Diao, Y., Jamjoom, H. and Loewenstern, D., 2009, September. Rule-based problem classification in it service management. In 2009 IEEE International Conference on Cloud Computing (pp. 221-228). IEEE.

[Dong and Liu, 2018] Dong, G. and Liu, H. eds., 2018. Feature engineering for machine learning and data analytics. CRC Press.

[Efstathiou et al., 2018] Efstathiou, V., Chatzilenas, C. and Spinellis, D., 2018, May. Word embeddings for the software engineering domain. In Proceedings of the 15th international conference on mining software repositories (pp. 38-41).

[Efstathiou, 2018] Efstathiou V. SO_word2vec [Source Code] https://github.com/vefstathiou/SO_word2vec (last accessed Oct. 15, 2022).

[Fu and Menzies, 2017] Fu, W. and Menzies, T., 2017, August. Easy over hard: A case study on deep learning. In Proceedings of the 2017 11th joint meeting on foundations of software engineering (pp. 49-60).

[Han and Akbari, 2018] Han, J. and Akbari, M., 2018, April. Vertical domain text classification: towards understanding IT tickets using deep neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 32, No. 1).

[Harris, 1954] Harris, Z.S., 1954. Distributional structure. Word, 10(2-3), pp.146-162.

[Joachims, 1998] Joachims, T., 1998, April. Text categorization with support vector machines: Learning with many relevant features. In European conference on machine learning (pp. 137-142). Springer, Berlin, Heidelberg.

[Krouska et al., 2016] Krouska, A., Troussas, C. and Virvou, M., 2016, July. The effect of preprocessing techniques on Twitter sentiment analysis. In 2016 7th international conference on information, intelligence, systems & applications (IISA) (pp. 1-5). IEEE.

[Lilleberg et al., 2015] Lilleberg, J., Zhu, Y. and Zhang, Y., 2015, July. Support vector machines and word2vec for text classification with semantic features. In 2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC) (pp. 136-140). IEEE.

[Liu et al., 2019] Liu, Y., Che, W., Wang, Y., Zheng, B., Qin, B. and Liu, T., 2019. Deep contextualized word embeddings for universal dependency parsing. ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP), 19(1), pp.1-17.

[Lyubinets et al., 2018] Lyubinets, V., Boiko, T. and Nicholas, D., 2018, August. Automated labeling of bugs and tickets using attention-based mechanisms in recurrent

neural networks. In 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP) (pp. 271-275). IEEE.

[Ma and He, 2013] Ma, Y. and He, H. eds., 2013. Imbalanced learning: foundations, algorithms, and applications.

[Miháltz, 2016] Miháltz M. word2vec-GoogleNews-vectors [Source Code] https://github.com/mmihaltz/word2vec-GoogleNews-vectors (last accessed Oct. 15, 2022).

[Mikolov, Chen, et al., 2013] Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[Mikolov, Sutskever et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26.

[Miller, 1995] Miller, G.A., 1995. WordNet: a lexical database for English. Communications of the ACM, 38(11), pp.39-41.

[Musa, 2013] Musa, A.B., 2013. Comparative study on classification performance between support vector machine and logistic regression. International Journal of Machine Learning and Cybernetics, 4, pp.13-24.

[Orsenigo et al., 2018] Orsenigo, C., Vercellis, C. and Volpetti, C., 2018, November. Concatenating or averaging? Hybrid sentences representations for sentiment analysis. In International Conference on Intelligent Data Engineering and Automated Learning (pp. 567-575). Springer, Cham.

[Paramesh and Shreedhara, 2019] Paramesh, S.P. and Shreedhara, K.S., 2019. Automated IT service desk systems using machine learning techniques. In Data Analytics and Learning (pp. 331-346). Springer, Singapore.

[Paramesh et al., 2018] Paramesh, S.P., Ramya, C. and Shreedhara, K.S., 2018, December. Classifying the unstructured IT service desk tickets using ensemble of classifiers. In 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS) (pp. 221-227). IEEE.

[Pedregosa et al., 2011a] sklearn.svm.LinearSVC. https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC (last accessed Oct. 17, 2022).

[Pedregosa et al., 2011b] sklearn.linear_model.LogisticRegression. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (last accessed Oct. 17, 2022).

[Pennington et al., 2014] Pennington, J., Socher, R. and Manning, C.D., 2014, October. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[Powers, 2015] Powers, D.M., 2015. What the F-measure doesn't measure: Features, Flaws, Fallacies and Fixes. arXiv preprint arXiv:1503.06410.

[Rabut et al., 2019] Rabut, B.A., Fajardo, A.C. and Medina, R.P., 2019, October. Multi-class document classification using improved word embeddings. In Proceedings of the 2nd International Conference on Computing and Big Data (pp. 42-46).

[Risch and Krestel, 2018] Risch, J. and Krestel, R., 2018, September. Learning patent speak: Investigating domain-specific word embeddings. In 2018 Thirteenth International Conference on Digital Information Management (ICDIM) (pp. 63-68). IEEE.

[Roy et al., 2019] Roy, A., Park, Y. and Pan, S., 2019, November. Incorporating domain knowledge in learning word embedding. In 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI) (pp. 1568-1573). IEEE.

[Sarkar, 2016] Sarkar, D., 2016. Text analytics with python. New York, NY, USA:: Apress.

[Sennet, 2014] Sennet, A., 2014. Polysemy. Oxford Handbooks Online.

[Sheng et al., 2014] Sheng, V.S., Gu, B., Fang, W. and Wu, J., 2014. Cost-sensitive learning for defect escalation. Knowledge-Based Systems, 66, pp.146-155.

[Son et al., 2014] Son, G., Hazlewood, V. and Peterson, G.D., 2014, July. On automating XSEDE user ticket classification. In Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment (pp. 1-7).

[Telnoni et al., 2019] Telnoni, P.A., Budiawan, R. and Qana'a, M., 2019, November. Comparison of machine learning classification method on text-based case in twitter. In 2019 International Conference on ICT for Smart Society (ICISS) (Vol. 7, pp. 1-5). IEEE.

[Wang and Zhang, 2018] Wang, J. and Zhang, M.L., 2018, July. Towards mitigating the class-imbalance problem for partial label learning. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 2427-2436).

[Wu et al., 2017] Wu, F., Huang, Y. and Yuan, Z., 2017. Domain-specific sentiment classification via fusing sentiment knowledge from multiple sources. Information Fusion, 35, pp.26-37.

[Xu et al., 2016] Xu, J., Tang, L. and Li, T., 2016. System situation ticket identification using SVMs ensemble. Expert Systems with Applications, 60, pp.130-140.

[Yao et al., 2018] Yao, Z., Sun, Y., Ding, W., Rao, N. and Xiong, H., 2018, February. Dynamic word embeddings for evolving semantic discovery. In Proceedings of the eleventh acm international conference on web search and data mining (pp. 673-681).

[Yap et al., 2014] Yap, B.W., Rani, K.A., Rahman, H.A.A., Fong, S., Khairudin, Z. and Abdullah, N.N., 2014. An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets. In Proceedings of the first international conference on advanced data and information engineering (DaEng-2013) (pp. 13-22). Springer, Singapore.

[Yen et al., 2017] Yen, A.Z., Huang, H.H. and Chen, H.H., 2017, August. Fusing domain-specific data with general data for in-domain applications. In Proceedings of the International Conference on Web Intelligence (pp. 566-572).

[Zeng et al., 2014] Zeng, C., Li, T., Shwartz, L. and Grabarnik, G.Y., 2014, May. Hierarchical multi-label classification over ticket data using contextual loss. In 2014 IEEE Network Operations and Management Symposium (NOMS) (pp. 1-8). IEEE.

[Zeng et al., 2017] Zeng, C., Zhou, W., Li, T., Shwartz, L. and Grabarnik, G.Y., 2017. Knowledge guided hierarchical multi-label classification over ticket data. IEEE Transactions on Network and Service Management, 14(2), pp.246-260.

[Zhou and Liu, 2005] Zhou, Z.H. and Liu, X.Y., 2005. Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Transactions on knowledge and data engineering, 18(1), pp.63-77.

[Zhu et al., 2017] Zhu, T., Lin, Y. and Liu, Y., 2017. Synthetic minority oversampling technique for multiclass imbalance problems. Pattern Recognition, 72, pp.327-340.

Abstract: The emergence of pre-trained language models (PLMs) has shown great success in many Natural Language Processing (NLP) tasks including text classification. Due to the minimal to no feature engineering required when using these models, PLMs are becoming the de facto choice for any NLP task. In this paper, we compare the performance of four different PLMs on three public domain-free datasets and a real-world dataset containing domain-specific words, against a simple SVM linear classifier with TFIDF vectorized text.

Chapter 3: A Comparison of SVM against Pre-trained Language Models (PLMs) for Text Classification Tasks[9]

3.1 Introduction

Text classification is the task of classifying text (e.g., tweets, news, and customer reviews) into different categories (i.e., tags). It is a challenging task especially when the text is 'technical'. We define 'technical' text in terms of the vocabulary used to describe a given document, e.g., classifying health records, human genomics, IT discussion forums, etc. These kinds of documents require special pre-processing since the basic NLP pre-processing steps may remove critical words necessary for correct classification, resulting in a performance drop in the deployed system [Brundage et al., 2021].

Recently, pre-trained language models (PLMs) such as BERT [Devlin et al., 2018] and ELMO [Neumann et al., 2018] have shown promising results in several NLP tasks, including spam filtering, sentiment analysis, and question answering. In comparison to traditional models, PLMs require less feature engineering and minimal effort in data cleaning. Thus becoming the consensus for many NLP tasks [Han et al, 2021].

With an enormous number of trainable parameters, these PLMs can encode a substantial amount of linguistic knowledge that is beneficial to contextual

[9] A version of this chapter has been published in (Wahba, Y., Madhavji, N.H. and Steinbacher, J., 2022, A Comparison of SVM against Pre-trained Language Models (PLMs) for Text Classification Tasks, 8th International Conference on Machine Learning, Optimization, and Data Science (LOD 2022), Lecture Notes in Computer Science (LNCS),Cham: Springer Nature Switzerland, pp. 304-313)

representations [Han et al, 2021]. For example, word polysemy (i.e., the coexistence of multiple meanings for a word or a phrase –e.g., 'bank' could mean 'river bank' or 'financial bank') in a domain-free text.

In contrast, in a domain-specific text that contains technical jargon, a word has a more precise meaning (i.e., monosemy) [Aronoff and Rees-Miller, 2020]. For example, the word 'run' in an IT text would generally only mean 'execute' and not 'rush'. Thus, it appears that domain-specific text classification will likely not benefit from the rich linguistic knowledge encoded in PLMs.

Despite the widespread use of PLMs in a broad range of downstream tasks, their performance is still being evaluated by researchers for their drawbacks [Acheampong et al., 2021]. For example: (i) the large gap between the pre-training objectives (e.g., predict target words) and the downstream objectives (e.g., classification) limits the ability to fully utilize the knowledge encoded in PLMs [Han et al., 2021], (ii) the high computational cost and the large set of trainable parameters make these models impractical for training from scratch, (iii) dealing with rare words is a challenge for PLMs [Schick and Schütze, 2020], and (iv) the performance of PLMs may not be generalizable [McCoy et al., 2019].

Thus, this paper evaluates the performance of different pre-trained language models (PLMs) against a linear Support Vector Machine (SVM) classifier. The motivation for this comparative study is rooted in the fact that: (i) while PLMs are being used in text classification tasks [Zhao et al., 2021; Zheng and Yang, 2019], they are more computationally expensive than the simpler SVMs, and (ii) PLMs have been used predominantly on public or domain-free datasets and it is not clear how they fare against simpler SVMs on domain-specific datasets.

The findings of our study suggest that the problem of classifying domain-specific or generic text can be addressed efficiently using old traditional classifiers such as SVM and a vectorization technique such as TFIDF that do not involve the complexity found in neural network models such as PLMs. To the best of our knowledge, no such comparative analysis has so far been described in the scientific literature.

The rest of the paper is organized as follows. Section 3.2 describes related work. Section 3.3 describes the empirical study. Section 4.4 presents the research results. Section 4.5 concludes the paper.

3.2 Related Work

In this section, we give an overview of the existing literature on the applications of PLMs and some of the drawbacks reported.

Pre-trained language models (PLMs) are deep neural networks trained on unlabeled large-scale corpora. The motivation behind these models is to capture rich linguistic knowledge that could be further transferred to target tasks with limited training samples (i.e., fine-tuning). BERT [Devlin et al., 2018], XLM [Lample and Conneau, 2019], RoBERTa [Liu et al., 2019], and XLNet [Yang et al., 2019] are examples of PLMs that have achieved significant improvements on a large number of NLP tasks (e.g., question answering, sentiment analysis, text generation).

Nevertheless, the performance of these models on domain-specific tasks was questioned [Gururangan et al., 2020] as these models are trained on general domain corpora such as Wikipedia, news websites, and books. Hence, fine-tuning or fully re-training PLMs for downstream tasks has become a consensus. Beltagi et al. [Beltagy et al., 2019] released SciBERT which is fully retrained on scientific text (i.e., papers). Lee et al. [Lee et al., 2020] released BioBERT for biological text. Similarly, Clinical BERT [Huang et al., 2019; Alsentzer et al., 2019] was released for clinical text and FinBERT [Araci, 2019] for the financial domain.

Other researchers applied PLMs by fine-tuning the final layers to the downstream task. For example, Elwany et al. [Elwany et al., 2019] report valuable improvements on legal corpora after fine-tuning. Lu [Lu, 2020] fine-tuned RoBERTa for Commonsense Reasoning and Tang et al. [Tang et al., 2020] fine-tuned BERT for multi-label sentiment analysis in code-switching text. Finally, Yuan et al. [Yuan et al., 2020] fine-tuned BERT and ERNIE [Sun et al., 2020] for the detection of Alzheimer's Disease.

However, Gururangan et al. [Gururangan et al., 2020] show that simple fine-tuning of PLMs is not always sufficient for domain-specific applications. Their work suggests that

the second phase of pre-training can provide significant gains in task performance. Similarly, Kao et al. [Kao et al., 2020] suggest that duplicating some layers in BERT prior to fine-tuning can lead to better performance on downstream tasks.

Another body of research focuses on understanding the weaknesses of PLMs by either applying them to more challenging datasets or by investigating their underlying mechanisms. For example, McCoy et al. [McCoy et al., 2019] report the failure of BERT when evaluated on the HANS dataset. Their work suggests that evaluation sets should be drawn from a different distribution than the train set. Also, Schick and Schütze [Schick and Schütze, 2020] introduce WNLaMPro (WordNet Language Model Probing) dataset to assess the ability of PLMs to understand rare words. Lastly, Kovaleva et al. [Kovaleva et al., 2019] show redundancy in the information encoded by different heads in BERT, and manually disabling attention in certain heads will lead to performance improvement.

This paper adds to the growing literature on evaluating PLMs. In particular, our investigative question is: How does a linear classifier such as SVM compare against the state-of-the-art PLMs on both general and technical domains?

3.3 Empirical Study

In this section, we describe the empirical study that we conducted. In particular, we describe the infrastructure used, the datasets, and the different PLMs used. Finally, we describe the SVM algorithm used, and the pre-processing steps done prior to applying SVM. The experimental algorithms are written in Python 3.8.3. The testing machine is Windows 10 with an Intel Core i7 CPU 2.71 GHz and 32GB of RAM.

3.3.1 Text Classification Datasets

Our experiments were evaluated on four datasets:

1. BBC News [Greene and Cunningham, 2006]: a public dataset originating from BBC News. It consists of 2,225 documents, categorized into 5 groups, namely: business, entertainment, politics, sport, and tech.

2. 20NewsGroup [20Newsgroups, 2022]: a public dataset consisting of 18,846 documents, categorized into 20 groups.

3. Consumer Complaints [Bureau of Consumer Financial Protection, 2022]: a public benchmark dataset published by the Consumer Financial Protection Bureau; it is a collection of complaints about consumer financial products and services. It consists of 570,279 documents categorized into 15 classes.

4. IT Support tickets: a private dataset obtained from a large industrial partner. It is composed of real customer issues related to a cloud-based system. It consists of 194,488 documents categorized into 12 classes.

Table 3-1 summarizes the properties of the four datasets.

Table 3-1: Dataset properties

Dataset	**# of classes**	**# of instances**	**# of features (n-gram=1)**	**# of features (n-gram=3)**
BBC News	5	2,225	26,781	811,112
20NewsGroup	20	18,846	83,667	2,011,358
Consumer Complaints	15	570,279	53,429	6,112,905
IT Support tickets	12	194,488	16,011	3,185,796

The IT Support tickets dataset will be referred to hereon as the ‘domain-specific’ dataset. This dataset suffers from a severe imbalance as seen in Figure 3-1. However, we prefer to avoid the drawbacks of sampling techniques [Zhou and Liu, 2006; He and Ma, 2013] and keep the distribution as is.

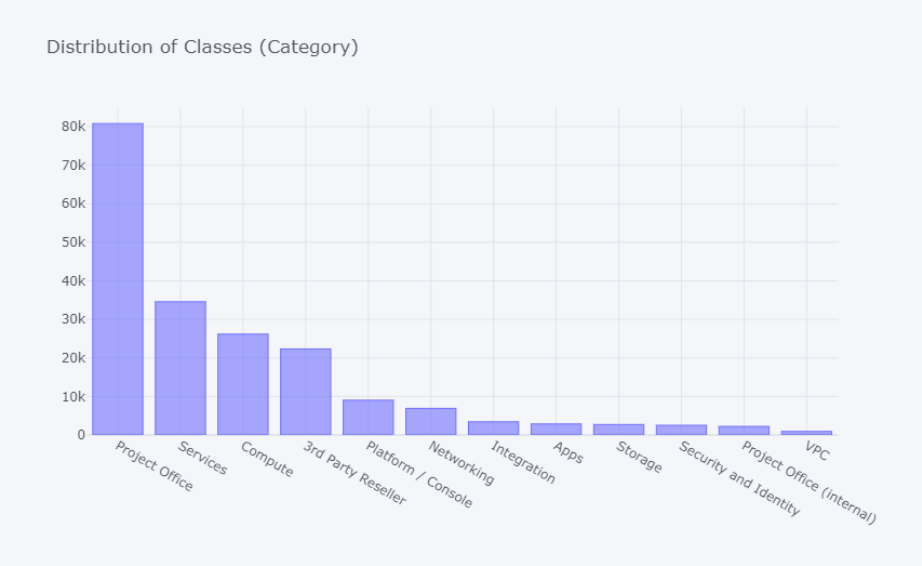


Figure 3-1: Class distribution of the domain-specific dataset showing

Another problem with this dataset is the presence of a large number of technical words (i.e., jargon) related to the Cloud terminologies (e.g., Bluemix, Kubernetes, Iaas, Vmware, etc.). These words are not found in the PLMs vocabulary and hence, they get broken down into subwords using a subword tokenization algorithm. For instance, BERT uses a WordPiece tokenizer [Wu et al., 2016] which handles non-technical words quite well. However, we notice that it fails to tokenize technical words and domain-specific abbreviations in our domain-specific dataset. For example:

"Kubernetes" ⇒ ['ku', '##ber', '##net', '##es']

"configuration" ⇒ "config" ⇒ ['con', '##fi', '##g']

3.3.2 Pre-trained Language Models (PLMs)

The following PLMs were considered for this study:

1. BERT [Devlin et al., 2018]: A widely used pre-training language model that is based on a bidirectional deep Transformer as the main structure. BERT achieved state-of-the-art results on 11 different NLP tasks including question answering and named entity recognition (NER).
2. DistilBERT [Sanh et al., 2019]: A lighter, smaller, and faster version of BERT. By reducing the size of the BERT model by 40%, while keeping 97% of its language understanding capability, it's considered 60% faster than BERT.
3. RoBERTa [Liu et al., 2019]: One of the successful variants of BERT that achieved impressive results on many NLP tasks. By changing the MASK pattern, discarding the NSP task, and using a larger batch size and longer training sentences.
4. XLM [Lample and Conneau, 2019]: Designed specifically for cross-lingual classification tasks by leveraging bilingual sentence pairs. XLM uses a known pre-processing technique (BPE) and a dual-language training mechanism.

For this study, we fine-tuned all the PLMs to the domain-specific dataset and the three generic datasets. In all our experiments, we use the following hyperparameters for

fine-tuning: maximum sequence length of 256, adam learning rate (*lr*) of 1e-5, batch size of 16, and a train-test split ratio of 80:20.

3.3.3 Support Vector Machines (SVM)

A Support Vector Machine is a popular supervised margin classifier, reported as one of the best algorithms for text classification [Joachims, 1998; Telnoni et al., 2019]. We chose the LinearSVC algorithm in the Scikit-learn library [Pedregosa et al., 2011], which implements a one-versus-all (OVA) multi-class strategy. This algorithm is suitable for high-dimensional datasets and is characterized by a low running time [Chauhan et al., 2019].

Unlike PLMs, traditional machine learning models require pre-processing data cleaning steps. In our study, we used the following pre-processing steps on the four datasets: (i) removing missing data; (ii) removing numbers and special characters; (iii) lower casing; (iv) tokenization; (v) lemmatization; and (vi) word vectorization using TFIDF[10].

It is important to note that when applying the TFIDF vectorizer, we tried different N-grams. An 'N-gram' is simply a sequence of N words that predicts the occurrence of a word based on the occurrence of its (N – 1) previous words. The default setting is Unigrams. In our study, we used trigrams which means that we included feature vectors consisting of all unigrams, bigrams, and trigrams.

3.4 Results

In this section, we discuss the results of applying four different fine-tuned PLMs (i.e., BERT, DistilBERT, RoBERTa, XLM) and a linear SVM classifier on the four datasets described in Section 3.1.

---

[10] TFIDF stands for Term Frequency-Inverse Document Frequency, which is a combination of two metrics:

1. Term frequency (*tf*): a measure of how frequently a term *t*, appears in a document *d*.

2. Inverse document frequency(idf): a measure of how important a term is. It is computed by dividing the total number of documents in our corpus by the document frequency for each term and then applying logarithmic scaling on the result.

Table 3-2 shows the F1-scores obtained when applying the four PLMs and a linear SVM classifier on the four datasets. When evaluating PLMs, we used 3 epochs because we observed that when the number of epochs exceeds 3, the training loss decreases with each epoch, and the validation loss increases. This translates to overfitting. Thus, all our experiments are run for 3 epochs only.

For the domain-specific dataset, it is clear how the linear SVM achieves a comparable performance (0.79) as any of the fine-tuned PLMs. Similarly, for the BBC dataset, SVM surprisingly achieves the same F1-score (0.98) as RoBERTa on the third epoch. However, we expected that PLMs would significantly outperform SVM on general domain datasets.

For the 20NewsGroup, SVM outperformed all PLMs with an F1-score of 0.93. This accuracy score was a result of considering the meta-data (i.e., headers, footers, and quotes) as part of the text that is fed to the classifier. However, when we ignored the meta-data, there was a performance drop of 15%.

The last dataset is the Consumer Complaints which is the largest dataset (570,279 instances) as described in Table 3-1. The accuracy of the linear SVM (0.82) was very close to the highest accuracy of 0.85 obtained by BERT and RoBERTa. While 0.82 is very competitive, we believe there is room for improvement if feature selection techniques were considered as this dataset is characterized by a large feature set.

The accuracy scores of PLMs are generally higher on generic datasets that do not contain domain-specific or rare words. Also, we notice a small gap between the accuracy scores of all PLMs in the third epoch for all datasets.

In summary, the key points are:

- Linear SVM proved to be comparable to PLMs for text classification tasks.
- PLMs accuracy scores are generally higher on generic datasets.
- The importance of feature engineering for text classification is highlighted by including meta-data.

Table 3-2: Comparison of four PLMs against SVM Linear classifier in terms of accuracy (F1-score)

Dataset	**Model**	**Epoch 1**	**Epoch 2**	**Epoch**
		Accuracy (F1-score)		

IT Support Tickets	BERT	0.78	0.79	0.79
	DistilBERT	0.77	0.78	0.79
	XLM	0.77	0.79	0.79
	RoBERTa	0.77	0.78	0.79
	LinearSVM(n-gram=3)	0.79		
BBC	BERT	0.97	0.97	0.97
	DistilBERT	0.97	0.97	0.97
	XLM	0.88	0.96	0.97
	RoBERTa	0.97	0.97	0.98
	LinearSVM(n-gram=3)	0.98		
20NewsGroup	BERT	0.85	0.91	0.92
	DistilBERT	0.82	0.90	0.90
	XLM	0.89	0.91	0.92
	RoBERTa	0.84	0.87	0.90
	LinearSVM	0.93		
Consumer Complaints	BERT	0.83	0.84	0.85
	DistilBERT	0.82	0.84	0.84
	XLM	0.80	0.82	0.83
	RoBERTa	0.83	0.84	0.85
	LinearSVM	0.82		

3.5 Conclusions

The study described in this paper compares the performance of several fine-tuned PLMs (see Section 3.3.2) against that of a linear SVM classifier (see Section 3.3.3) for the task of text classification. The datasets used in the study are: a domain-specific dataset of real-world support tickets from a large organization as well as three generic datasets (see Table 3-1).

To our surprise, we found that a pre-trained language model does not provide significant gains over the linear SVM classifier. We expected PLMs to outperform SVM on the generic datasets, however, our study indicates comparable performance for both models (see Table 3-2). Also, our study indicates that SVM outperforms PLMs on one of the generic datasets (i.e., 20NewsGroup).

Our finding goes against the trend of using PLMs on any NLP task. Thus, for text classification, we recommend prudence when deciding on the type of algorithms to use. Since our study seems to be the first comparative study of PLMs against SVM on generic datasets as well as on a domain-specific dataset, we encourage replication of this

study to create a solid body of knowledge for confident decision-making on the choice of algorithms.

References

[20Newsgroups, 2022] 20 Newsgroups Data Set Homepage, http://qwone.com/~jason/20Newsgroups/. (last accessed Oct. 15, 2022).

[Acheampong et al., 2021] Acheampong, F.A., Nunoo-Mensah, H. and Chen, W., 2021. Transformer models for text-based emotion detection: a review of BERT-based approaches. Artificial Intelligence Review, 54(8), pp.5789-5829.

[Alsentzer et al., 2019] Alsentzer, E., Murphy, J.R., Boag, W., Weng, W.H., Jin, D., Naumann, T. and McDermott, M., 2019. Publicly available clinical BERT embeddings. In: Proceedings of the 2nd Clinical Natural Language Processing Workshop, Association for Computational Linguistics, Minneapolis, Minnesota, USA, pp.72-78.

[Araci, 2019] Araci, D., 2019. FinBERT: financial sentiment analysis with pre-trained language models. arXiv preprint. arXiv:1908.10063.

[Aronoff and Rees-Miller, 2020] Aronoff, M. and Rees-Miller, J. eds., 2020. The handbook of linguistics. John Wiley & Sons.

[Beltagy et al., 2019] Beltagy, I., Lo, K. and Cohan, A., 2019. SciBERT: A pretrained language model for scientific text. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Hong Kong, pp. 3613– 3618.

[Brundage et al., 2021] Brundage, M.P., Sexton, T., Hodkiewicz, M., Dima, A. and Lukens, S., 2021. Technical language processing: Unlocking maintenance knowledge. Manufacturing Letters, 27, pp.42-46.

[Bureau of Consumer Financial Protection, 2022] Consumer Complaint Database Homepage, https://www.consumerfinance.gov/data-research/consumer-complaints. (last accessed Oct. 15, 2022).

[Chauhan et al., 2019] Chauhan, V.K., Dahiya, K. and Sharma, A., 2019. Problem formulations and solvers in linear SVM: a review. Artificial Intelligence Review, 52(2), pp.803-855.

[Devlin et al., 2018] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Minneapolis. pp.4171-4186.

[Elwany et al., 2019] Elwany, E., Moore, D. and Oberoi, G., 2019. Bert goes to law school: Quantifying the competitive advantage of access to large legal corpora in contract understanding. In: Proceedings of NeurIPS Workshop on Document Intelligence.

[Greene and Cunningham, 2006] Greene, D., and Cunningham, P., 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In: Proceedings of the 23rd international conference on Machine learning (ICML). pp. 377–384.

[Gururangan et al., 2020] Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D. and Smith, N.A., 2020. Don't stop pretraining: adapt language models to domains and tasks. In: Proceedings of ACL.

[Han et al, 2021] Han, X., Zhang, Z., Ding, N., Gu, Y., Liu, X., Huo, Y., Qiu, J., Yao, Y., Zhang, A., Zhang, L. and Han, W., 2021. Pre-trained models: Past, present and future. AI Open, 2, pp.225-250.

[Han et al., 2021] Han, X., Zhao, W., Ding, N., Liu, Z. and Sun, M., 2021. Ptr: Prompt tuning with rules for text classification. arXiv preprint arXiv:2105.11259.

[He and Ma, 2013] He, H., Ma, Y., 2013. Imbalanced Learning: Foundations, Algorithms, and Applications, 1st edn. Wiley-IEEE Press, New York.

[Huang et al., 2019] Huang, K., Altosaar, J. and Ranganath, R., 2019. ClinicalBERT: Modeling clinical notes and predicting hospital readmission. ArXiv: 1904.05342.

[Joachims, 1998] Joachims, T., 1998, April. Text categorization with Support Vector Machines: Learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds) Machine Learning: ECML-98. ECML 1998. Lecture Notes in Computer Science, vol 1398. Springer, Berlin, Heidelberg, pp.137–142.

[Kao et al., 2020] Kao, W.T., Wu, T.H., Chi, P.H., Hsieh, C.C. and Lee, H.Y., 2020. BERT's output layer recognizes all hidden layers? Some Intriguing Phenomena and a simple way to boost BERT. arXiv preprint arXiv:2001.09309.

[Kovaleva et al., 2019] Kovaleva, O., Romanov, A., Rogers, A., Rumshisky, A., 2019. Revealing the dark secrets of BERT. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China.

[Lample and Conneau, 2019] Lample, G. and Conneau, A., 2019. Cross-lingual language model pretraining. In: Proceedings of the Advances in Neural Information Processing Systems. Vancouver. pp. 7057–7067.

[Lee et al., 2020] Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C.H. and Kang, J., 2020. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. Bioinformatics, pp. 1234–1240.

[Liu et al., 2019] 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 pretraining approach. arXiv preprint arXiv:1907.11692.

[Lu, 2020] Lu, D., 2020, December. Masked Reasoner at SemEval-2020 Task 4: Fine-Tuning RoBERTa for Commonsense Reasoning. In SemEval@ COLING (pp. 411-414).

[McCoy et al., 2019] McCoy, R. T., Pavlick, E. and Linzen, T., 2019. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy.

[Neumann et al., 2018] Neumann, M.P.M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L., 2018. Deep contextualized word representations. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. New Orleans. pp.2227–2237.

[Pedregosa et al., 2011] sklearn.svm.LinearSVC. https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC (last accessed Oct. 17, 2022).

[Sanh et al., 2019] Sanh, V., Debut, L., Chaumond, J. and Wolf, T., 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In: Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing (EMC2) co-located with the Thirty-third Conference on Neural Information Processing Systems (NeurIPS 2019), pp. 1–5.

[Schick and Schütze, 2020] Schick, T. and Schütze, H., 2020, April. Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 05, pp. 8766-8774).

[Sun et al., 2020] Sun, Y., Wang, S., Li, Y., Feng, S., Tian, H., Wu, H. and Wang, H., 2020. Ernie 2.0: A continual pre-training framework for language understanding. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, No. 05, pp. 8968-8975.

[Tang et al., 2020] Tang, T., Tang, X. and Yuan, T., 2020. Fine-Tuning BERT for Multi-Label Sentiment Analysis in Unbalanced Code-Switching Text. IEEE Access, vol. 8, pp. 193248-193256.

[Telnoni et al., 2019] Telnoni, P.A., Budiawan, R. and Qana'a, M., 2019, November. Comparison of machine learning classification method on text-based case in twitter. In 2019 International Conference on ICT for Smart Society (ICISS) (Vol. 7, pp. 1-5). IEEE.

[Wu et al., 2016] Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., and Klingner, J., 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

[Yang et al., 2019] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R. and Le, Q.V., 2019. XLNet: Generalized autoregressive pretraining for language understanding. In: Proceedings of the Advances in Neural Information Processing Systems. Vancouver. pp. 5754–5764.

[Yuan et al., 2020] Yuan, J., Bian, Y., Cai, X., Huang, J., Ye, Z., Church, K., 2020. Disfluencies and fine-tuning pre-trained language models for detection of alzheimer's disease. In: INTER-SPEECH, pp. 2162–2166.

[Zhao et al., 2021] Zhao, Z., Zhang, Z. and Hopfgartner, F., 2021, April. A comparative study of using pre-trained language models for toxic comment classification. In Companion Proceedings of the Web Conference 2021 (pp. 500-507).

[Zheng and Yang, 2019] Zheng, S. and Yang, M., 2019, October. A new method of improving BERT for text classification. In International Conference on Intelligent Science and Big Data Engineering (pp. 442-452). Springer, Cham.

[Zhou and Liu, 2006] Zhou, Z.H. and Liu, X.Y., 2006. Training cost-sensitive neural networks with methods addressing the class imbalance problem, IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 1, pp. 63-77.

Addendum to Chapter 3: Attention is Not *Always* What You Need: Towards Efficient Classification of Domain-Specific Text[11]

This Chapter is an addendum to Chapter 3. The reason that led to the inclusion of this addendum is to increase the empirical power of the results reported in Chapter 3.

In **Chapter 3,** we compared the performance of four PLMs (i.e., BERT, DistilBERT, XLM, RoBERTa) against a linear SVM on four multi-class datasets (see Section 3.3.1). Our results showed that the linear SVM achieves comparable performance to the fine-tuned PLMs, and even outperformed on one of the datasets (i.e., 20NewsGroup) (see Table 3-2).

However, we had trained (i.e., fine-tuned) our PLMs for only *3 epochs,* because we experienced overfitting after the third epoch. There could be an assumption that the low number of epochs used (i.e., 3) is the reason behind the comparable performance. To eliminate that assumption, we surveyed the literature on various SOTA models using a higher number of epochs (4-15) and compared their performance against our linear SVM classifier. Results are shown in Table Add-1 where 'Add' stands for the addendum.

The following three datasets are used for the comparison:

1. 20NewsGroup [20Newsgroups, 2022]: a public dataset consisting of 18,846 documents, categorized into 20 groups. We note that some research paper uses a version of this dataset with only four major categories (comp, politics, rec, and religion), hence their results were not included in this paper.
2. BBC News [Greene and Cunningham, 2006]: a public dataset originating from BBC News. It consists of 2,225 documents, categorized into 5 groups, namely: business, entertainment, politics, sport, and tech.
3. IT Support tickets: a private dataset obtained from a large IT industrial partner. It is composed of real customer issues related to a cloud-based system. It consists of 194,488 documents categorized into 12 classes.

[11] A version of this addendum has been accepted in (Wahba, Y., Madhavji, N. and Steinbacher, J., 2023. Attention is Not Always What You Need: Towards Efficient Classification of Domain-Specific Text . In Intelligent Computing: Proceedings of the 2023 Computing Conference. Cham: Springer International Publishing, 2023.

Table Add-1: Accuracy results of SOTA models reported in the literature on two TC datasets against a Linear SVM classifier with the highest accuracies in bold.

Dataset	Model	Accuracy(%)	Reference
20NewsGroup (20 classes)	TFIDF with Naive-Bayes	81.69	[Wagh et al., 2021]
	GloVe+Average	80.43	[Wagh et al., 2021]
	GloVe+Attention	81.65	[Wagh et al., 2021
	LSTM+CNN	79.74	[Wagh et al., 2021]
	BiLSTM+Max	83.02	[Wagh et al., 2021]
	BiLSTM+Attention	81.76	[Wagh et al., 2021]
	Universal Sentence Encoder (USE)	81.76	[Wagh et al., 2021]
	ULMFiT	82.4	[Wagh et al., 2021]
	Hierarchical Attention Network (HAN)	85.01	[Wagh et al., 2021]
	BERT	85.78	[Wagh et al., 2021]
	DistilBERT	85.43	[Wagh et al., 2021]
	fastText	79.4	[Joulin et al., 2017]
	MS-CNN	86.1	[Pappagari et al., 2018]
	Text GCN	86.3	[Yao et al., 2019]
	TensorGCN	87.74	[Liu et al., 2020]
	Simplified GCN	88.50	[Wu et al., 2019]
	MLP over BERT	85.5	[Pappagari et al., 2018]
	LSTM over BERT	84.7	[Pappagari et al., 2018]
	LEAM	81.91	[Wang et al., 2018]
	CogLTX (Glove init)	87.0	[Ding et al., 2020]
	BoW + SVM	63.0	[Ding et al., 2020]
	Bi-LSTM	73.2	[Ding et al., 2020]
	RoBERTaGCN	**89.5**	[Lin et al., 2021]
	SVM+TFIDF	**90.0**	
BBC News (5 classes)	*BERT*	97	[Arslan et al., 2021]
	DistilBERT	97	[Arslan et al., 2021]
	XLM	97	[Arslan et al., 2021]
	RoBERTa	**99**	[Arslan et al., 2021]
	XLNET	98	[Arslan et al., 2021]
	TFIDF with Naive-Bayes	95.73	[Wagh et al., 2021]
	GloVe+Average	94.16	[Wagh et al., 2021]
	GloVe+Attention	95.28	[Wagh et al., 2021]
	LSTM+CNN	96.18	[Wagh et al., 2021]
	BiLSTM+Max	95.73	[Wagh et al., 2021]
	BiLSTM+Attention	96.63	[Wagh et al., 2021]
	Universal Sentence Encoder (USE)	96.63	[Wagh et al., 2021]
	ULMFiT	97.07	[Wagh et al., 2021]

	Hierarchical Attention Network (HAN)	97.75	[Wagh et al., 2021]
	BERT	**98.2**	[Wagh et al., 2021]
	DistilBERT	97.3	[Wagh et al., 2021]
	SVM+TFIDF	**98.0**	
IT Support Tickets (12 classes)	BERT	0.79	
	DistilBERT	0.78	
	XLM	0.79	
	RoBERTa	0.79	
	SVM+TFIDF	**0.79**	

Table Add-1 shows that the linear model (i.e., SVM) is comparable to several SOTA models reported in the literature on three text classification datasets.

It is to be noted that for the 20NewsGroup dataset, some authors reported accuracies higher than 90%. For instance, [Zhou et al., 2016] reported an accuracy of 96.5% using a 2D Convolutional Filter. Similarly, [Lai et al., 2015] reported an accuracy of 96.49% using recurrent convolutional neural networks. However, we note that they use only four major categories (comp, politics, rec, and religion) out of the original 20 categories for the 20NewsGroup. Hence, we strongly recommend renaming this dataset to include the number of categories (e.g., 20NewsGroup-4) to denote using only four categories and to provide a fair comparison.

To elaborate more on the results, we provide another reason behind the comparable performance of PLMs against a linear model such as SVM, and their failure to utilize their huge linguistic knowledge when employed for a domain-specific task. The reason lies behind the phenomenon of 'monosemy'. The term 'Monosemy' from the Greek roots: mono ("one") and semainein ("to signify") -- stands for words with only one meaning [Wielgosz, 2017]. It is the opposite of 'polysemy' where words could have more than one meaning [Ravin and Leacock, 2000]. For domain-specific text, the monosemic nature of words is intrinsically linked to the technical/specialized vocabulary (e.g., DNS). The reason behind this is that scientific terms need a precise meaning in order to function and be easily recognized [Wielgosz, 2017].

Table Add-2 shows a sample of pre-processed tickets (see the first column) from our support tickets dataset. The second column highlights a specialized (i.e., domain-specific) word that could have different possible meanings if appeared in a different context.

However, the actual meaning (in the third column) is the only logical/intended meaning for the word in the context of a ticketing system.

Table Add-2: The monosemic nature of some words that appear in the IT Support Tickets dataset, their actual meaning in the text, and another possible meaning.

Examples of support tickets	**Specialized word**	**Actual meaning in the text**	**Other possible meaning**
Subscription account link **cloud** …	Cloud	A system hosting software services	A visible mass of particles of condensed vapor
Cancel line item **whiskey**	Whiskey	A user-interface	A drink
Slave node serve customer traffic …	Slave	A device	A person held in forced servitude
Host freeze case brings production back …	Host	A computer	A person who talks to guests on a program
Good regard organization **space** resource field …	Space	A container	The region beyond the earth's atmosphere
Clear **cookie** success	Cookie	A file	A cake
Make **soap** connection web team …	Soap	Simple Object Access Protocol	A cleansing agent
Boot access web service …	Boot	Verb- to reload	A footwear

This study raises the question of whether PLMs are the most cost-efficient solution for domain-specific TC tasks. We encourage the replication of this study on more domain-specific datasets (e.g., law, medicine, and finance) for greater validity of the findings.

References

[20Newsgroups, 2022] 20 Newsgroups Data Set Homepage, http://qwone.com/~jason/20Newsgroups/. (last accessed Oct. 15, 2022).

[Arslan et al., 2021] Arslan, Y., Allix, K., Veiber, L., Lothritz, C., Bissyandé, T.F., Klein, J. and Goujon, A., 2021, April. A comparison of pre-trained language models for multi-

class text classification in the financial domain. In Companion Proceedings of the Web Conference 2021 (pp. 260-268).

[Ding et al., 2020] Ding, M., Zhou, C., Yang, H. and Tang, J., 2020. Cogltx: Applying bert to long texts. Advances in Neural Information Processing Systems, 33, pp.12792-12804.

[Greene and Cunningham, 2006] Greene, D., and Cunningham, P., 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In: Proceedings of the 23rd international conference on Machine learning (ICML). pp. 377–384.

[Joulin et al., 2017] Joulin, A., Grave, É., Bojanowski, P. and Mikolov, T., 2017, April. 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 (pp. 427-431).

[Lai et al., 2015] Lai, S., Xu, L., Liu, K. and Zhao, J., 2015, February. Recurrent convolutional neural networks for text classification. In Twenty-ninth AAAI conference on artificial intelligence.

[Lin et al., 2021] Lin, Y., Meng, Y., Sun, X., Han, Q., Kuang, K., Li, J. and Wu, F., 2021, August. BertGCN: Transductive Text Classification by Combining GNN and BERT. In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021 (pp. 1456-1462).

[Liu et al., 2020] Liu, X., You, X., Zhang, X., Wu, J. and Lv, P., 2020, April. Tensor graph convolutional networks for text classification. In Proceedings of the AAAI conference on artificial intelligence (Vol. 34, No. 05, pp. 8409-8416).

[Pappagari et al., 2018] Pappagari, R., Villalba, J. and Dehak, N., 2018, April. Joint verification-identification in end-to-end multi-scale CNN framework for topic identification. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 6199-6203). IEEE.

[Ravin and Leacock, 2000] Ravin, Y. and Leacock, C., 2000. Polysemy: an overview. Polysemy: Theoretical and computational approaches, pp.1-29.

[Wagh et al., 2021] Wagh, V., Khandve, S., Joshi, I., Wani, A., Kale, G. and Joshi, R., 2021, December. Comparative study of long document classification. In TENCON 2021-2021 IEEE Region 10 Conference (TENCON) (pp. 732-737). IEEE.

[Wagh et al., 2021] Wagh, V., Khandve, S., Joshi, I., Wani, A., Kale, G. and Joshi, R., 2021, December. Comparative study of long document classification. In TENCON 2021-2021 IEEE Region 10 Conference (TENCON) (pp. 732-737). IEEE.

[Wang et al., 2018] Wang, G., Li, C., Wang, W., Zhang, Y., Shen, D., Zhang, X., Henao, R. and Carin, L., 2018, July. Joint Embedding of Words and Labels for Text Classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 2321-2331).

[Wielgosz, 2017] Wielgosz, A.K., 2017. Meaning In Terms: A Monosemic Approach To The Lexical Semantics Of English And Japanese Terms Taken From Narrative Contexts. The Asian Conference on Arts & Humanities.

[Wu et al., 2019] Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T. and Weinberger, K., 2019, May. Simplifying graph convolutional networks. In International conference on machine learning (pp. 6861-6871). PMLR.

[Yao et al., 2019] Yao, L., Mao, C. and Luo, Y., 2019, July. Graph convolutional networks for text classification. In Proceedings of the AAAI conference on artificial intelligence (Vol. 33, No. 01, pp. 7370-7377).

[Zhou et al., 2016] Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H. and Xu, B., 2016, December. Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers (pp. 3485-3495).

Abstract: Correct classification of customer support tickets or complaints can help companies to improve the quality of their services to the customers. One of the challenges in text classification is when certain classes tend to share the same vocabulary. In this paper we propose a stacking algorithm based on combining different selected classifiers that operate on different feature subsets, depending on those features that tend to improve the recall and the precision of the overlapped classes.

Chapter 4: Reducing Misclassification Due to Overlapping Classes in Text Classification via Stacking Classifiers on Different Feature Subsets[12]

4.1 Introduction

Due to the rapid increase in the complexity of IT environments, support agents need to handle thousands of incoming user tickets daily. To resolve these tickets efficiently, automation of ticket classification is considered crucial for IT service management [Paramesh and Shreedhara, 2019].

Text classification is a challenging task due to the complexity of the unstructured nature of human language and the explosive growth of documents on the internet. The task is even harder when the dataset suffers from imbalance and overlapping classes [Lee and Kim, 2018]. In a multi-class text classification task, when two or more classes share the same features (i.e., words), the class boundaries are not clearly defined, and thus, the classifier's performance to distinguish between the overlapped classes decreases. This problem is known as 'overlapping classes' [Xiong et al., 2013]. Different schemes have been proposed to handle the problem of overlapping classes [Sáez et al., 2019; Sit et al., 2009]. These schemes generally fall into two categories: either modifying the classification algorithm or altering the original data by separating or

[12] A version of this chapter has been published in (Wahba, Y., Madhavji, N. and Steinbacher, J., 2022, March. Reducing Misclassification Due to Overlapping Classes in Text Classification via Stacking Classifiers on Different Feature Subsets. In Future of Information and Communication Conference (pp. 406-419). Springer, Cham).

merging the overlapped region. Despite their ability to reduce the effect of overlap, they have their drawbacks. The first technique lacks generalizability to other classification algorithms. In contrast, the second technique suffers from a loss of predictability if the overlapped region is merged or designed for data having special characteristics [Liu, 2008].

To avoid these drawbacks, we propose a new method to reduce the misclassification of overlapped classes based on "stacking" different classifiers on different feature subsets. Stacking or stacked generalizer is introduced by Wolpert [Wolpert, 1992], to combine the predictions of multiple base models. This results in reducing the bias and minimizing the error rate.

The success of stacking stems from utilizing the diversity of the base-level models' predictions. Unlike Bagging [Breiman, 1996] and Boosting [Freud, 1996], stacking combines heterogeneous classifiers (e.g., decision trees, logistic regression, and neural networks). The basic stacked model consists of two levels. Level-0 is composed of the base models (i.e., learners), and Level-1 is the final learner or so-called the meta learner. Stacking can be extended to include more levels; however, the only disadvantage would be increased computational complexity and therefore increased model training time.

In our approach, first, we train different linear and non-linear classifiers on the full feature set. Second, we use the Chi2 test to determine the best feature set for all our pre-trained classifiers that improve the f1-score for the overlapped class(es). Finally, we train a two-layered stacked model composed of the best base learners obtained from the first step as layer-1 and combine it with a strong meta-learner for the second layer.

Unlike the previous approaches, our method is generalizable as it does not depend on a specific classifier, but on the features that improve the classifier's accuracy to classify the overlapped class. Recursive searching of feature subsets is employed to identify features that tend to improve the classifier's ability to distinguish the overlapped class. By leveraging the power of using ensembles of classifiers (i.e., stacking) combined with pre-chosen features, the overall accuracy is shown to increase and the misclassification for the overlapped class(es) is shown to decrease.

The rest of the paper is organized as follows. Section 4.2 describes related work. Section 4.3 describes our methodology. Section 4.4 presents the experiments. Section 4.5 describes the results, and Section 4.6 concludes the paper.

4.2 Related Work

In this section, we give an overview of the existing literature on the domain of classification in the presence of overlapping regions and class imbalance.

Xiong et al. [Xiong et al., 2010] have proposed three different schemes for handling overlapping regions: discarding, merging, and separating: (i) The 'discarding' scheme simply ignores the overlapping region and the classifier learns only from non-overlapping regions. This can be achieved using techniques of imbalanced learning such as Tomek Links [Ivan, 1976] and SMOTE [Chawla et al., 2002]. (ii) The 'merging' scheme considers the overlapping region as an extra new class (i.e., metaclass), and the classification is done in a 2-tier approach. The top tier handles the entire dataset with the additional class that represents the overlapping region, and the lower tier handles those instances belonging to the 'overlapped region' class. (iii) In the 'separating' scheme, data belonging to the overlapped region is not modified (i.e., either ignored or merged). However, each region is treated separately by a learning model.

Trappenberg and Back [Trappenberg and Back, 2000] followed the merging approach to tackle the overlapping problem. They referred to the new class as the 'I don't know' class. The authors stated that by sacrificing the predictability of the overlapped region, they gained a drastic increase in the confidence of other classes. However, a major drawback of the merging (and discarding) schemes is the loss of prediction accuracy for data belonging to the overlapping region.

Tang and Gao [Tang and Gao, 2007] applied the reverse k-nearest neighbour algorithm (RkNN) [Korn and Muthukrishnan, 2000] to eliminate noisy patterns and the k-nearest neighbor algorithm (KNN) [Cover and Hart, 1967] is applied to extract boundary patterns. Then they utilize rough set theory to train a Support Vector Classifier (SVC) on the classes represented by lower and upper approximation sets. While this approach achieves an improved classification performance without losing predictability for

instances belonging to the overlapping regions, a major drawback is the high complexity of KNN which makes it inappropriate for high-dimensional datasets.

Following the 'separating' scheme, Liu [Liu, 2008] propose a new scheme and called it partial discriminative training (PDT) scheme that attempts to improve the separation between metaclasses, where the pattern of an overlapping class is used as a positive sample of its labeled class, and neither positive nor negative sample of the allied classes. In contrast, [Fu et al., 2015] and [Xiong et al., 2013] tackled the problem by modifying the learning algorithm. The above techniques belong to a crisp decision, where only a single label is assigned to a pattern. A different solution that uses a soft decision strategy was proposed by Tang et al. [Tang et al., 2010], this solution provides multiple decisions to the system operators which the authors believe is better than providing a wrong classification. While the previous approaches handled the problem of overlapping classes separately, Lee and Kim in [Lee and Kim, 2018] addressed both overlapping and imbalance using an overlap-sensitive margin (OSM) classifier based on a modified fuzzy support vector machine and k -nearest neighbor algorithm.

Different from the above, some researchers usually deal with poor model accuracies due to overlap in text classification tasks by switching to more complex models such as Deep Learning or Neural Networks (NN). Saeed et al. [Saeed et al., 2018] propose Deep Neural Network to classify overlapped toxic sentiments with high accuracy. Similarly, Zhang et al. [Zhang et al., 2018] and Badjatiya et al. [Badjatiya et al., 2017] utilize deep neural networks to detect hate speech in tweets. Various authors (e.g., [Agrawal and Awekar, 2018] & [Ptaszynski et al., 2017]) have also used Convolutional Neural Networks (CNNs) for cyberbullying detection, and Zhou et al. [Zhou et al., 2016] used Bidirectional LSTM to improve different text classification tasks.

We classify the above approaches for handling overlap as non-deep learning and deep-learning approaches. Drawbacks for the non-deep learning approaches are either loss of predictability for the overlapping regions or lack of generalizability to other algorithms. On the other hand, deep learning approaches are known for their high computational complexity and significant training time, they also require huge amounts of data to train.

Our research follows a different route by leveraging ensemble methods based on the stacking of different linear and tree-based machine learning models. These models do not require large volumes of data and are characterized by a fast training time. We train the stacked models on different feature subsets selected prior to model training. To the best of our knowledge, in prior research, stacking classifiers on different feature subsets has not been considered for handling the problem of class overlap.

4.3 Methodology

Section 4.3.1 gives an overview of the dataset cleaning steps used in our study and the exploratory steps performed to analyze and gain more insights into the data. We then discuss the two main techniques used in our study: Stacking in Section 4.3.2 and Feature Selection in Section 4.3.3. In Section 4.3.4, we list the various classification algorithms we use in our study.

4.3.1 Exploratory Data Analysis (EDA)

Figure 4-1 (upper) depicts the distribution of the Customer Support Tickets dataset across different categories. Likewise, Figure 4-1 (lower) depicts the distribution of the Customer Complaint dataset. As evident from Figure 4-1, both datasets are imbalanced, where the distribution of class samples is uneven by a large amount in the training dataset. This is expected to bias the classifier towards the major classes and in many cases lead to poor classification accuracies for the minor classes [Wang and Zhang, 2018]. As in our previous work [Wahba et al., 2020], we prefer not to over-sample the minority and to keep the original distributions as is. To remove noise (i.e., words that do not contribute to the learning), we removed time indicators (e.g., last week, now, etc.) and location names (i.e., countries and cities) and extended our Stop Words list to include common generic words (e.g., please, kindly, help, etc.). This greatly reduced our feature size by 12,000 features for the Customer Support Tickets dataset (hereon, D1) and by 500 features for the Consumer Complaint dataset (hereon, D2).

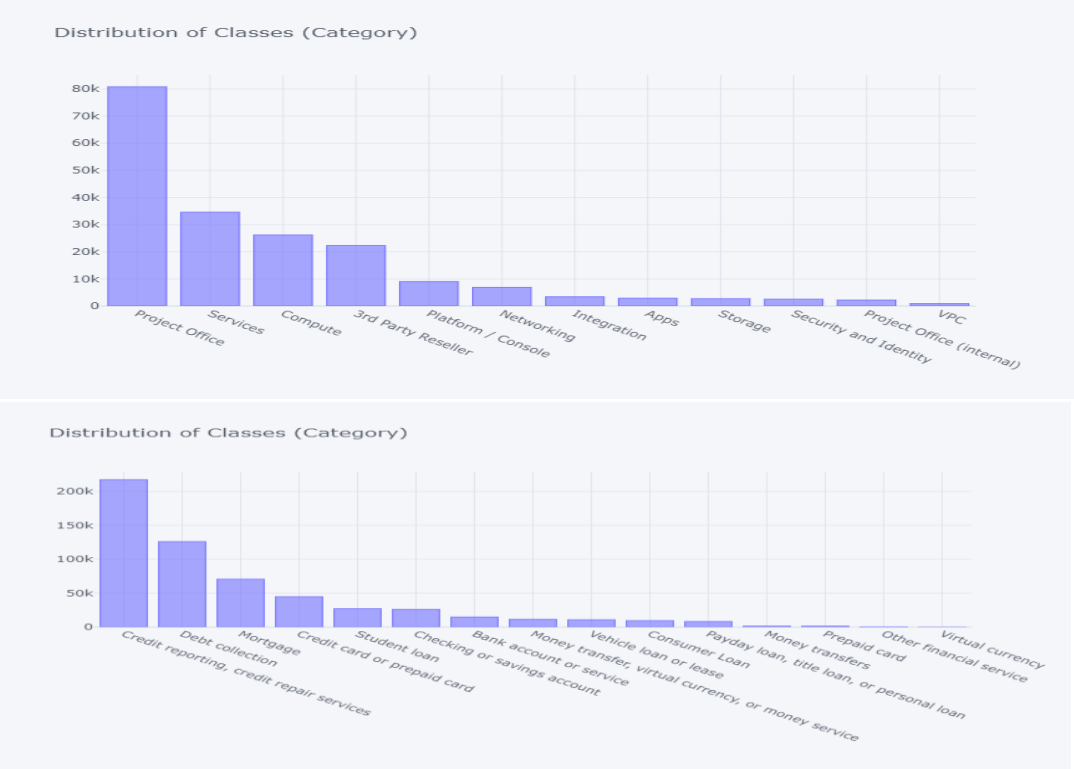


Figure 4-1: (upper) Class distribution for Customer Support Tickets dataset, (lower) Class distribution for Consumer Complaint dataset.

The next step after cleaning the dataset is experimenting with some baseline classifiers (i.e., simple basic classifiers with no hyperparameter optimization) and their accuracies will be referred to as baseline accuracies. Our baseline classifiers are described in Section 4.4. This step is important for determining: (i) whether or not feature selection is needed and (ii) whether or not adding new features would help improve the accuracy of classification. While inspecting the accuracy of our baseline classifier(s), we noticed that some classes (e.g., Apps and Bank account) with a large number of instances suffer from low accuracy (F1-scores: 0.26 and 0.4, resp., -- see Table 4-1); whereas other classes with a relatively smaller number of instances (e.g., VPC and Virtual currency) were successfully classified with higher accuracy (F1 scores: 0.56 and 0.5, resp., -- see Table 4-1). This concludes that poor accuracy is not always the result of an imbalance or a low number of instances. Other reasons might involve class overlap, outliers, and noise.

Table 4-1: Baseline accuracies (F1-scores) for two minor and major classes

Class name	Precision	Recall	F1-score	# of Instances
Apps (D1)	0.47	0.18	0.26	2,872
VPC (D1)	0.67	0.48	0.56	919
Bank account(D2)	0.40	0.39	0.40	14,885
Virtual currency(D2)	1	0.3 3	0.50	16

We excluded noise as a reason behind the misclassification as both our datasets were cleaned and pre-processed extensively before training our ML models. So, to check for features' overlap, we utilize the confusion matrix [Kulkarni et al., 2020] and the popular Venn diagrams [Baron, 1969]. Figure 4-2 presents the confusion matrix for D1, which shows that the classifier (i.e., Logistic Regression) is confusing 'Apps' with 'Platform/Console' and 'Services'. Only 106 instances are correctly classified as 'Apps', while 157 instances are mistakenly classified as 'Platform/Console' and 171 are classified as 'Services'. Similarly, Figure 4-3 shows the high confusion between 'Bank account' and 'Checking/saving' in D2, with 1266 correctly classified as 'Bank account' while 1107 mistakenly classified as 'Checking/saving'.

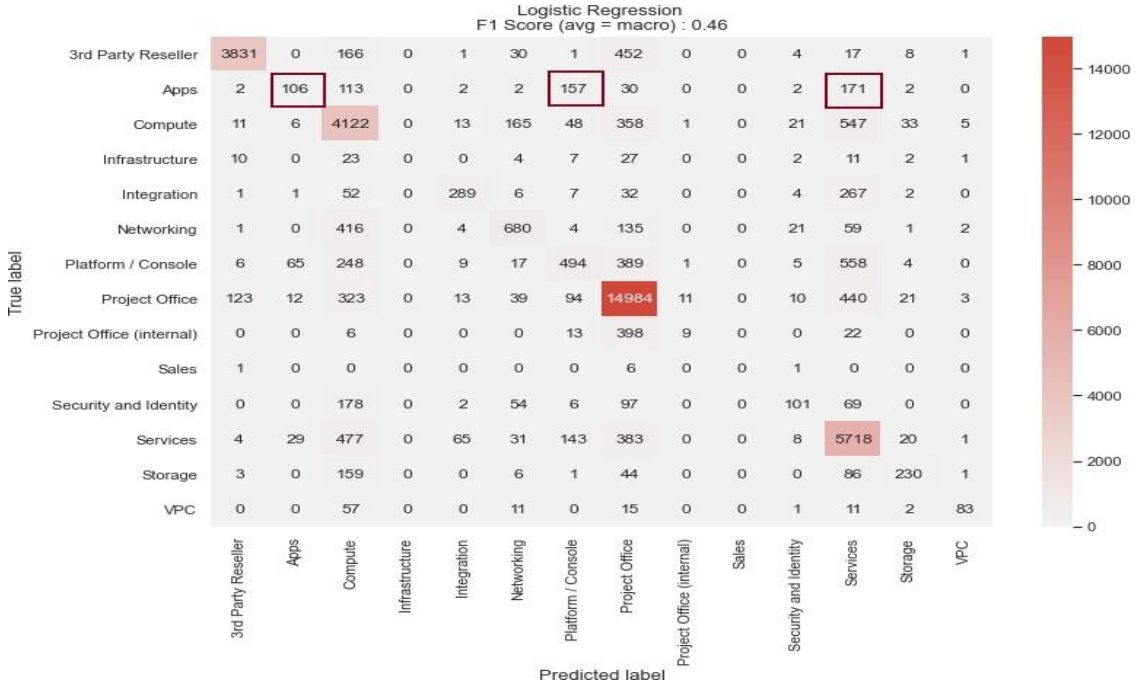


Figure 4-2: Confusion matrix for D1 showing the overlap between 'Apps', 'Platform/Console' and 'Services'

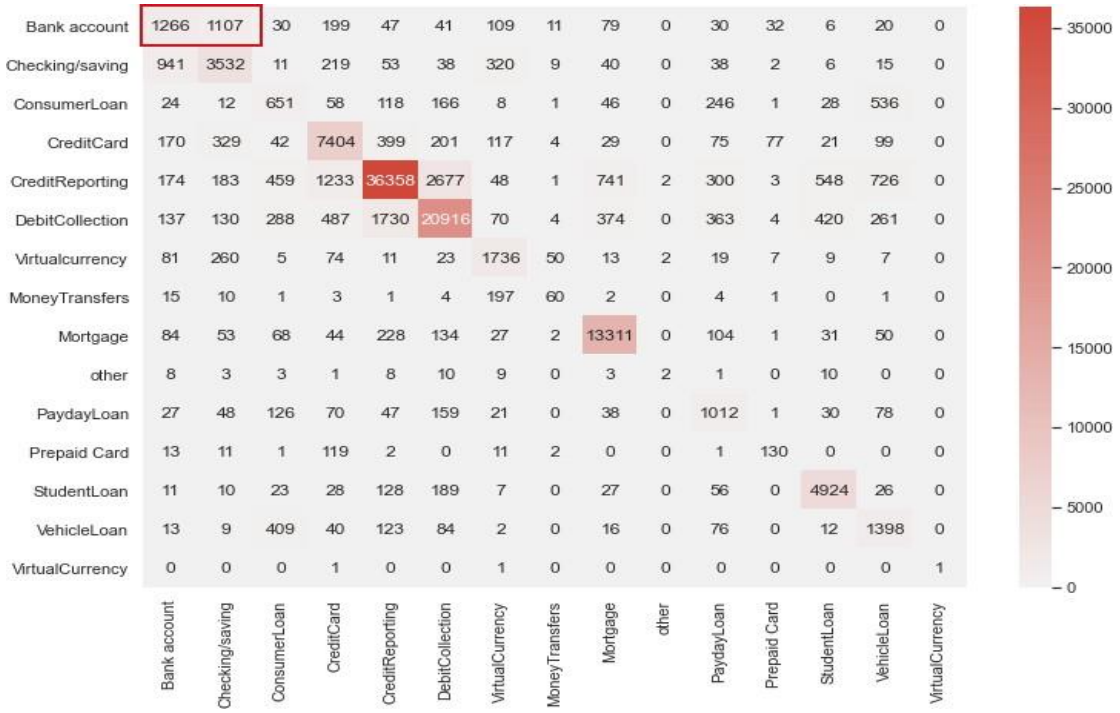


Figure 4-3: Confusion matrix for D2 showing the overlap between 'Bank account' and 'Checking/Saving'

While the confusion matrix is a great way to detect overlapping classes, we also used Ven diagrams to investigate the overlapping vocabulary. Figure 4-4 displays the top 50 words in each of the ‘Apps’ class (left circle) and ‘Platform/Console’ class (right circle), and the intersection between the two sets is shown in the middle region. It is clear from Figure 4-4 that the number of shared vocabularies between ‘Apps’ and ‘Platform/Console’ is large. The same steps were performed on the Consumer Complaint dataset.

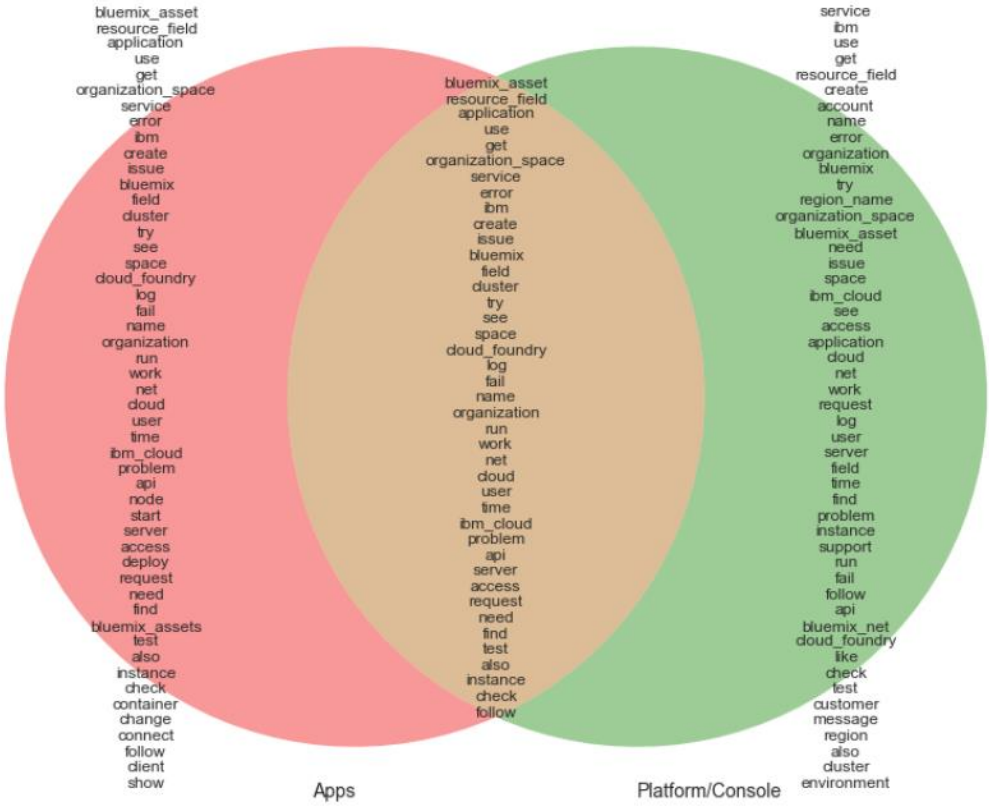


Figure 4-4: Venn diagram showing vocabulary overlap between two classes

4.3.2 Stacking

Model stacking is an efficient ensemble method that has been widely used to improve prediction accuracy [Bennett et al., 2007]. We utilize stacking to reduce the misclassification rate caused by the overlapping phenomenon. By (a) training our machine learning classifiers on feature subsets that showed an improvement to the F1-score of the overlapped classes and (b) combining the predictions of the base-level classifiers through stacking, it can result in a model with an improved classification accuracy and reduced misclassification rate for overlapped classes. Figure 4-5 shows a basic diagram for a stacked model composed of combining the predictions of m base classifiers (i.e., learners), which then provides the input for the meta-learner level.

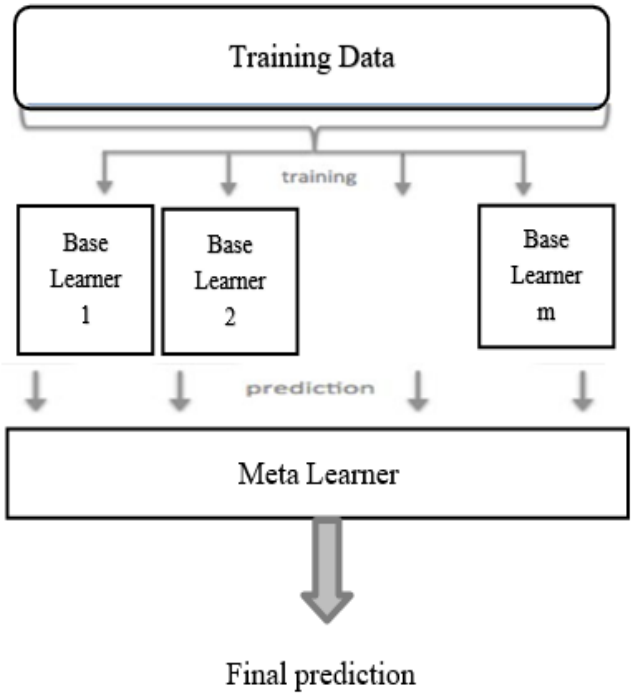


Figure 4-5: Basic stacked model

4.3.3 Feature Selection

Feature selection is the process that aims to reduce the size of the features (i.e., vocabulary) to only those features that contribute to the learning process of the classifier. This process helps in reducing text classification errors [Kou et al., 2020] and increases the model's accuracy. Feature selection methods can be classified into three main categories: filters, wrappers, and embedded methods [Chandrashekar and Sahin, 2014]. The latter two (wrapper and embedded methods) select features using a classification algorithm and a search strategy. These methods are computationally expensive and thus are not suitable for high-dimensional datasets.

However, filter methods exhibit a very low computational cost since they are classifier independent and thus, they are more commonly used for text classification tasks [Yang and Pedersen, 1997]. We use the Chi-squared test [Thomas et al., 2020] for feature selection on the two linear classifiers: SVC and LR (Section 4.3.4). While for the Extra Trees classifier, we experimented with both the Gini index and information gain

[Rokach and Maimon, 2005]. In our experiments, we tested both datasets with different numbers of features: 100, 500, 1000, 2000,5000,8000,10000,12000,500000 and 600000.

4.3.4 Classification Algorithms

The following are the classification algorithms considered for our experiments: Linear Support Vector Classifier (SVC) [Joachims, 1998], Logistic Regression (LR) [Zou et al., 2019], Extremely randomized trees (ET) [Geurts et al., 2006], Extreme Gradient Boosting (XGBoost) [Chen and Guestrin, 2016], K- nearest neighbor (KNN) [Cover and Hart, 1967].

4.4 Experiments

This section gives an overview of the datasets used and their properties. This is followed by the experimental steps taken in this study.

4.4.1 Text Classification Datasets

Our experiments were evaluated on two datasets. The first one (D1) is a real-world dataset for a global IT industrial partner; it is a collection of customer support tickets for a cloud-based system. The second dataset (D2) is a benchmark dataset published by the Consumer Financial Protection Bureau; it is a collection of complaints about consumer financial products and services [Bureau of Consumer Financial Protection, 2022]. The properties of both datasets are described in Table 4-2.

Table 4-2: Dataset properties

Dataset	**# of classes**	**# of instances**	**# of features (n-gram= 1)**	**# of features (n-gram= 3)**
Customer Support Tickets (D1)	13	194,488	15,886	2,496,703
Consumer Complaint (D2)	15	570,279	53,444	6,112,905

As can be seen from the table, both datasets are characterized by a huge feature set which when used in conjunction with trigrams (i.e., n=3) would greatly exceed the number of instances in the dataset.

4.4.2 Empirical Procedure

The following are the experimental steps performed for this study:

Step 1: Dataset pre-processing and exploratory analysis (see Section 4.3.1).

Step 2: Text vectorization. This transforms the natural text into vectors (i.e., numbers). We use Term Frequency Inverse Document Frequency[13] (i.e., TFIDF) technique for its simplicity and efficacy.

It is important to note that we experiment with different n-grams to check whether they improve the performance of the model. For dataset D1, trigrams, delivered the highest performance on all our baseline classifiers, with a low training time given our choice of learning algorithms. For dataset D2, as seen in Table 4-2, the feature dimensions for trigrams exceeded 6 million features. So, we kept the feature set size as the default unigrams (i.e., n=1) for the sake of computational complexity.

Step 3: Training on a baseline model(s) described in Section 4.3.4 and recording accuracy. Baseline models are basic ML models (e.g., Logistic regression, SVC, etc.) with the default parameters.

Step 4: Evaluation of the baseline model(s). If accuracy was satisfactory then hyperparameter optimization (i.e., parameter tuning) would be the last step before deploying the model.

Step 5: If baseline model(s) accuracy is poor, then reinspect the dataset (step 1) and add new features or identify new relevant features in conjunction with domain experts and business stakeholders.

---

[13] TFIDF stands for Term Frequency-Inverse Document Frequency, which is a combination of two metrics: Term frequency (*tf*): a measure of how frequently a term, t, appears in a document, d. And Inverse document frequency(*idf*): a measure of how important a term is. It is computed by dividing the total number of documents in our corpus by the document frequency for each term and then applying logarithmic scaling on the result.

Step 6: If adding new features is not possible or did not improve accuracy, then try different feature selection techniques described in Section 4.3.3.

Step 7: Model Stacking or ensemble learning is the final step for improving the model's accuracy. Try a different combination of machine learning models with different meta-learners.

4.5 Results

Table 4-3 shows the overall F1-score[14]for the six baseline classification algorithms (see Section 4.3.4) on the two text classification datasets. Due to the fundamental recall precision trade-off [Gordon and Kochen, 1989] (where improving the precision always results in lowering the recall and vice versa), we chose the F1-score as our performance metric. The highest overall accuracy for D1 is 79% which is achieved by SVC and XGB; whereas, for D2, 82% is the highest accuracy achieved by XGB and LR.

Table 4-3: Baseline F1-scores for our classification algorithms

Dataset	SVC	XGB	LR	KNN	ET	NB
D1	0.79	0.79	0.78	0.70	0.77	0.68
D2	0.80	0.82	0.82	0.42	0.81	0.75

In our experiments, we split the dataset into 80% for training samples and 20% for testing (i.e., evaluation). Since both datasets suffer from imbalance, we use the stratified splitting approach that preserves the same proportions of examples in each class [Sechidis et al., 2011]. To avoid overfitting [Dietterich, 1995], the final estimator (i.e., meta-learner) is trained on K-fold cross-validation where K=5.

The experimental results obtained after experimenting with different ensembles of stacked classifiers are presented in Table 4-4 and Table 4-5. We use the following abbreviations for simplicity:

SVC1: Support Vector classifier trained on 500 features selected by using the Chi2 test.

[14] F1-score = 2x Precision x Recall / Precision+ Recall

SVC2: Support Vector classifier trained on 10,000 features selected by using the Chi2 test.

ET1: Extra Trees classifier trained on features selected to split on by using the Gini index.

ET2: Extra Trees classifier trained on features selected to split on by using Information gain.

LR1: Logistic Regression classifier trained on 10,000 features.

Table 4-4: Results of different stacked models on the overall accuracy and the overlapped class on the Customer Support Tickets dataset (D1)

Experiment No.	Base Models	Meta-Learner	F1-score	Macro-Avg	Precision (Apps)	Recall (Apps)	F1-Score (Apps)
1	4 base-learners: SVC1, SVC, LR, ET1	LR	0.794	0.56	0.43	0.36	0.39
2	5 base-learners: SVC1, SVC, LR, ET1, ET2	LR	0.80	0.56	0.45	0.35	0.40
3	5 base-learners: SVC1, SVC, LR, XGB, ET1	LR	0.80	0.55	0.50	0.30	0.37
4	4 base-learners: SVC1, SVC, ET1, ET2	LR	0.794	0.56	0.43	0.36	0.39
5	5 base-learners: SVC1, SVC, KNN, ET1, ET2	LR	0.80	0.56	0.45	0.36	0.40
6	3 base-learners: SVC1, SVC, KNN	LR	0.792	0.55	0.42	0.34	0.38
7	3 base learners: SVC, LR, XGB	LR	0.77	0.55	0.36	0.45	0.40
8	3 base-learners: SVC, LR, KNN	LR	0.792	0.55	0.42	0.36	0.39

The choice of the number of features used for training the classifiers is based on several experiments conducted prior to stacking. For each feature subset selected by the Chi-squared test, the accuracy (F1-score) was recorded for the two overlapped classes: “Apps” and “Bank account”. We notice that the SVC classifier when trained on 500 features (out of 15,886) delivered the highest F1- score for the ‘Apps’ class. While for dataset D2, the highest F1- score is obtained by training both the SVC and LR on 10,000 features (out of 53,444).

Tables 4-4 and 4-5 demonstrate that our approach of stacking classifiers based on different feature subsets achieves superior overall performance compared to stacking on all features. This is clear in experiments number 2,3 and 5 in Table 4-4, where the F1-score (0.80) is higher than the highest baseline accuracy (0.79) as described in Table 4-3. Also, for the overlapped class ‘Apps’, it is clear that all experiments (1-8) achieved a higher F1-score than the baseline accuracy for ‘Apps’ (0.26) as described in Table 4-1.

Table 4-5: Results of different stacked models on the overall accuracy and the overlapped class on the Consumer Complaint dataset (D2)

Experiment No.	Base Models	Meta-Learner	F1-score	Macro-Avg	Precision (Bank account)	Recall (Bank account)	F1-Score (Bank account)
1	3 base-learners: SVC, LR, XGB	SVC	0.81	0.60	0.48	0.42	0.45
2	3 base-learners: SVC, LR, XGB	LR	0.82	0.54	0.52	0.32	0.40
3	3 base-learners: SVC2, LR, XGB	LR	0.82	0.55	0.53	0.32	0.40
4	4 base-learners: SVC2, LR1, SVC, XGB	LR	0.83	0.56	0.54	0.34	0.42
5	4 base-learners: SVC2, SVC, LR, XGB	SVC	0.82	0.61	0.49	0.41	0.45
6	5 base-learners: SVC2, LR1, SVC, ET1,ET2	SVC	0.83	0.56	0.54	0.41	0.46

7	3 base-learners: SVC2, SVC, LR1	LR	0.82	0.57	0.52	0.35	0.42

Similarly, for dataset D2, experiment number 4 and 6 in Table 4-5 achieved a higher F1-score (0.83) than the highest baseline accuracy (0.82) in Table 4-3. Also, for the overlapped class 'Bank account', experiments 4,5,6, and 7 achieved a higher F1-score than the baseline accuracy for 'Bank account' (0.40) as described in Table 4-1. We excluded KNN from our stacking experiments on D2 due to the slow running time as well as the low baseline accuracy (0.42) for KNN on dataset D2 as shown in Table 4-3. Similarly, NB (0.68) was excluded from our stacking experiments on D1.

4.6 Conclusion and Future Work

Correct classification of customers' support tickets is crucial to organizations. However, one of the challenges that face text classification is the presence of common words between different classes, known as class overlap.

This paper proposes a new method for reducing the misclassification caused by the class overlapping problem in multi-class text classification tasks. The proposed solution is based on stacking different ML models that are trained on different feature subsets described in Tables 4-4 and 4-5. The feature selection step is done prior to stacking to determine the best feature set for the given overlapped class(es).

Our experimental results on two multi-class text classification datasets show that our method achieves an improvement in the overall accuracy for our classifiers as well as an improvement for the misclassification rate given by a high F-score for our two chosen classes (i.e., 'Apps' and 'Bank account').

In the future, we plan to investigate the problem of class overlapping in the presence of imbalance for large-scale hierarchical datasets. Also, we plan to include more feature selection approaches in our experiments.

References

[Agrawal and Awekar, 2018] Agrawal, S. and Awekar, A., 2018, March. Deep learning for detecting cyberbullying across multiple social media platforms. In European conference on information retrieval (pp. 141-153). Springer, Cham.

[Badjatiya et al., 2017] Badjatiya, P., Gupta, S., Gupta, M. and Varma, V., 2017, April. Deep learning for hate speech detection in tweets. In Proceedings of the 26th international conference on World Wide Web companion (pp. 759-760).

[Baron, 1969] Baron, M.E., 1969. A note on the historical development of logic diagrams: Leibniz, Euler and Venn. The mathematical gazette, 53(384), pp.113-125.

[Bennett et al., 2007] Bennett, J., Elkan, C., Liu, B., Smyth, P. and Tikk, D., 2007. Kdd cup and workshop 2007. ACM SIGKDD explorations newsletter, 9(2), pp.51-52.

[Breiman, 1996] Breiman, L., 1996. Bagging predictors. Machine learning, 24(2), pp.123-140.

[Bureau of Consumer Financial Protection, 2022] Consumer Complaint Database Homepage, https://www.consumerfinance.gov/data-research/consumer-complaints. (last accessed Oct. 15, 2022).

[Chandrashekar and Sahin, 2014] Chandrashekar, G. and Sahin, F., 2014. A survey on feature selection methods. Computers & Electrical Engineering, 40(1), pp.16-28.

[Chawla et al., 2002] Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P., 2002. SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, pp.321-357.

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C., 2016, August. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794). (Association for Computing Machinery).

[Cover and Hart, 1967] Cover, T. and Hart, P., 1967. Nearest neighbor pattern classification. IEEE transactions on information theory, 13(1), pp.21-27.

[Dietterich, 1995] Dietterich, T., 1995. Overfitting and undercomputing in machine learning. ACM computing surveys (CSUR), 27(3), pp.326-327.

[Freud, 1996] Freud, Y., 1996. Experiments with a new boosting algorithm. In Proc. Thirteenth International Conference on Machine Learning (pp. 148-156).

[Fu et al., 2015] Fu, M., Tian, Y. and Wu, F., 2015. Step-wise support vector machines for classification of overlapping samples. Neurocomputing, 155, pp.159-166.

[Geurts et al., 2006] Geurts, P., Ernst, D. and Wehenkel, L., 2006. Extremely randomized trees. Machine learning, 63(1), pp.3-42.

[Gordon and Kochen, 1989] Gordon, M. and Kochen, M., 1989. Recall-precision trade-off: A derivation. Journal of the American Society for Information Science, 40(3), pp.145-151.

[Ivan, 1976] Ivan, T., 1976. Two modifications of CNN. IEEE transactions on Systems, Man and Communications, SMC, 6, pp.769-772.

[Joachims, 1998] Joachims, T., 1998, April. Text categorization with support vector machines: Learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds) Machine Learning: ECML-98. ECML 1998. Lecture Notes in Computer Science, vol 1398. Springer, Berlin, Heidelberg, pp.137–142.

[Korn and Muthukrishnan, 2000] Korn, F. and Muthukrishnan, S., 2000. Influence sets based on reverse nearest neighbor queries. ACM Sigmod Record, 29(2), pp.201-212.

[Kou et al., 2020] Kou, G., Yang, P., Peng, Y., Xiao, F., Chen, Y. and Alsaadi, F.E., 2020. Evaluation of feature selection methods for text classification with small datasets using multiple criteria decision-making methods. Applied Soft Computing, 86, p.105836.

[Kulkarni et al., 2020] Kulkarni, A., Chong, D. and Batarseh, F.A., 2020. Foundations of data imbalance and solutions for a data democracy. In data democracy (pp. 83-106). Academic Press.

[Lee and Kim, 2018] Lee, H.K. and Kim, S.B., 2018. An overlap-sensitive margin classifier for imbalanced and overlapping data. Expert Systems with Applications, 98, pp.72-83.

[Liu, 2008] Liu, C.L., 2008. Partial discriminative training for classification of overlapping classes in document analysis. International Journal of Document Analysis and Recognition (IJDAR), 11(2), pp.53-65.

[Paramesh and Shreedhara, 2019] Paramesh, S.P. and Shreedhara, K.S., 2019. Automated IT service desk systems using machine learning techniques. In Data Analytics and Learning (pp. 331-346). Springer, Singapore.

[Ptaszynski et al., 2017] Ptaszynski, M., Eronen, J.K.K. and Masui, F., 2017, August. Learning Deep on Cyberbullying is Always Better Than Brute Force. In LaCATODA@ IJCAI (pp. 3-10).

[Rokach and Maimon, 2005] Rokach, L. and Maimon, O., 2005. Decision trees. In Data mining and knowledge discovery handbook (pp. 165-192). Springer, Boston, MA.

[Saeed et al., 2018] Saeed, H.H., Shahzad, K. and Kamiran, F., 2018, November. Overlapping toxic sentiment classification using deep neural architectures. In 2018 IEEE international conference on data mining workshops (ICDMW) (pp. 1361-1366). IEEE.

[Sáez et al., 2019] Sáez, J.A., Galar, M. and Krawczyk, B., 2019. Addressing the overlapping data problem in classification using the one-vs-one decomposition strategy. IEEE Access, 7, pp.83396-83411.

[Sechidis et al., 2011] Sechidis, K., Tsoumakas, G. and Vlahavas, I., 2011, September. On the stratification of multi-label data. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 145-158). Springer, Berlin, Heidelberg.

[Sit et al., 2009] Sit, W.Y., Mak, L.O. and Ng, G.W., 2009. Managing category proliferation in fuzzy ARTMAP caused by overlapping classes. IEEE transactions on neural networks, 20(8), pp.1244-1253.

[Tang and Gao, 2007] Tang, Y. and Gao, J., 2007. Improved classification for problem involving overlapping patterns. IEICE TRANSACTIONS on Information and Systems, 90(11), pp.1787-1795.

[Tang et al., 2010] Tang, W., Mao, K.Z., Mak, L.O. and Ng, G.W., 2010, July. Classification for overlapping classes using optimized overlapping region detection and soft decision. In 2010 13th International Conference on Information Fusion (pp. 1-8). IEEE.

[Thomas et al., 2020] Thomas, T., P Vijayaraghavan, A. and Emmanuel, S., 2020. Applications of decision trees. In Machine learning approaches in cyber security analytics (pp. 157-184). Springer, Singapore.

[Trappenberg and Back, 2000] Trappenberg, T.P. and Back, A.D., 2000, July. A classification scheme for applications with ambiguous data. In Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium (Vol. 6, pp. 296-301). IEEE.

[Wahba et al., 2020] Wahba, Y., Madhavji, N.H. and Steinbacher, J., 2020, November. Evaluating the effectiveness of static word embeddings on the classification of IT support tickets. In Proceedings of the 30th Annual International Conference on Computer Science and Software Engineering (CASCON), (pp. 198-206).

[Wang and Zhang, 2018] Wang, J. and Zhang, M.L., 2018, July. Towards mitigating the class-imbalance problem for partial label learning. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 2427-2436).

[Wolpert, 1992] Wolpert, D.H., 1992. Stacked generalization. Neural networks, 5(2), pp.241-259.

[Xiong et al., 2010] Xiong, H., Wu, J. and Liu, L., 2010, December. Classification with classoverlapping: A systematic study. In 1st International Conference on E-Business Intelligence (ICEBI 2010) (pp. 303-309). Atlantis Press.

[Xiong et al., 2013] Xiong, H., Li, M., Jiang, T. and Zhao, S., 2013. Classification algorithm based on NB for class overlapping problem. Appl. Math, 7(2L), pp.409-415.

[Yang and Pedersen, 1997] Yang, Y. and Pedersen, J.O., 1997, July. A comparative study on feature selection in text categorization. In Proceedings of the Fourteenth International Conference on Machine Learning (ICML) (Vol. 97, No. 412-420, p. 35). (Morgan Kaufmann Publishers Inc.).

[Zhang et al., 2018] Zhang, Z., Robinson, D. and Tepper, J., 2018, June. Detecting hate speech on twitter using a convolution-gru based deep neural network. In European semantic web conference (pp. 745-760). Springer, Cham.

[Zhou et al., 2016] Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H. and Xu, B., 2016, December. Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers (pp. 3485-3495).

[Zou et al., 2019] Zou, X., Hu, Y., Tian, Z. and Shen, K., 2019, October. Logistic regression model optimization and case analysis. In 2019 IEEE 7th international conference on computer science and network technology (ICCSNT) (pp. 135-139). IEEE. (Institute of Electrical and Electronics Engineers Inc.)

Abstract: Classifying customer support tickets according to the desired criteria is an important task in IT service management. One of the biggest challenges is the presence of a large number of shared words between different classes. This problem is widely known as overlapping classes. Misclassification due to overlapping regions is a critical problem that is not well addressed in the NLP field. In this paper, we detect overlapping classes from an ML algorithm perspective and propose a hybrid machine learning model based on a linear SVM classifier and a set of N hand-crafted rules to classify the incoming ticket with high accuracy where N is the number of overlapped classes.

Chapter 5: A Hybrid Machine Learning Model for Efficient Classification of IT Support Tickets in The Presence of Class Overlap[15]

5.1 Introduction

In today's world, support ticketing systems are employed by a wide range of businesses. The ticketing system facilitates the interaction between customers and the support teams when the customer faces an issue with a product or a service. For large-scale IT companies with a large number of clients and a great volume of communications, the task of automating the classification of incoming tickets is key to guaranteeing long-term clients and ensuring business growth.

According to a survey by Zendesk [Zendesk, 2022], quick resolution time was rated as a top factor for a good customer experience. The fastest way to resolve a ticket is by accurately classifying the incoming tickets which would then be routed to the right support team, avoiding any significant delays in resolution.

[15] A version of this chapter has been published in (Wahba, Y., Madhavji, N.H. and Steinbacher, J., 2022, A Hybrid Machine Learning Model for Efficient Classification of IT Support Tickets in The Presence of Class Overlap, In Proceedings of the 32nd Annual International Conference on Computer Science and Software Engineering (CASCON22), (pp. 151-156).

For large-scale IT firms, classification involves hundreds or thousands of ticket categories which poses a challenge for accurate classification. As the number of classes increases, the possibility of overlapping between the classes also increases. Overlapping classes is a critical problem where an incoming ticket appears as a valid classification for more than one class.

The class overlap problem has been widely studied by researchers in myriad subjects (e.g., smart cities, image segmentation, document analysis, pattern recognition, etc..) either in isolation [Xiong et al., 2013] or in conjunction with class imbalance [Das et al., 2014; Lee and Kim, 2018]. However, few studies have investigated the aforementioned problem in the NLP domain and specifically for large-scale text classification tasks [Liu et al., 2019; Wahba et al., 2022].

There are two approaches for classifying instances belonging to an overlapping region. The first approach is when the system generates a single label (i.e., target) for a given instance which is known as a 'crisp decision' strategy. The second approach is known as a 'soft decision' strategy where the system generates multiple labels for a given instance that are further analyzed and judged by the support teams. Our work follows a crisp decision strategy.

In this paper, we propose a hybrid solution based on a linear SVM classifier and a set of simple rules for text classification scenarios involving class overlap. The rules are for classifying only those classes with low accuracy due to overlap. Hence, our rules are easy to update. Formulation of the rules is based on expert knowledge of the support agents and a Python library (i.e., eli5) to extract the most unique and important words for each of the overlapped classes. We investigated overlap from the classification algorithms' (i.e., SVM) perspective since SVM has proven to be a robust choice for text classification tasks with high dimensional feature space [Joachims, 1998]. Thus, we analyze its confusion matrix to determine severe overlaps, then a set of handcrafted simple rules is created based on the presence of unique keywords.

The rest of the paper is organized as follows: Section 5.2 describes related work. Section 5.3 describes the empirical study. Section 5.4 presents the experiments and research results. Section 5.5 concludes the paper.

5.2 Related Work

In this section, we first give an overview of the literature on the problem of overlapping classes, and then we present a sample of studies for handling overlapping in text classification tasks.

The early work by [Xiong et al., 2010] examined the problem of overlapping classes on five real-world binary datasets. They propose three different modelling schemes for handling overlapping regions, namely: discarding, merging, and separating. The discarding scheme ignores the overlapping region and learns only from non-overlapping regions. While this scheme might perform well on datasets with minimal overlapping, it is not suitable when the overlapping ratio is high because the discarding scheme will result in losing important information and hence a poor classification accuracy for the overlapped classes.

The merging scheme considers the overlapping region as a new class labelled as 'overlapping', and the classification is done in a 2-tier approach. The top tier learns the entire data with the additional class that represents the overlapping region, and the lower tier learns those instances belonging to the 'overlapping class. Authors in [Trappenberg and Back, 2000] followed the merging approach on two UCI Machine Learning Repository datasets and referred to the overlapping region as 'I don't know' class. The authors indicated that by sacrificing the predictability of the overlapped region, they gained a significant increase in the confidence of other classes.

In the separating scheme, two models are used to learn about both the overlapping and non-overlapping regions separately. Authors in [Tang and Gao, 2007] followed the separating scheme on five benchmark data sets from UCI Machine Learning Repository. They applied the k-nearest neighbour algorithm (KNN) [Cover and Hart, 1967] to extract boundary patterns and rough set theory to train a Support Vector Classifier (SVC) on the classes represented by lower and upper approximation sets. While this approach does not sacrifice the predictability of the overlapped region, the high complexity of KNN renders it not suitable for high-dimensional text datasets. Similarly, the work in [Xiong et al., 2013] reports improvements following the separating approach using Naïve Bayes classifier on five real-world binary data sets from UCI. Similarly,

authors in [Fu et al., 2015] propose a Two-Step Classification SVM (TSC-SVM) and applied wavelet transform to denoise adjacent samples on the abalone dataset.

None of the aforementioned works, which handle the problem of overlapping classes, were evaluated on datasets involving natural language text. Only a few works addressed the overlapping problem in the NLP domain. For example, the work in [Liu et al., 2019] proposes a fuzzy approach for hate speech text classification with overlapping instances. Their work shows that fuzzy approaches are superior in dealing with the fuzziness and ambiguity of the text. Likewise, authors in [Wahba et al., 2022a] propose the stacking of different machine learning models based on different feature subsets and show accuracy improvements on two domain-specific text datasets.

However, some researchers favor more complex models such as Neural Networks or Deep Learning in an effort to handle the problem of overlapping classes. For instance, authors in [Saeed et al., 2018] recommend the use of Deep Neural Network (DNN) models on unprocessed datasets for overlapping multi-label text classification problems. The study was evaluated on a real-world dataset of toxic comments and the text was vectorized using the pre-trained word embedding FastText by Facebook. Similarly, the work by [Georgakopoulos et al., 2018] shows promising results using Convolutional Neural Networks (CNN) based models for the task of toxic comment classification. However, their study does not investigate the presence of overlap.

Therefore, our study contributes to the meager literature on overlapping classes in the NLP domain. Specifically, the classification of domain-specific text with multiple categories that overlap. We propose a hybrid ML model based on a linear classifier and a set of simple handcrafted rules for overlapped categories with minimal human intervention.

5.3 Empirical Study

Below, we describe the infrastructure and the datasets we used in the empirical study. We also outline the overlapped classes for each dataset considered for this study and the formulation of the rules. The experimental algorithms are written in Python 3.8.3. The testing machine is Windows 10 with an Intel Core i7 CPU 2.71 GHz and 32GB of RAM.

5.3.1 Text Classification Datasets

We notice that most publicly available large-scale text datasets with a hierarchical nature contain minor to no overlap; thus, they were not suitable for this study. Hence, we selected the following datasets carefully based on the presence of a clear overlap between two or more classes (as described in Section 5.3.2):

1. IT Support Tickets: a private dataset obtained from a large industrial IT partner with real customer issues concerning a cloud-based system. The dataset consists of 194,488 documents categorized into 12 classes on the first level of the hierarchy and 110 classes on the second level.
2. MIND [Wu et al., 2020]: a large-scale hierarchical dataset for news recommendation. It was collected from anonymized behaviour logs of the Microsoft News website. It consists of 101,527 documents categorized into 15 classes.
3. Endava Anonymized Support Tickets (EAST) [Preda G., 2020]: an anonymized hierarchical dataset imported from Endavas' helpdesk system for customer support tickets. It consists of 48, 549 documents categorized into 13 classes.
4. Consumer Complaints [Bureau of Consumer Financial Protection, 2022]: a large-scale hierarchical dataset published by the Consumer Financial Protection Bureau; it is a collection of complaints about consumer financial products and services. It consists of 570,279 documents categorized into 15 classes.

5.3.2 Overlapped Classes

A simple and straightforward way to determine overlap between classes is the confusion matrix [Kulkarni et al., 2020] which shows the correct and wrong predictions in terms of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). This way detects overlap in the space inferred by the classifier. Figure 5-1 shows the confusion matrix of a linear SVM classifier applied to each of the four datasets described in Section 5.3.1.

For the IT Support Tickets, we notice that the SVM classifier is confusing 'Apps' with 'Platform/Console' and 'Service. Only 272 instances are correctly classified as

‘Apps’, while 227 are mistakenly classified as ‘Platform/Console’ and ‘181’ are mistakenly classified as ‘Services’. For the MIND dataset, the classifier is confusing ‘news’ with ‘video’ with 490 instances correctly classified as video, while mistakenly classifying 597 video instances as ‘news’. Similarly, for the EAST dataset, class ‘4’ is greatly confused with class ‘6’ with ‘426’ instances mistakenly classified as class ‘4’ while in fact, they belong to class ‘6’. The final confusion matrix is for the Consumer Complaints dataset, which shows a large confusion between ‘Bank account’ and ‘Checking/saving’. For the sake of space, we did not consider all overlapped classes for the four datasets.

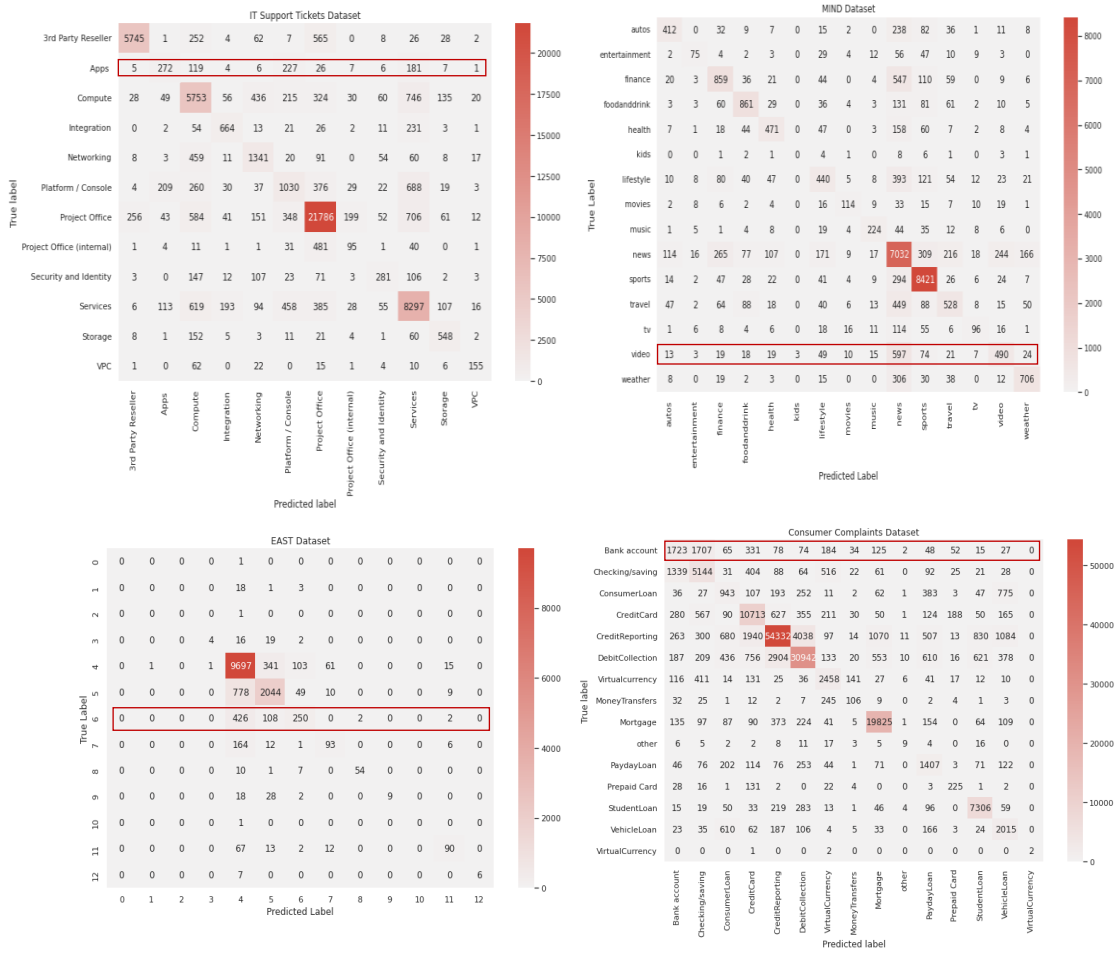


Figure 5-1: Confusion matrix for linear SVM showing overlap between two or more classes for IT Support Tickets, MIND, EAST and Consumer Complaints datasets

It is to be noted that the degree of overlap between two classes Ci and Cj is not always symmetric, meaning that the classifier could be confusing class Ci with class Cj but not necessarily confusing class Cj with class Ci. For instance, in our private IT Support Tickets dataset, the classifier is greatly confusing 'Apps' with 'Platform/Console' with 227 correctly classified and 272 misclassified as 'Platform/Console'. However, for the 'Platform/Console', the confusion is more clear with 'Services' (688 instances) and less severe with 'Apps' (209 instances).

The above asymmetrical nature is a result of class imbalance which is discussed in several research studies to be strongly correlated with the problem of overlap [Das et al., 2014; Lee and Kim, 2018]. This overlap leads to poor accuracy for the minor class involved in the overlap. We measure the class accuracies in terms of the F1-score (i.e., the harmonic mean of precision and recall) [E. Zhang and Y. Zhang, 2009] which is a widely accepted measure for imbalanced datasets.

5.3.3 Rules Formulation

In order to formulate the rules for the overlapped classes, we first created a list of unique keywords for each overlapped class. These words were selected based on the domain knowledge of the experts (i.e., support agents) combined with the most important words (i.e., features) determined by the python library 'eli5' [Eli5, 2022]. The library helps to determine important words for each category based on their weights (see Table 5-1) which shows the weights based on the predictions of the linear SVM classifier on the Consumer Complaints dataset. Positive values (green highlight) indicate a high score and hence are considered important for the accuracy of the model; whereas, negative values (red highlight) indicate a low score and hence could be removed without affecting the classifier's accuracy.

Table 5-1: Linear SVM top 20 features and their weights displayed as an HTML table using the eli5 library

y=Credit card or prepaid card top features		**y=Credit reporting, credit repair services, or other personal consumer reports** top features		**y=Debt collection** top features		**y=Money transfer, virtual currency, or money service** top features	
Weight[?]	**Feature**	**Weight[?]**	**Feature**	**Weight[?]**	**Feature**	**Weight[?]**	**Feature**
+3.202	card	+3.903	experian	+2.777	debt	+5.608	coinbase
+3.061	reliacard	+3.736	transunion	+2.430	slandering	+3.572	transferwise
+2.793	circulating	+3.711	equifax	+2.428	breathe	+3.068	zelle
+2.677	waygo	+3.010	amendments	+2.383	possibilities	+3.058	paypal
+2.442	sapphire	+2.643	sagestream	+2.328	precisely	+2.971	bueruo
+2.430	rebilled	+2.628	trans	+2.276	pressler	+2.877	consolodate
+2.339	coaching	+2.489	sworn	+1.916	midland	+2.864	desparity
+2.326	paced	+2.426	inquiries	+1.868	gla	+2.722	adopted
... 23173 more positive ...		+2.391	relate	+1.866	trs	+2.712	worldremit
... 30529 more negative ...		+2.279	lexisnexis	+1.846	optimum	+2.672	square
-2.489	modification	+1.898	appl	+1.837	distributing	+2.661	paypals
-2.498	amendments	+1.893	freeze	+1.832	fbcs	+2.613	etoro
-2.691	escrow	+1.892	inquries	+1.803	trueaccord	+2.591	remitly
-2.759	debit	*... 29518 more positive ...*		+1.798	llp	+2.587	muitiple
-2.913	equifax	*... 35217 more negative ...*		*... 31384 more positive ...*		+2.491	postman
-3.131	loans	-1.871	debt	*... 36252 more negative ...*		+2.484	sonething
-3.254	navient	-1.921	posses	-1.792	mortgage	+2.460	legitament
-3.757	experian	-1.986	tracked	-1.822	profane	*... 12313 more positive ...*	
-3.860	mortgage	-2.060	distributing	-2.030	loan	*... 27140 more negative ...*	
-3.893	transunion	-2.139	coorporating	-2.669	transunion	-2.478	collection
-4.841	citigold	-2.270	suspects	-2.691	equifax	-3.142	ebay
-4.862	loan	-2.542	contesting	-2.868	experian	-4.100	equifax

The eli5 library was used to compensate for the lack of domain knowledge for the three public datasets used in this study as well as to enrich the list of keywords of our private dataset. There is no limit to the number of keywords selected for an overlapping class.

However, it would be difficult to guarantee the uniqueness of chosen words if the list is large. Second, a score is calculated for each overlapped class based on the number (i.e., count) of keywords present in the input/test sentence. The class with the highest score is selected as the target class. The number of rules is determined by the number of overlapped classes for a specific dataset. Hence, our rules are simple and easy to interpret and update by the support agents.

5.3.4 Algorithm

We propose a hybrid algorithm where rules are used as a pre-processing step to classify the N overlapped classes only. First, the incoming ticket is passed to our rule-based model, it would then be evaluated against the rules, and would either be successful (i.e., the incoming ticket contains one or more words specified in the list of keywords) and outputs the target label or it would then be passed to the ML model (LinearSVM) in our case. Our proposed Support Vector Machine Rule-based classifier "SVM-RB" algorithm is described in Algorithm 1.

Algorithm 1 (SVM-RB): Support Vector Machine Rule-Based Classifier

Input: Ticket *t*
Output: Class *C*
Initialize: *C1*- score, *C2*-score, ..., *Cn-score*=0. Where *n* is the number of overlapped classes
Method **Calculate-scores (*t*):**
C1-keywords = [list of words based on domain knowledge from experts + Top *k* words based on eli5]
..
Cn-keywords = [list of words based on domain knowledge from experts + Top *k* words based on eli5]
Calculate a score for each class Cn
C1-score = (foreach word in *t* present in *C1*-keywords *C1*-score++)
..
Cn-score = (foreach word in *t* present in *Cn*-keywords *Cn*-score++)
return C*1*-score, *C2*-score, *..., Cn*-score

if (*C1-score* ≥ *C2-score) & (C1-score* ≥ *C3-score) ... & (C1-score* ≥ *Cn-score) and C1-score* ≠ 0:
then → *C* = C1
return *C*
else if (*C2*-score ≥ *C1*-score) *& (C2-score* ≥ *C3-score) ... & (C2-score* ≥ *Cn-score*) and *C2*-score ≠ 0:
then → *C* = *C2*
return *C*
else if (*Cn*-score ≥*C1*-score) & (*Cn-score* ≥ *C2-score) ... & (Cn-score*≥ *Cn-1-score) and Cn*-score ≠ 0:
then → *C* = *Cn*
return *C*
else: Move to SVM classifier
1 V-text = Vectorize *t* using *TFIDF*
2 C = Predict on trained model (V-text)
return *C*
end

5.4 Experiments and Results

In this section, we discuss the learning algorithms used for this study and the results of applying our hybrid model (SVM-RB) to the four datasets described in Section 5.3.1. For this study, we chose three popular machine learning algorithms: support vector machines, logistic regression, and decision trees.

For the support vector machines, we use the LinearSVC algorithm from the Scikit-learn library [Pedregosa et al., 2011]. LinearSVC proved to be efficient for high-dimensional datasets as it achieves high classification accuracy with low training time [Chauhan et al., 2019].

Furthermore, the work in [Wahba et al., 2022b] shows that Linear SVM provides comparable performance to state-of-the-art Pre-trained Language Models (PLMs) (e.g., BERT).

Another reason why we choose LinearSVM for our hybrid model is that most text classification problems are linearly separable (i.e., if graphed in two dimensions, can be separated by a straight line) [Joachims, 1998] and thus mapping the data to a higher dimension space using an SVM kernel (e.g., RBF kernel) would be futile.

For decision trees, we chose eXtreme Gradient Boosting 'XGBoost', an efficient implementation of the gradient boosting framework by [Chen and Guestrin, 2016]. The number of trees for each dataset is determined based on a grid search using 10-fold cross-validation.

In all our experiments, we use stratified splitting [Sechidis et al., 2011] of the datasets into 70% for training samples and 30% for testing. Stratification ensures that the train and test have the same percentage of samples of each target class.

For text vectorization, we use TFIDF [Sammut and Webb, 2010] which is a simple yet powerful technique. Moreover, we use n-grams as sometimes single words are not sufficient to determine the category. For instance, for the real-world dataset of IT Support Tickets, we use trigrams (i.e., n=3) as they deliver the highest accuracy.

Table 5-2 shows a comparison of the accuracies (i.e., F1-scores) of the three chosen classification models against our proposed hybrid model (SVM-RB) on the overlapped

classes of the four datasets described in Section 5.3.1. The comparison data in Table 5-2 clearly show that our hybrid approach with rules integration performs significantly better than other classification models, in terms of improving the accuracy of all overlapped classes (higher accuracies in bold). For the IT support tickets dataset, our model achieves 53% accuracy for the 'Apps' class, which shows an increase of 17% higher than the highest accuracy achieved by the XGboost (36%). Similarly, the hybrid model achieves a 17% increase in accuracy for the 'Platform/Console' which is higher than the highest accuracy achieved by SVM (40%). However, for the 'Services' class, our model shows an increase of only 5% higher than the highest accuracy achieved by SVM (77%). This is because 'Services' is a major class with no significant overlap with other classes (see Figure 5-1).

For the MIND dataset, our model achieves a significant increase in the accuracy of the 'Video' class (70%) which shows an increase of 27% higher than the highest accuracy achieved by XGBoost (43%). Whereas an increase of 8% is achieved for the 'News' class with minor overlap with other classes (see confusion matrix Figure 5-1).

Similarly for the EAST and the Consumer Complaints datasets, our model shows a large increase in accuracy for class '6' and class 'Bank Account' with 28% and 14% respectively, while achieving a 4% and 5% increase for class '4' and class Checking/Saving'. It is to be noted that the performance of our proposed hybrid model depends mainly on the unique list of keywords chosen for each overlapped class in the pre-processing step using expert domain knowledge and the eli5 library.

Table 5-2: Comparison of SVM, LR, and XGboost against our proposed hybrid approach in terms of accuracy (F1-score)

Dataset	**Overlapped Classes**	**SVM**	**LR**	**XGBoost**	**Proposed Hybrid Model (SVM-RB)**
		F1-score (CV = 10-folds)			
IT Support Tickets	Apps	0.35	0.27	0.36	0.53
	Platform/Console	0.40	0.32	0.37	0.57
	Services	0.77	0.75	0.76	0.82
MIND	News	0.69	0.73	0.72	0.81
	Video	0.42	0.41	0.43	0.70

EAST	6	0.51	0.41	0.40	0.79
	4	0.90	0.91	0.91	0.95
Consumer Complaints	Bank Account	0.40	0.39	0.19	0.54
	Checking/Saving	0.63	0.67	0.61	0.72

To better assess the performance of our hybrid model to reduce the degree of overlap, we include the confusion matrix (Figure 5-2) of our proposed hybrid model for the four datasets. The proposed hybrid model shows a considerable reduction in the overlapping between the selected classes for all four datasets. For instance, for the IT Support Tickets, we notice the confusion between 'Apps' and 'Platform/Console' is reduced from '227'misclassified instances to '124'. Also, the confusion between 'Apps' and 'Services' is reduced from '181' to '82' instances. Hence, the total number of correctly classified instances for the 'Apps' class is increased from '272' to '536'.

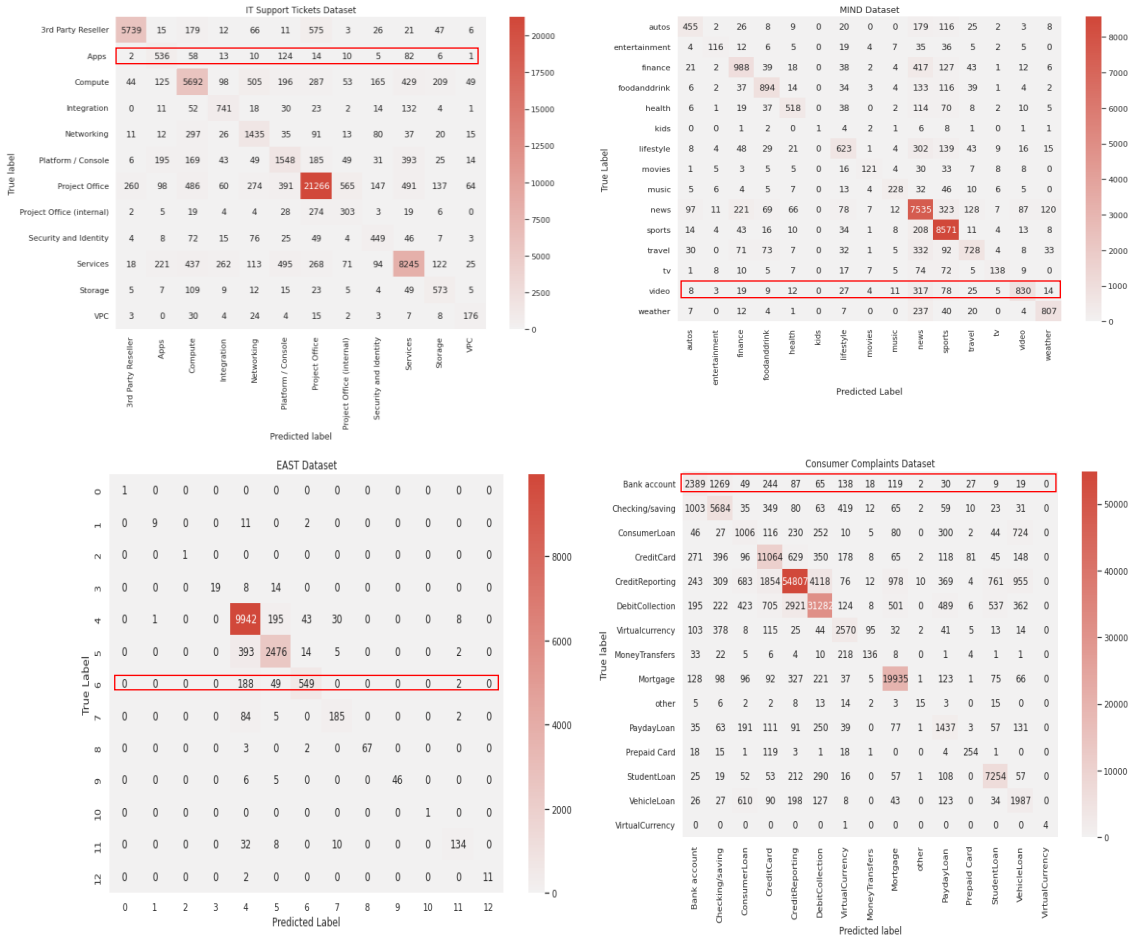


Figure 5-2: Confusion matrix for hybrid (SVM-RB) showing reduced overlap between classes for IT Support Tickets, MIND, EAST and Consumer Complaints datasets

Similarly, for the MIND dataset, our hybrid model reduces the confusion between the 'News' and 'Video' classes from '597' misclassified instances to '317'.
Furthermore, we notice that our hybrid model also improves the accuracy of all other classes that are not covered by the rules of our model. This can be noticed by looking at the diagonal of the confusion matrix (Figure 5-2), where almost all classes have a higher number of correctly classified instances. For instance, the number of correctly classified instances of the 'Project Office(internal)' class of the IT Support Tickets dataset was increased from 95 instances to 303.

Our experiments suggest that for domain-specific text classification tasks (e.g., IT Support Tickets) with a clear presence of class overlap, a simple linear model (e.g., SVM) along with a set of handcrafted rules can reduce the degree of overlapping as well as enhance the overall classification accuracy with the advantage of a fast-running time using a linear algorithm and better interpretability.

5.5 Conclusions and Future Work

The task of classifying IT support tickets becomes challenging as the number of classes grows and classes tend to overlap. This paper focuses on the task of classification of domain-specific text in the presence of clear overlap between two or more classes leading to poor accuracy for the minor class involved in the overlap.

We propose a hybrid method based on a linear SVM classifier and a rule-based algorithm. First, we detect classes involved in the overlap using the classifier's confusion matrix. Second, we generate N rules with minimal intervention from the support agents (i.e., domain expertise) and a python library (i.e., eli5). The number of these rules is determined by the number of overlapped classes for a given problem. Finally, the tickets are sent to the rule-based algorithm to filter the confusing N classes and if none of the rules apply, tickets are classified using the linear SVM classifier.

Results show that the proposed hybrid model achieves significant improvements over the three text classification algorithms namely (LR, SVM, and XGBoost) in terms of the F1-score. The hybrid linear model provides a cheap and interpretable solution to the problem of classifying support tickets in the presence of overlap. For future work, we

plan to enhance the hybrid algorithm to support hierarchical classification. Also, we intend to include more datasets in our studies.

References

[Bureau of Consumer Financial Protection, 2022] Consumer Complaint Database Homepage, https://www.consumerfinance.gov/data-research/consumer-complaints. (last accessed Oct. 18, 2022).

[Chauhan et al., 2019] Chauhan, V.K., Dahiya, K. and Sharma, A., 2019. Problem formulations and solvers in linear SVM: a review. Artificial Intelligence Review, 52(2), pp.803-855.

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C., 2016, August. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794). (Association for Computing Machinery).

[Cover and Hart, 1967] Cover, T.M., Hart, P.E., 1967. Nearest Neighbor Pattern Classification. IEEE Transactions on Information Theory. 13, pp. 21–27.

[Das et al., 2014] Das, B., Krishnan, N.C. and Cook, D.J., 2014. Handling imbalanced and overlapping classes in smart environments prompting dataset. In Data mining for service (pp. 199-219). Springer, Berlin, Heidelberg.

[E. Zhang and Y. Zhang, 2009] Zhang E., Zhang Y., 2009. F-Measure. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA.

[Eli5, 2022] Eli5 Homepage: https://eli5.readthedocs.io/en/latest/index.html. (last accessed Oct 17, 2022).

[Fu et al., 2015] Fu, M., Tian, Y. and Wu, F., 2015. Step-wise support vector machines for classification of overlapping samples. Neurocomputing, 155, pp.159-166.

[Georgakopoulos et al., 2018] Georgakopoulos, S.V., Tasoulis, S.K., Vrahatis, A.G. and Plagianakos, V.P. "Convolutional neural networks for toxic comment classification." In Proceedings of the 10th hellenic conference on artificial intelligence, pp. 1-6.

[Joachims, 1998] Joachims, T., 1998, April. Text categorization with Support Vector Machines: Learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds) Machine Learning: ECML-98. ECML 1998. Lecture Notes in Computer Science, vol 1398. Springer, Berlin, Heidelberg, pp.137–142.

[Kulkarni et al., 2020] Kulkarni, A., Chong, D. and Batarseh, F.A., 2020. Foundations of data imbalance and solutions for a data democracy. In data democracy (pp. 83-106). Academic Press.

[Lee and Kim, 2018] Lee, H.K. and Kim, S.B., 2018. An overlap-sensitive margin classifier for imbalanced and overlapping data. Expert Systems with Applications, 98, pp.72-83.

[Liu et al., 2019] Liu, H., Burnap, P., Alorainy, W. and Williams, M.L., 2019. A fuzzy approach to text classification with two-stage training for ambiguous instances. IEEE Transactions on Computational Social Systems, 6(2), pp.227-240.

[Pedregosa et al., 2011] sklearn.svm.LinearSVC. https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC (last accessed Oct. 17, 2022).

[Preda G., 2020] Github Homepage: https://github.com/gabrielpreda/Support-Tickets-Classification#22-dataset. (last accessed Oct. 17, 2022).

[Saeed et al., 2018] Saeed, H.H., Shahzad, K. and Kamiran, F., 2018, November. Overlapping toxic sentiment classification using deep neural architectures. In 2018 IEEE international conference on data mining workshops (ICDMW) (pp. 1361-1366). IEEE.

[Sammut and Webb, 2010] Sammut, C. and Webb, G.I., 2010. Tf–idf. Encyclopedia of machine learning, pp.986-987.

[Sechidis et al., 2011] Sechidis, K., Tsoumakas, G. and Vlahavas, I., 2011. On the stratification of multi-label data. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 145-158). Springer, Berlin, Heidelberg.

[Tang and Gao, 2007] Tang, Y. and Gao, J., 2007. Improved classification for problem involving overlapping patterns. IEICE TRANSACTIONS on Information and Systems, 90(11), pp.1787-1795.

[Trappenberg and Back, 2000] Trappenberg, T.P. and Back, A.D., 2000, July. A classification scheme for applications with ambiguous data. In Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium (Vol. 6, pp. 296-301). IEEE.

[Wahba et al., 2022a] Wahba, Y., Madhavji, N., Steinbacher, J., 2022. Reducing Misclassification Due to Overlapping Classes in Text Classification via Stacking Classifiers on Different Feature Subsets. In: Arai, K. (eds) Advances in Information and Communication. FICC 2022. Lecture Notes in Networks and Systems, vol 439. Springer, Cham (2022).

[Wahba et al., 2022b] Wahba, Y., Madhavji, N., Steinbacher, J., 2022. A Comparison of SVM against Pre-trained Language Models (PLMs) for Text Classification Tasks. In: International Conference on Machine Learning, Optimization, and Data Science. LOD 2022. Lecture Notes in Computer Science (LNCS), Springer, Cham (in press).

[Wu et al., 2020] Wu, F., Qiao, Y., Chen, J.H., Wu, C., Qi, T., Lian, J., Liu, D., Xie, X., Gao, J., Wu, W. and Zhou, M., 2020. "Mind: A large-scale dataset for news recommendation." In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 3597-3606.

[Xiong et al., 2010] Xiong, H., Wu, J. and Liu, L., 2010, December. Classification with class overlapping: A systematic study. In 1st International Conference on E-Business Intelligence (ICEBI 2010) (pp. 303-309). Atlantis Press.

[Xiong et al., 2013] Xiong, H., Li, M., Jiang, T., Zhao, S., 2013. Classification algorithm based on NB for class overlapping problem. Applied Mathematics and Information Sciences. 7, pp. 409–415.

[Zendesk, 2022] Zendesk Homepage: https://www.zendesk.com/blog/ticketing-system-tips/. (last accessed Oct. 17, 2022).

Abstract: For large-scale IT corpora with hundreds of classes organized in a hierarchy, the task of classifying support tickets is vital to guarantee long-term clients. Due to the complexity of the unstructured nature of human language, text classification is challenging. The task is even harder when classes overlap.

In the business world, an efficient and interpretable ML model is preferred over an expensive black-box model. In this paper, we propose a Hybrid Online Offline Model (HOOM) for efficient classification of hierarchical text documents using linear ML models.

Chapter 6: A Hybrid Continual Learning Approach for Efficient Hierarchical Classification of IT Support Tickets in A Real-World Scenario[16]

6.1 Introduction

Continual Learning (CL) [Parisi et al., 2019], also known as Lifelong learning (LL) [Chen and Liu, 2018] is inspired by the human intelligence to learn continuously. Humans learn from past experiences and accumulate the knowledge to improve generalization for future tasks [Chomsky, 2009]. CL which is commonly used in the deep learning field aims to solve a serious problem called catastrophic forgetting (CF) when learning a series of tasks [McCloskey and Cohen, 1989]. CF refers to a significant drop in performance on previous tasks.

In IT Support ticketing systems, users submit a support ticket as a bug report to the IT support team. With a wide user base and system issues, there will be an ongoing influx of generated support tickets. The need to automate ticket classification becomes crucial for rendering quality service and high customer satisfaction.

[16] A version of this chapter is to appear in (Wahba, Y., Madhavji, N. and Steinbacher, J., 2023. A Hybrid Continual Learning Approach for Efficient Hierarchical Classification of IT Support Tickets in A Real-World Scenario. The 24th IEEE International Conference on Industrial Technology (ICIT).

While traditional linear classifiers such as SVM or LR proved comparable to state-of-the-art (SOTA) deep-learning models for domain-specific text classification [Wahba et al., 2022c] (Chapter 3), their performance is not reliable for practical environments and real-world scenarios. This is because a traditional ML model cannot handle the problem of concept drift (i.e., changes in the underlying data distribution over time) [Widmer and Kubat, 1993; Sayed-Mouchaweh, 2016], which could also involve the emergence of new classes/features (i.e., target drift).

In order to ensure the effectiveness of the deployed ML classifiers over time, CL is employed to allow classifiers to adapt to new changes. In a typical CL scenario, the model receives a data stream one at a time and predicts a class label, then the model reveals the true label, and then updates the classifier, and repeats the process with the new incoming stream. This is different from traditional batch learning where we have all the data available when training our model.

The existing body of work on the topic of CL is oriented towards deep neural network models [Hadsell et al., 2020; van de Ven et al., 2020; Chaudhry et al., 2021]. However, the topic of applying CL to traditional (i.e., classical) ML models is scarcely discussed. This work adds to the growing literature on the topic of applying CL to traditional ML models for classifying domain-specific text (i.e., IT Support tickets).

In this paper, we propose a Hybrid Online Offline Model (HOOM) for efficient classification of hierarchical text documents. Hierarchical classification problems can be classified into two main categories: Hierarchical Single Label (HSL) and Hierarchical Multi-Label (HML). In HSL problems, instances/samples are classified into a single path of classes; whereas in HML, instances can have more than one label assigned to them. Our work is categorized as an *HSL* problem.

The motivation behind (HOOM) was realized during the evaluation phase of the pre-trained ML model. The pre-trained model performed well on the historical dataset. However, it suffered from a sudden drop in performance on new, unseen instances. The reason for this was a change in the original taxonomy where new classes were added to or removed from the hierarchy. This motivated us to think of integrating an online learning model that continuously learns about changes in the incoming data and passes this knowledge on to the offline pre-trained ML model

The experimental results on a private dataset of IT Support tickets show that the hybrid model (HOOM) provides superior results over the individual models and is anticipated to have a fast inference time given the underlying linear classifiers.

The rest of the paper is organized as follows. Section 6.2 describes related work. Section 6.3 describes our proposed hybrid model. Section 6.4 describes the dataset and Section 6.5 presents the research results. Section 6.6 concludes the paper.

6.2 Related Work

The research scope of this paper is somewhat diverse. Thus, we divide the related work into the following subsections:

I. Handling class overlapping in text classification problems

The problem of overlapping classes is extensively studied in the literature, however, only a few works addressed the overlapping problem in the NLP domain. We classify the literature approaches for handling overlap as non-deep learning [Liu et al., 2019; Wahba et al., 2022a] and deep-learning approaches [Saeed et al., 2018; Georgakopoulos et al., 2018]. Our study contributes to the non-deep learning approaches in the NLP domain. This study extends the work of [Wahba et al., 2022b] (Chapter 5) to address hierarchical text classification scenarios in the presence of class overlap. Our approach follows a top-down strategy using a linear SVM classifier as the base classifier.

II. Continual learning in text classification

The field of CL in the NLP domain is still nascent [Sun et al., 2019; Greco et al., 2019]. [Shu et al., 2016] follow an unsupervised CL approach to classify opinion targets. Furthermore, the work of [Shu et al., 2017] specifically contributes to supervised aspect extraction using conditional random fields. However, the work of [D'Autume et al., 2019] uses episodic memory to mitigate catastrophic forgetting in unsupervised text classification tasks. The majority of the literature on the topic of CL is geared towards deep-learning methods. However, the topic of applying CL to traditional (i.e., classical) ML models is scarcely discussed.

Our work contributes to the supervised CL approaches for text classification using traditional ML models. In particular, we study the problem of classifying overlapped

domain-specific text (i.e., IT Support tickets) in a CL environment. We propose a hybrid model based on an offline pre-trained linear classifier and an online classifier that can adapt to real drift (i.e., the emergence of new classes). The offline classifier acts as a backup model to the online classifier that is subject to the inevitable issue of CF [French, 1999].

6.3 Proposed Hybrid Online Offline Model (HOOM)

In this section, we first describe each of the offline and online models separately. Then we present our proposed Hybrid Online Offline Model (HOOM), which combines a static ML model trained in an offline setting with an online ML model trained in a dynamic (real-time) environment.

6.3.1 The Offline Model

The offline learning model is based on a hierarchical classifier called (HSVM-RB), which is an extension of the algorithm (SVM-RB) proposed in a previous study [Wahba et al., 2022a] (Chapter 5). The algorithm (HSVM-RB) extends the capabilities of (SVM-RB) to support hierarchical classification scenarios.

For the hierarchical classification, we utilize HiClass [Miranda et al., 2021]. An open-source Python library that contains implementations for the most common design patterns found in the literature (e.g., local classifier per node, local classifier per level, etc.).

For the model (HSVM-RB), we employ a top-down approach called *Local Classifier Per Parent Node* (LCPPN) [Silla and Freitas, 2011]. In this approach, for each parent node in the class hierarchy, a multi-class classifier (i.e., SVM) is trained to differentiate between its child nodes. This approach avoids the problem of inconsistent predictions (i.e., prediction does not satisfy the ancestral relations for some class C) and respects the natural class hierarchy memberships.

We chose the linear SVM, which proved to be efficient for high-dimensional datasets with superior accuracy and low training time [Chauhan et al., 2019]. Furthermore, the work of [Wahba et al., 2022c] (Chapter 3) shows that Linear SVM provides comparable performance to Pre-trained Language Models (PLMs) (e.g., BERT). Another reason why we choose a linear kernel for SVM is that most text classification problems are linearly

separable [Joachims, 1998]. This is because text datasets are characterized by a high number of features that inaugurate the linear separability of the data.

Algorithm 1 (HSVM-RB): Hierarchical Support Vector Machine Rule-Based Classifier

Input: Ticket *t*

Output: Class *C-1, C-2* (numbers indicate the level in the hierarchy)

Initialize: *C1*- score, *C2*-score, …, *Cn-score=0*.Where *n* is the number of overlapped classes in the top level only.

Initialize: LocalClassifierPerParentNode (LCPPN) → Local-classifier = SVM

Method **Calculate-scores (*t*):**

C1-keywords = [list of words based on domain knowledge from experts + Top *k* words based on eli5]

..

Cn-keywords = [list of words based on domain knowledge from experts + Top *k* words based on eli5]

Calculate a score for each class Cn

C1-score = (foreach word in *t* present in *C1*-keywords *C1*-score++)

..

Cn-score = (foreach word in *t* present in *Cn*-keywords *Cn*-score++)

return C*1*-score, *C2*-score, *..., Cn*-score

if (*C1-score ≥ C2-score) & (C1-score ≥ C3-score) ... & (C1-score ≥ Cn-score) and C1-score >2*:

then → *C-1* = C1

V-text = Vectorize *t* using *TFIDF*

C-2 = SVM.predict (V-text)

return *C-1,C-2*

else if (*C2*-score ≥ *C1*-score) *& (C2-score ≥ C3-score) ... & (C2-score ≥ Cn-score*) *and C2-score >2:*

then → *C-1 = C2*

V-text = Vectorize *t* using *TFIDF*

C-2 = SVM.predict (V-text)

return *C-1, C-2*

else if (*Cn*-score ≥*C1*-score) & (*Cn-score ≥ C2-score) ... & (Cn-score≥ Cn-1-score) and Cn*-score >2:

then → *C-1 = Cn*

V-text = Vectorize *t* using *TFIDF*

C-2 = SVM.predict (V-text)

return *C-1, C-2*

else: Move to Hierarchical SVM classifier

1. V-text = Vectorize *t* using *TFIDF*
2. *C-1, C-2* = LCPPN pipeline predict (V-text)

return *C-1, C-2*

end

HSVM-RB (described in Algorithm 1) is a hybrid model that uses rules to classify N overlapped classes determined in the Exploratory Data Analysis (EDA) stage (see Chapter5).

The incoming ticket is first checked against the hand-crafted rules based on the N overlapped classes. If *true*, the class in the first level (level-1) is classified based on the rules, and the lower levels are predicted based on SVM classifier trained only on the parent node. If *false*, the incoming ticket is classified using a (LCPPN) approach with SVM as the local classifier.

6.3.2 The Online Model

In the online learning model, the learning is performed in a dynamic environment as data arrives one after another. To accomplish the task of CL in a streaming setting, we utilize a recent open-source Python library called *River* [Montiel et al., 2021]. The library provides several machine learning algorithms such as Decision Trees (DT), Naïve Bayes (NB), and Logistic Regression (LR). The source code for the library is available on Github[17]. The online ML model is based on a Passive Aggressive Classifier (PAC), that is first proposed by Crammer [Crammer et al., 2006]. This classifier belongs to a family of margin-based online learning algorithms, that can handle large datasets.

In each iteration, PAC takes in a new instance, checks whether it has been correctly classified or not, and then updates its weights accordingly. If the instance is correctly classified, there is no change in weight. However, if it is misclassified, the classifier adjusts its weights to better classify future instances. The degree to which the PAC adjusts its weights is based on a regularization parameter *C*.

The PAC is referred to as PAC Pipeline (Figure 6-1). We use 'pipeline' to denote a two-step task where the text is first vectorized by TFIDF[18] technique and then classified using the PAC.

---

[17] https://github.com/online-ml/river

[18] TFIDF stands for Term Frequency-Inverse Document Frequency, which is a combination of two metrics: 1. Term frequency (*tf*): a measure of how frequently a term, t, appears in a document, d.
2. Inverse document frequency(*idf*): a measure of how important a term is. It is computed by dividing the total number of documents in our corpus by the document frequency for each term and then applying logarithmic scaling on the result.

6.3.3 Hybrid Model: HOOM

Figure 6-1 shows the proposed hybrid model that combines a pre-trained classifier (i.e., offline learning model) and an online classifier (i.e., online learning model). The offline model serves as a backup model to the online classifier and is initially trained on a large amount of historical data.

Input: The input to the hybrid model is a data point *x* (i.e., a support ticket) fetched from an online stream of incoming data. The data point is received by both offline and online classifiers and undergoes the step of pre-processing. Then the data point is passed to the offline classifier described in Section 6.3.1 and the online classifier described in Section 6.3.2.

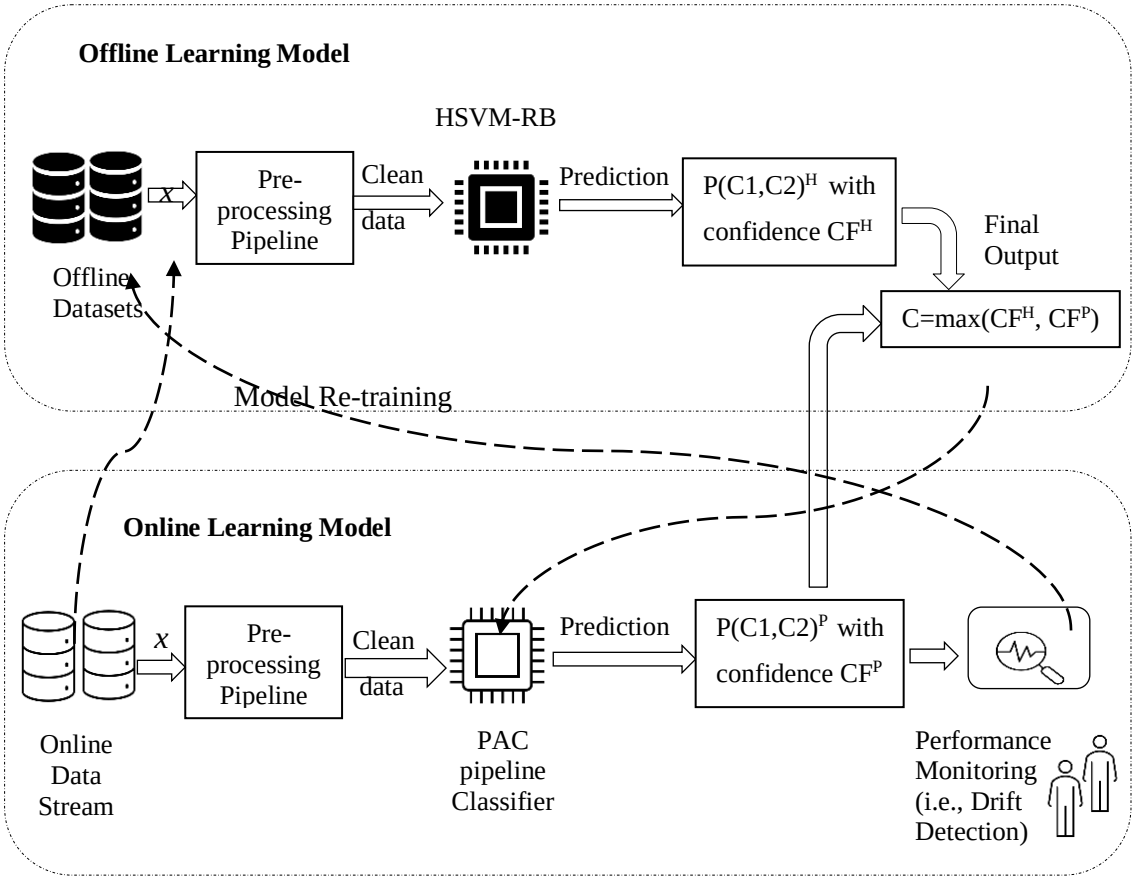


Figure 6-1: Proposed Hybrid Online Offline Model (HOOM)

Output: The prediction (i.e., output) of the offline hybrid classifier (HSVM-RB) is a class label $P(C1, C2)^H$ (where C1 denotes the class in the first level and C2 denotes the class in the second level) with a confidence score denoted by CF^H. Similarly, the output of the online (PAC) is a class label $P(C1, C2)^P$ with a confidence score denoted by CF^P. The higher the confidence score, the more confident the model's prediction is. The final output of HOOM is the model with the highest confidence score (i.e., max (CF^H, CF^P). The calculation of confidence scores is described in Section 6.3.4.

Drift Detection: The final step of HOOM is a proactive measure against concept drift or what is known as data drift. Concept drift is a serious problem in production where the incoming data stream differs from the historical data the ML model was trained and evaluated on, leading to performance degradation [Bayram et al., 2022].

The topic of concept drift is extensively researched and there exist different categorizations for the term 'concept drift'. We follow the work of [Straat et al., 2022] and [Gama et al., 2014] where concept drift is categorized into two major types: *Virtual drift* and *Real drift*. A virtual drift refers to changes in the distribution (i.e., statistical properties) of the incoming data without affecting the target data. Real drift refers to a change in the target data (i.e., classification scheme) over time. The focus of this work is real drift, where similar data points are labeled differently over time.

There are several drift detection methods proposed in the literature categorized according to the test statistics they apply (e.g., error rate-based methods, data distribution-based methods, and multiple hypothesis tests) [Lu et al., 2018]. We employ a data-distribution-based method that is the Chi-square test [Maaradji et al., 2017].

6.3.4 Confidence Scores

For a given classification task, a confidence score is considered an evaluation metric for the classifier to indicate how confident the classifier's prediction is correct. It calculates the probability of the predicted class label by the classifier given as a percentage.

Below, we describe how the confidence scores are calculated for the proposed model HOOM.

Offline model: The offline model is based on a hybrid classifier (i.e., HSVM-RB), hence, our confidence score is calculated in two different ways: (1) If the incoming data point *x* is classified by the Rule-Based classifier, the metric we use to estimate the confidence of the match is called *coverage*.

The calculation of the *coverage* score takes into account how many words are matched (i.e., *Cn-score;* where n is the number of overlapped classes in the top level only) (See Algorithm 1). The coverage score is calculated as follows:

$$Coverage = \frac{1}{frequency(Cn-score)} \times 100 \qquad (1)$$

This implies that predictions by the rule-based classifier would always have a 100% confidence score unless two or more classes have the same *Cn-score* (i.e., the number of words matched), then the confidence score for our model decreases.

For example: Assume n=3:

If (*C1-score* > *C2-score*) and (*C1-score*>*C3-score*) THEN *coverage* = 100%

If (*C1-score*=*C2-score)* and (*C1-score*>*C3-score*) THEN *coverage* =50%

(2) if the incoming data point *x* is classified by the Support Vector Machine classifier (i.e., did not match any of the rules), the confidence score is the calibrated probability [Rüping, 2006] of the data point *x* belonging to class C. For the calibration function, we use Platt's method (i.e., a method for transforming SVM outputs from [−∞, +∞] to posterior probabilities) [Platt, 1999] which is shown to work well with maximum margin classifiers such as SVM [Niculescu-Mizil and Caruana, 2005].

Online model: Similar to SVM calibrated probabilities for the offline model, the confidence scores for the online PAC are calculated based on Platt's method and given as a percentage score.

6.4 The Dataset

We use a private dataset of IT support tickets, that is obtained from a large industrial partner with real customer issues concerning a cloud-based system. The dataset is a

hierarchical dataset that is composed of 194,488 documents categorized into 12 classes on the first level of the hierarchy and 110 classes on the second level.

This dataset is used to train and evaluate the offline model and will be referred to as **D1**. However, for the sake of testing (HOOM), we use a recent dump of support tickets, that was collected by pulling the tickets from the server and passing them as a stream (i.e., one by one) to the hybrid model. This recent dump is composed of 200,000 instances of support tickets and will be referred to as **D2**.

6.5 Results

In this section, we present the accuracies of the offline and the online model separately. Then we describe how we detect concept drift. Finally, we present the performance of the hybrid model (HOOM).

Offline model: First, to evaluate the effectiveness of the rule-based model (HSVM-RB), we present the accuracies of a hierarchical SVM on 'level-1' of the hierarchy (see Figure 6-2), then we present the accuracies of the offline classifier (HSVM-RB) on 'level-1' of the hierarchy (see Figure 6-3).

Figure 6-2 shows the overall accuracy of a linear SVM in predicting the accuracies of the first level of the hierarchy using the (LCPPN) approach. The red highlight indicates the classes that suffer from poor accuracies due to overlap (see Chapter 5). The model achieves an overall F1-score (i.e., weighted avg) of **78%**.

Testing Accuracy for HiClass (F1-score): {0.6966596397415463}

	precision	recall	f1-score	support
3rd Party Reseller	0.96	0.85	0.90	6728
Apps	0.49	0.21	0.30	859
Compute	0.67	0.76	0.72	7904
Integration	0.72	0.51	0.60	1036
Networking	0.66	0.57	0.61	2099
Platform / Console	0.50	0.31	0.39	2670
Project Office	0.88	0.93	0.90	24276
Project Office (internal)	0.40	0.05	0.09	664
Security and Identity	0.59	0.25	0.35	752
Services	0.72	0.83	0.77	10225
Storage	0.69	0.51	0.59	839
VPC	0.79	0.53	0.63	295
accuracy			0.79	58347
macro avg	0.67	0.53	0.57	58347
weighted avg	0.78	0.79	0.78	58347

Figure 6-2: Accuracies of the hierarchical SVM on 'level-1' of dataset D1

Figure 6-3 shows that the hierarchical rule-based model (HSVM-RB) provides a significant improvement for almost all classes of the first level. For instance, an increase of **15%** is achieved for the 'Apps' class, while for 'Project Office (internal)', the model shows a substantial increase of '**70%**'. Also, an increase of '**16%**' is achieved for 'Platform/Console', and an increase of '**49%**' for the 'Security and Identity' class.

Testing Accuracy for HybridHiersvm (F1-score): {0.7821310435840745}

	precision	recall	f1-score	support
3rd Party Reseller	0.98	0.91	0.94	6728
Apps	0.32	0.78	0.45	859
Compute	0.99	0.74	0.85	7904
Integration	1.00	0.65	0.79	1036
Networking	0.99	0.86	0.92	2099
Platform / Console	0.46	0.70	0.55	2670
Project Office	0.97	0.75	0.85	24276
Project Office (internal)	1.00	0.65	0.79	664
Security and Identity	1.00	0.72	0.84	752
Services	0.55	0.91	0.69	10225
Storage	1.00	0.75	0.86	839
VPC	1.00	0.78	0.88	295
accuracy			0.80	58347
macro avg	0.85	0.77	0.78	58347
weighted avg	0.87	0.80	0.81	58347

For the second level of the hierarchy (102 classes), Figure 6-4 shows a portion of 'level-2' classes wi

Figure 6-3: Accuracies of (HSVM-RB) on 'level-1' of dataset D1

SSL Certificate	1.00	0.67	0.80	64
Secure Gateway	1.00	0.63	0.77	97
Security	0.73	0.76	0.74	97
Security (Infrastructure)	1.00	0.67	0.80	9
Service Dispute	1.00	0.33	0.50	24
Service Dispute (Internal)	0.00	0.00	0.00	1
Starter Kits	0.71	0.86	0.78	35
Storage (Infrastructure)	1.00	0.54	0.70	28
Storage (Services)	0.62	0.78	0.69	156
Subnet and IP	0.99	0.83	0.90	497
Subscription Extension	1.00	0.65	0.79	136
Subscription Extension (Internal)	0.00	0.00	0.00	2
Terminations	0.92	0.68	0.78	2934
Terminations (Internal)	1.00	0.53	0.69	32
Trial Extensions	1.00	0.45	0.62	101
Trial Extensions (Internal)	1.00	0.60	0.75	5
VLAN Spanning	0.98	0.90	0.94	210
VMware	1.00	0.78	0.88	717
Virtual Server Instance	0.97	0.82	0.89	2033
Vyatta	1.00	0.84	0.91	488
Watson	0.42	0.96	0.58	3600
WebSphere Application Server	1.00	0.72	0.84	25
accuracy			0.75	58347
macro avg	0.87	0.66	0.72	58347
weighted avg	0.84	0.75	0.77	58347

Figure 6-4: Truncated accuracies of (HSVM-RB) on level-2 of dataset D1

Online model: The online PAC implemented in River [Montiel et al., 2021] does not support hierarchical classification. Therefore, unlike the 'top-down' approach (i.e., LCPPN) used for the offline classifier, we use a 'flat' classification approach for the PAC [Silla and Freitas, 2011]. This approach implies implicit assignment of the ancestor classes (i.e., level-1) to the leaf classes (i.e., level-2) that are predicted by the PAC.

Figure 6-5 shows two different performance measures (i.e., accuracy and weighted F1-score) of the online PAC that takes a data stream as input (i.e., continuous flow of instances). We tested the PAC on the 200k instances of D2.

It is clear from the figure that the performance of the online classifier is affected by the number of iterations (i.e., instances). The more data streams, the better the performance. The accuracy score for the first iteration is zero as the model is predicting with no prior knowledge (i.e., training). Then the performance of the model starts to improve as more streams arrive. The highest accuracy achieved on 200k iterations is around 65%.

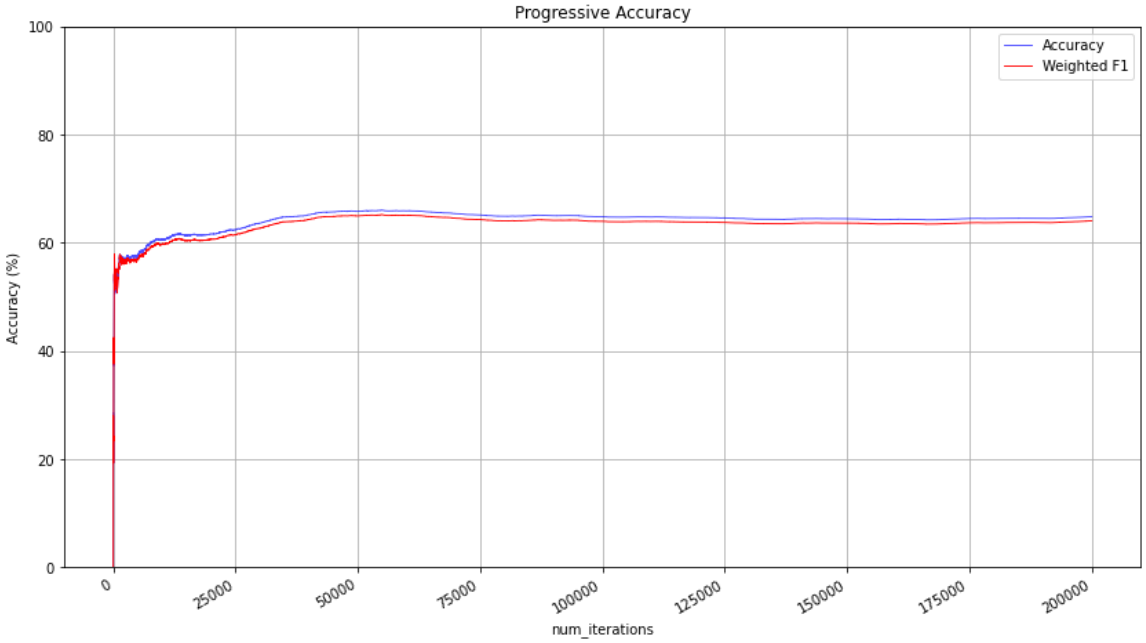


Figure 6-5: The accuracy of PAC on the 200k instances of D2

Concept Drift: Checking for concept drifts can be performed manually or can be embedded in the online ML model where an alarm is triggered upon the detection of a drift.

For the purpose of this paper, we perform a manual target drift detection after *N* iterations where *N*=200k (i.e., the number of instances of D2) using the Chi-square test [Maaradji et al., 2017]. We use two distributions of the same size: the first distribution is the one the model is trained and evaluated on (i.e., D1) and we call that a *reference* distribution. The other distribution is built from the most recent runs pulled from the incoming data streams and we call that a *current* distribution (i.e., D2).

Figure 6-6 shows the drift in category distributions and the emergence of new classes (e.g., Infrastructure, Sales, and Sales Office).

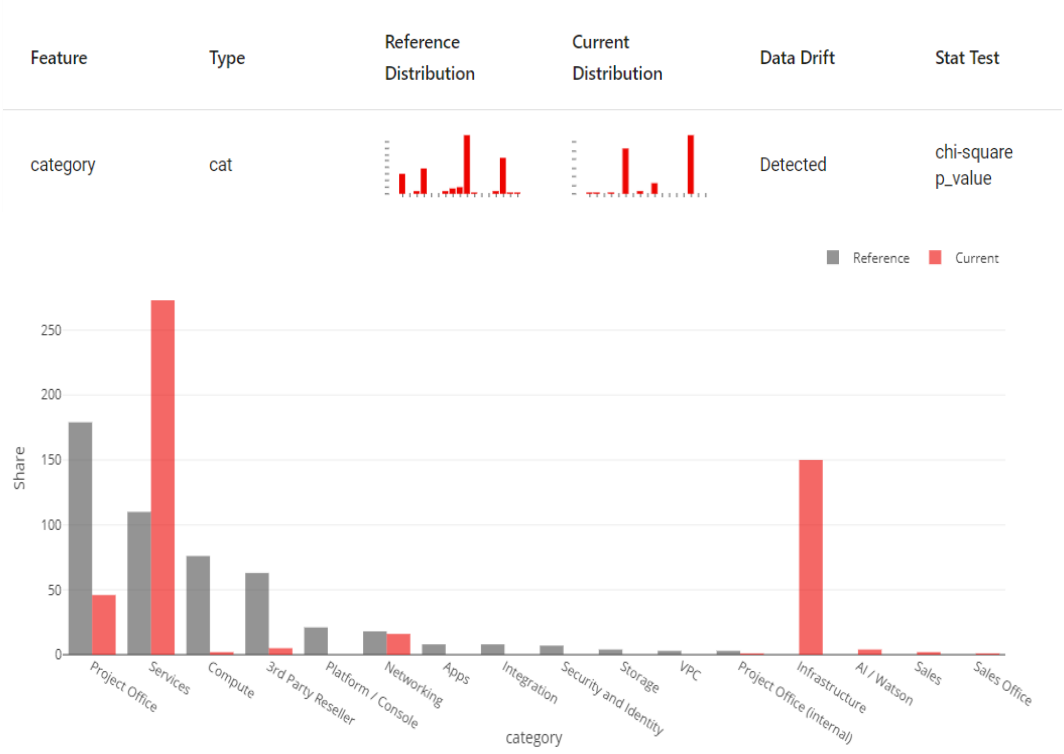


Figure 6-6: Detecting target drifts using two data distributions (i.e., D1 and D2)

Hybrid model (HOOM): To assess the performance of the proposed hybrid model (HOOM), we use the recent dump (i.e., D2) of our support tickets. As mentioned earlier, the offline model is re-trained on D2 after N=200k iterations. The model with the highest confidence score (see Section 6.3.4) is the model that determines the class label.

Figure 6-7 shows the accuracy of (HOOM) on D2. We note that the accuracy score of the first iteration is not zero as the model is getting predictions from the offline classifier. The highest accuracy achieved on 200k iterations is around 87%.

The performance of the model shows a significant increase of 20% over the online model (PAC) (Figure 6-8). We note that we cannot compare HOOM to the offline model (HSVM-RB) as it is not possible to train a batch model (i.e., offline model) on a data stream. Overall, HOOM has demonstrated promising results in classifying IT support tickets in a simulated real-time environment.

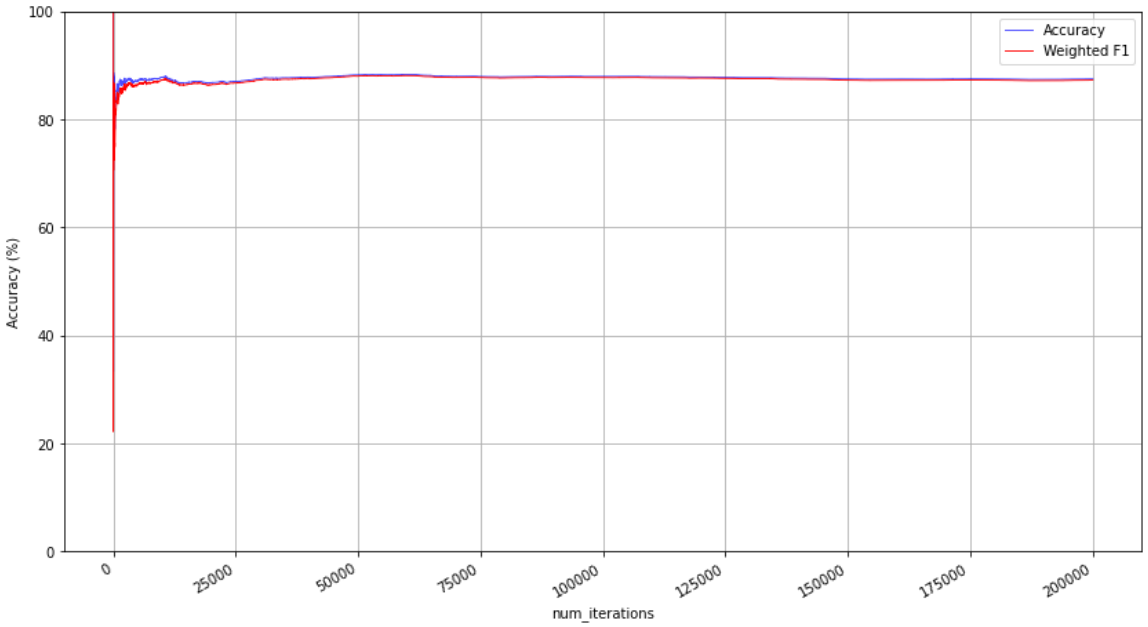


Figure 6-7: Accuracy of HOOM on the 200k instances of D2

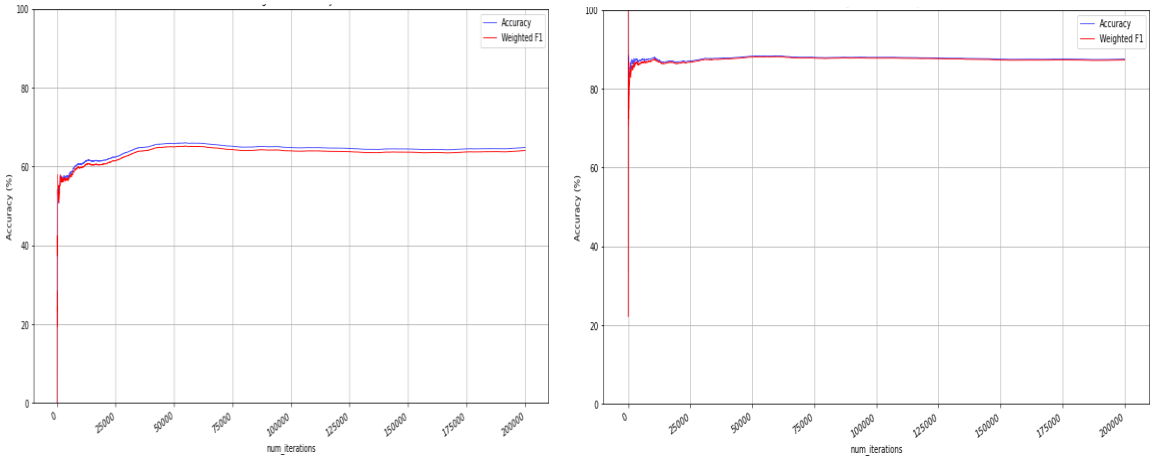


Figure 6-8: Performance of HOOM on D2 showing 20% increase in accuracy

6.6 Conclusions and Future Work

Classifying customer support tickets is fundamental to any help desk system. Automation of the tickets' classification help improve the resolution time significantly and minimize errors in the escalation process. However, a problem that appears when deploying a classification ML model into production (i.e., real-time environment) is the emergence of new classes. This is known as concept drift (or real drift).

In this paper, we propose a hybrid Online Offline Model (HOOM) that is based on the combined predictions of a pre-trained offline model and an online model. The offline model is based on a hierarchical rule-based model that can handle class overlaps. The purpose of the offline model is to serve as a backup model to the online classifier which is subject to the issue of catastrophic forgetting.

Results showed that the proposed hybrid model (HOOM) is promising if deployed in a real-time environment. The model achieves good classification accuracy and would exhibit a fast inference time due to the underlying linear models (i.e., SVM and PAC). For future work, we plan to study the *virtual* concept drift and include more evaluation metrics.

References

[Bayram et al., 2022] Bayram, F., Ahmed, B.S. and Kassler, A., 2022. From concept drift to model degradation: An overview on performance-aware drift detectors. Knowledge-Based Systems, p.108632.

[Chaudhry et al., 2021] Chaudhry, A., Gordo, A., Dokania, P., Torr, P. and Lopez-Paz, D., 2021, May. Using hindsight to anchor past knowledge in continual learning. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 35, No. 8, pp. 6993-7001).

[Chen and Liu, 2018] Chen, Z. and Liu, B., 2018. Lifelong machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 12(3), pp.1-207.

[Chomsky, 2009] Chomsky, N., 2009. Syntactic structures. In Syntactic Structures. De Gruyter Mouton.

[Crammer et al., 2006] Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S. and Singer, Y., 2006. Online Passive-Aggressive Algorithms. Journal of Machine Learning Research, 7(19), pp.551-585.

[D'Autume et al., 2019] de Masson D'Autume, C., Ruder, S., Kong, L. and Yogatama, D., 2019. Episodic memory in lifelong language learning. Advances in Neural Information Processing Systems, 32.

[French, 1999] French, R.M., 1999. Catastrophic forgetting in connectionist networks. Trends in cognitive sciences, 3(4), pp.128-135.

[Gama et al., 2014] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M. and Bouchachia, A., 2014. A survey on concept drift adaptation. ACM computing surveys (CSUR), 46(4), pp.1-37.

[Greco et al., 2019] Greco, C., Plank, B., Fernández, R. and Bernardi, R., 2019, July. Psycholinguistics Meets Continual Learning: Measuring Catastrophic Forgetting in Visual Question Answering. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (pp. 3601-3605).

[Hadsell et al., 2020] Hadsell, R., Rao, D., Rusu, A.A. and Pascanu, R., 2020. Embracing change: Continual learning in deep neural networks. Trends in cognitive sciences, 24(12), pp.1028-1040.

[Lu et al., 2018] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J. and Zhang, G., 2018. Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering, 31(12), pp.2346-2363.

[Maaradji et al., 2017] Maaradji, A., Dumas, M., La Rosa, M. and Ostovar, A., 2017. Detecting sudden and gradual drifts in business processes from execution traces. IEEE Transactions on Knowledge and Data Engineering, 29(10), pp.2140-2154.

[McCloskey and Cohen, 1989] McCloskey, M. and Cohen, N.J., 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In Psychology of learning and motivation (Vol. 24, pp. 109-165). Academic Press.

[Miranda et al., 2021] Miranda, F.M., Köehnecke, N. and Renard, B.Y., 2021. HiClass: a Python library for local hierarchical classification compatible with scikit-learn. arXiv preprint arXiv:2112.06560.

[Montiel et al., 2021] Montiel, J., Halford, M., Mastelini, S.M., Bolmier, G., Sourty, R., Vaysse, R., Zouitine, A., Gomes, H.M., Read, J., Abdessalem, T. and Bifet, A., 2021. River: machine learning for streaming data in Python. Journal of Machine Learning Research (JMLR), 22(110), pp.1-8.

[Niculescu-Mizil and Caruana, 2005] Niculescu-Mizil, A. and Caruana, R., 2005, August. Predicting good probabilities with supervised learning. In Proceedings of the 22nd international conference on Machine learning (ICML) (pp. 625-632).

[Parisi et al., 2019] Parisi, G.I., Kemker, R., Part, J.L., Kanan, C. and Wermter, S., 2019. Continual lifelong learning with neural networks: A review. Neural Networks, 113, pp.54-71.

[Platt, 1999] Platt, J., 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Smola, A., Bartlett, P., Sch¨olkopf, B., Schuurmans, D., eds.: Advances in Large Margin Classifiers. MIT Press, 10(3), pp.61-74.

[Rüping , 2006] Rüping, S., 2006, September. Robust probabilistic calibration. In European Conference on Machine Learning (pp. 743-750). Springer, Berlin, Heidelberg.

[Sayed-Mouchaweh, 2016] Sayed-Mouchaweh, M., 2016. Learning from data streams in dynamic environments. Berlin: Springer International Publishing.

[Shu et al., 2016] Shu, L., Liu, B., Xu, H. and Kim, A., 2016, November. Lifelong-rl: Lifelong relaxation labeling for separating entities and aspects in opinion targets. In Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing (Vol. 2016, p. 225). NIH Public Access.

[Shu et al., 2017] Shu, L., Xu, H. and Liu, B., 2017, July. Lifelong Learning CRF for Supervised Aspect Extraction. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 148-154).

[Silla and Freitas, 2011] Silla, C.N. and Freitas, A.A., 2011. A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery, 22(1), pp.31-72.

[Straat et al., 2022] Straat, M., Abadi, F., Kan, Z., Göpfert, C., Hammer, B. and Biehl, M., 2022. Supervised learning in the presence of concept drift: a modelling framework. Neural Computing and Applications, 34(1), pp.101-118.

[Sun et al., 2019] Sun, F.K., Ho, C.H. and Lee, H.Y., 2019, September. LAMOL: LAnguage MOdeling for Lifelong Language Learning. In International Conference on Learning Representations.

[van de Ven et al., 2020] van de Ven, G.M., Siegelmann, H.T. and Tolias, A.S., 2020. Brain-inspired replay for continual learning with artificial neural networks. Nature communications, 11(1), pp.1-14.

[Wahba et al., 2022a] Wahba, Y., Madhavji, N., Steinbacher, J., 2022. Reducing Misclassification Due to Overlapping Classes in Text Classification via Stacking Classifiers on Different Feature Subsets. In: Arai, K. (eds) Advances in Information and Communication. FICC 2022. Lecture Notes in Networks and Systems, vol 439. Springer, Cham (2022).

[Wahba et al., 2022b] Wahba, Y., Madhavji, N., Steinbacher, J., 2022. A Hybrid Machine Learning Model for Efficient Classification of IT Support Tickets in The Presence of Class Overlap, In Proceedings of the 32nd Annual International Conference on Computer Science and Software Engineering (CASCON22) (pp. 151-156).

[Wahba et al., 2022c] Wahba, Y., Madhavji, N.H. and Steinbacher, J., 2022. A Comparison of SVM against Pre-trained Language Models (PLMs) for Text Classification Tasks. In 8th International Conference on Machine Learning, Optimization, and Data Science (LOD 2022), Lecture Notes in Computer Science (LNCS), Springer, Cham.

[Widmer and Kubat, 1993] Widmer, G. and Kubat, M., 1993, April. Effective learning in dynamic environments by explicit context tracking. In European Conference on Machine Learning (pp. 227-243). Springer, Berlin, Heidelberg.

Chapter 7: Reflection

In this chapter, we reflect upon the issues found in Chapters (2,4, and 5) published as research papers. Then we describe a theory that is emerging from this work.

7.1 Introduction

This thesis presents an efficient solution to address a Natural Language Processing (NLP) problem, that of Text Classification (TC).

At the beginning of this research, we were curious to evaluate different methods for vectorizing text (i.e., the first step in TC). This is reflected in Chapters 2 and 3 where we evaluate both static and dynamic word vectorization techniques against a traditional TFIDF method.

In Chapter 2, despite the comparable performance of TFIDF against the three static word embeddings [Wahba et al., 2020], we feel that we need to experiment with more static words embeddings such as fastText [Bojanowski et al., 2017] and include more datasets to give more ground truth to our claim.

In Chapter 4, we utilized stacking (i.e., blending) of machine learning models based on different feature subsets as a way to overcome the problem of overlapping classes. Although the technique proposed shows improvements in terms of the accuracy score of the overlapped classes, the following issues during the testing stage rendered the technique not suitable for a real-world setting (i.e., productization):

1. Understanding the logic behind the predictions of the stacked model was not straightforward. Thus, the algorithm did not meet the explainability/interpretability criteria.
2. The computational complexity (i.e., time and resources) required to re-train the algorithm with more base models (i.e., classifiers) was high. Thus, the algorithm did not meet the efficiency criteria from the point of view of our industrial partner.

Hence, with the above criteria in mind, we propose an algorithm ‘SVM-RB’ in Chapter 5. However, when evaluating the model on more datasets, we realized the need for a threshold for the scores. For instance, a score of 1 indicates the presence of only one word in the incoming ticket that appears in one of the N keywords lists, which is not

enough to classify the incoming ticket to a certain category. Hence, we consider this threshold in the *hierarchical* version of our algorithm that we call 'HSVM-RB' described in Chapter 6.

7.2 Emerging Theory

In this section, we develop a theory (that we prefer to call at this stage, "an emerging theory" because it is borne out of the results of one thesis and we believe that it requires a community's concurrence in results in order to solidify the emergence into a concrete theory over time) that postulates the following:

1. Text classification tasks, especially domain-specific tasks, do not benefit from the rich linguistic knowledge of state-of-the-art language models (i.e., PLMs) such as BERT [Wahba et al., 2022] (Chapter 3).
2. Domain-specific text classification tasks such as IT Support tickets can be tackled efficiently using a traditional ML model such as SVM that provides a cheap, interpretable, and efficient alternative to a complex DL model [Wahba et al., 2022].

We note that the emerging theory statement is not underestimating the power of DL models for classification tasks. It is simply arguing against the use of such complex models for classifying specialized text where words have precise meanings (i.e., monosemy) [Aronoff and Rees-Miller, 2020].

For instance, sentiment analysis (or sentiment classification) is one use-case of text classification; however, words that express sentiment have fuzzy meanings (i.e., polysemy). An example of how the word 'funny' could be classified as 'happy' or 'suspicious' is found in [Song et al., 2020].

7.3 Evaluation of Emerging Theory

To evaluate the goodness of the proposed emerging theory, we first describe the evaluation criteria used according to [Boehm and Jain, 2006] and [Sjøberg et al., 2008], then we evaluate the emerging theory in Section 7.3.2.

7.3.1 Evaluation Criteria

1. Generality: Does the theory cover a wide range of situations and concerns (e.g., procedural, technical, economic, and human)?
2. Parsimony (i.e., simplicity): Does the theory avoid excess complexity? Is it simple to understand, learn, and apply?
3. Explanatory power: The degree to which a theory accounts for and predicts all known observations within its scope, is simple in that it has few ad hoc assumptions, and relates to that which is already well understood.
4. Empirical support: The degree to which a theory is supported by empirical studies that confirm its validity.
5. Utility: The degree to which a theory supports the relevant areas of the software industry.
6. Testability: The degree to which a theory is constructed such that empirical refutation is possible.

7.3.2 Theory Evaluation

Generality. The scope of the emerging theory covers specialized text classification such as IT Support tickets. Hence, it is deemed generalizable to other domains characterized by specialized vocabularies such as health care, mathematics, and law.

However, the empirical evidence from which the emerging theory is derived is based solely on experiments on datasets related to IT and news and does not consider other domains.

Therefore, the generality of the emerging theory is considered low to moderate.

Parsimony. The emerging theory is inspired by 'Occam's razor' in the sense of promoting the use of simple ML models over complex DL models for domain-specific text classification tasks.
Therefore, the parsimony of the emerging theory is considered high.

Explanatory power. The analogy (i.e., the degree to which a theory is supported by analogy to well-established theories) is low. The emerging theory's ability to provide explanations of why the theory is true is based on observations and experiments conducted in the thesis and not on well-established theories. With increased experimentation and consistent positive observations, there may be a gain in the confidence in the proposed theory.
Therefore, the explanatory power of the emerging theory is considered low to moderate.

Empirical support. The emerging theory is derived from empirical research [Wahba et al., 2022]. However, the number of testing benchmarks is considered low. If more empirical studies are conducted on other domain-specific benchmarks, it would enhance the empirical support of the theory.
Therefore, the empirical support of the emerging theory is considered moderate.

Utility. The propositions of the emerging theory can be used in decision-making in industrial contexts such as IT support ticketing systems. For example, employing a traditional/linear ML model instead of a black-box model (given the comparable performance) implies better interpretability of the model predictions and relatively less use of computational resources.
Therefore, the utility of the emerging theory is considered high.

Testability. The domain or situation in which the theory should be confirmed or disconfirmed is clear. Furthermore, the propositions of the emerging theory are testable and empirical refutation is possible.
Therefore, the testability of the emerging theory is considered high.

7.4 Conclusion

In summary, this chapter reflects upon the issues found in Chapters (2,4, and 5) and postulates an emerging theory based on the observations from the six studies reported in earlier chapters of this thesis. The emerging theory was evaluated based on the criteria list from [Boehm and Jain, 2006] and [Sjøberg et al., 2008]. The emerging theory is assessed logically, based on a set of criteria from [Boehm and Jain, 2006] and [Sjøberg et al., 2008] yielding the following assessments: Generality – low to moderate; Parsimony – high; Explanatory power – low to moderate; Empirical support – moderate; Utility – high; and Testability - high. Further and wider experimentation over time would no doubt lead to more insightful results for these criteria and, in turn, improved decision-making in practice.

References

[Aronoff and Rees-Miller, 2020] Aronoff, M. and Rees-Miller, J. eds., 2020. The handbook of linguistics. John Wiley & Sons.

[Boehm and Jain, 2006] Boehm, B.W. and Jain, A., 2006. An initial theory of value-based software engineering. In Value-Based Software Engineering (pp. 15-37). Springer, Berlin, Heidelberg.

[Bojanowski et al., 2017] Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T., 2017. Enriching word vectors with subword information. Transactions of the association for computational linguistics, 5, pp.135-146.

[Sjøberg et al., 2008] Sjøberg, D.I., Dybå, T., Anda, B.C. and Hannay, J.E., 2008. Building theories in software engineering. In Guide to advanced empirical software engineering (pp. 312-336). Springer, London.

[Song et al., 2020] Song, C., Wang, X.K., Cheng, P.F., Wang, J.Q. and Li, L., 2020. SACPC: A framework based on probabilistic linguistic terms for short text sentiment analysis. Knowledge-Based Systems, 194, p.105572.

[Wahba et al., 2020] Wahba, Y., Madhavji, N.H. and Steinbacher, J., 2020, November. Evaluating the effectiveness of static word embeddings on the classification of IT support tickets. In Proceedings of the 30th Annual International Conference on Computer Science and Software Engineering (CASCON), (pp. 198-206).

[Wahba et al., 2022] Wahba, Y., Madhavji, N., Steinbacher, J., 2022. A Comparison of SVM against Pre-trained Language Models (PLMs) for Text Classification Tasks. In: International Conference on Machine Learning, Optimization, and Data Science. LOD 2022. Lecture Notes in Computer Science (LNCS), Springer, Cham (in press).

Chapter 8: Conclusion and Future Work

In this section, we present the conclusions and future work of this thesis. In Section 8.1, we revisit the research problem and the emerging theory from Chapter 7. Then, we discuss future work in Section 8.2.

8.1 Conclusions

In today's world, support ticketing systems are employed by a wide range of businesses. A support ticket describes an issue faced by the customer that is submitted as a bug report to the IT support team. Service agents spend a large amount of time manually classifying the incoming tickets. Unfortunately, this process is complicated, and the support agents have no reference to best practices based on historical data. Ticket classification is an important process that ensures that tickets get routed to the right support agent. Otherwise, there can be delays, customer dissatisfaction, escalation to management, and reactionary fixes at high costs [Sheng et al., 2014].

The task of text classification is challenging; due to the complexity of the unstructured nature of human language. Recently, pre-trained language models (PLMs) such as BERT [Devlin et al., 2018] and ELMO [Neumann et al., 2018] have shown promising results in several NLP tasks, including spam filtering, sentiment analysis, and question answering. In comparison to traditional models, PLMs require less feature engineering and minimal effort in data cleaning, thus becoming the consensus for many NLP tasks [Han et al, 2021].

Despite the widespread use of attention-based models (i.e., PLMs) and their impressive performance in a broad range of NLP tasks, there is a lack of a clear and well-justified need to as why these models are being employed for domain-specific text classification tasks [Chalkidis et al., 2020; Blinov et al., 2020; Zhao et al., 2021] given the linearly separable nature of most text classification tasks [Joachims, 1998; Tong and Koller, 2001]. Thus, the key research question is: Are PLMs the most cost-efficient solution for domain-specific TC tasks?

Using the gap in the literature and our experience with industry as our motivation, we propose a novel Hybrid Online Offline Model (HOOM) to classify hierarchical domain-

specific text with overlapping classes (Chapter 6). The model aims to satisfy the needs of the support agents in practice by providing them with an interpretable, high-accuracy, and less-expensive model that could be easily reproduced.

The contribution of this thesis is a combination of five empirical studies that were conducted over the last four years. Based on the observations from these studies, an emerging theory is proposed in Chapter 7 (Section 7.2). The emerging theory stimulates the use of traditional ML models over transformer-based DL models (i.e., PLMs) for solving domain-specific text classification tasks. The proposition of the emerging theory is based on the following reasons:

1. Most text classification problems are linearly separable [Joachims, 1998; Tong and Koller, 2001], thus, a traditional model such as linear SVM would perform well.
2. The gap between the way PLMs were trained (i.e., cloze-style) to predict target words as the objective and the downstream objectives (e.g., classification) limit the ability to exploit the knowledge encoded in PLMs [Han et al., 2021].
3. The degree of polysemy in domain-specific (i.e., specialized) text is low. This is because scientific terms need a precise meaning in order to function and be easily recognized [Wielgosz, 2017], thus, defeating the purpose of contextualized embeddings that aim to capture word polysemy and provide several embeddings for a single word.
4. Domain-specific words (i.e., OOV) are challenging for PLMs since these models are trained on generic corpora [Bollegala et al., 2015; Pilehvar and Collier, 2016].

8.2 Future Work

The opportunities for future work are centered in three directions:

- The integration (i.e., merge) of HiClass [Miranda et al., 2021] with the online River library [Montiel et al., 2021] to employ hierarchical classification for the online learning algorithms implemented in River.

- Automation of the drift detection module of HOOM. This entails integrating an automatic drift detector that triggers (i.e., signals) an alarm every time the model detects a drift in the incoming data stream.
- Investigating more evaluation metrics for the online learning model such as how fast the model learns and how much the model forgets [Mai et al., 2022].

A closing remark: we are now working with our IT industrial partner to deploy the proposed hybrid model (HOOM) into production.

References

[Blinov et al., 2020] Blinov, P., Avetisian, M., Kokh, V., Umerenkov, D. and Tuzhilin, A., 2020, August. Predicting clinical diagnosis from patients electronic health records using BERT-based neural networks. In International Conference on Artificial Intelligence in Medicine (pp. 111-121). Springer, Cham.

[Bollegala et al., 2015] Bollegala, D., Maehara, T., Yoshida, Y. and Kawarabayashi, K.I., 2015, February. Learning word representations from relational graphs. In Twenty-Ninth AAAI Conference on Artificial Intelligence (pp. 730–740).

[Chalkidis et al., 2020] Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N. and Androutsopoulos, I., 2020, November. LEGAL-BERT: The Muppets straight out of Law School. In Findings of the Association for Computational Linguistics: EMNLP 2020 (pp. 2898-2904).

[Devlin et al., 2018] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Minneapolis. pp.4171-4186.

[Han et al, 2021] Han, X., Zhang, Z., Ding, N., Gu, Y., Liu, X., Huo, Y., Qiu, J., Yao, Y., Zhang, A., Zhang, L. and Han, W., 2021. Pre-trained models: Past, present and future. AI Open, 2, pp.225-250.

[Joachims, 1998] Joachims, T., 1998, April. Text categorization with support vector machines: Learning with many relevant features. In European conference on machine learning (pp. 137-142). Springer, Berlin, Heidelberg.

[Mai et al., 2022] Mai, Z., Li, R., Jeong, J., Quispe, D., Kim, H. and Sanner, S., 2022. Online continual learning in image classification: An empirical survey. Neurocomputing, 469, pp.28-51.

[Miranda et al., 2021] Miranda, F.M., Köehnecke, N. and Renard, B.Y., 2021. HiClass: a Python library for local hierarchical classification compatible with scikit-learn. arXiv preprint arXiv:2112.06560.

[Montiel et al., 2021] Montiel, J., Halford, M., Mastelini, S.M., Bolmier, G., Sourty, R., Vaysse, R., Zouitine, A., Gomes, H.M., Read, J., Abdessalem, T. and Bifet, A., 2021. River: machine learning for streaming data in Python. Journal of Machine Learning Research (JMLR), 22(110), pp.1-8.

[Neumann et al., 2018] Neumann, M.P.M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L., 2018. Deep contextualized word representations. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. New Orleans. pp.2227–2237.

[Pilehvar and Collier, 2016] Pilehvar, M.T. and Collier, N., 2016, August. Improved Semantic Representation for Domain-Specific Entities. In Proceedings of the 15th Workshop on Biomedical Natural Language Processing. ACL (pp. 12-16).

[Sheng et al., 2014] Sheng, V.S., Gu, B., Fang, W. and Wu, J., 2014. Cost-sensitive learning for defect escalation. Knowledge-Based Systems, 66, pp.146-155.

[Tong and Koller, 2001] Tong, S. and Koller, D., 2001. Support vector machine active learning with applications to text classification. Journal of machine learning research, 2(Nov), pp.45-66.

[Wielgosz, 2017] Wielgosz, A.K., 2017. Meaning In Terms: A Monosemic Approach To The Lexical Semantics Of English And Japanese Terms Taken From Narrative Contexts. The Asian Conference on Arts & Humanities.

[Zhao et al., 2021] Zhao, Z., Zhang, Z. and Hopfgartner, F., 2021, April. A comparative study of using pre-trained language models for toxic comment classification. In Companion Proceedings of the Web Conference 2021 (pp. 500-507).

Curriculum Vitae

Name: Yasmen Wahba

Post-secondary Education and Degrees:

The University of Western Ontario
London, Ontario, Canada
Ph.D., Computer Science (2018–2022)

Suez Canal University– Faculty of Computers & Informatics
Ismailia, Egypt
M.Sc., Computer Science (2013–2016)

Suez Canal University– Faculty of Computers & Informatics
Ismailia, Egypt
B.Sc., Computer Science (2004–2008)

Honours and Awards:

Western Graduate Research Scholarship (WGRS)
2018–2022

University of Western Ontario Research in Computer Science Conference (UWORCS) Second prize award – 2019

University of Western Ontario Research in Computer Science Conference (UWORCS) Second prize award – 2021

Related Work Experience:

Teaching Assistant
The University of Western Ontario – London, Ontario, Canada
2018–2022

Teaching Assistant
Suez Canal University – Ismailia, Egypt
2009–2016

Web Programmer (Freelance)
Bravo Serve Web Design Company– Cairo, Egypt
2008–2011

Publications:

1. Wahba, Y., Madhavji, N.H. and Steinbacher, J., 2020, November. Evaluating the effectiveness of static word embeddings on the classification of IT support tickets. In Proceedings of the 30th Annual International Conference on Computer Science and Software Engineering (CASCON) (pp. 198-206).

2. Wahba, Y., Madhavji, N. and Steinbacher, J., 2022, March. Reducing Misclassification Due to Overlapping Classes in Text Classification via Stacking Classifiers on Different Feature Subsets. In Future of Information and Communication Conference (FICC) (pp. 406-419). Springer, Cham.

3. Wahba, Y., Madhavji, N. and Steinbacher, J., 2022. A Comparison of SVM against Pre-trained Language Models (PLMs) for Text Classification Tasks. In 8th International Conference on Machine Learning, Optimization, and Data Science (LOD 2022), Lecture Notes in Computer Science (LNCS), (pp. 304-313). Cham: Springer Nature Switzerland.

4. Wahba, Y., Madhavji, N. and Steinbacher, J., 2022. A Hybrid Machine Learning Model for Efficient Classification of IT Support Tickets in The Presence of Class Overlap. In Proceedings of the 32nd Annual International Conference on Computer Science and Software Engineering (CASCON22) (pp. 151–156).

5. Wahba, Y., Madhavji, N. and Steinbacher, J., 2023. Attention is Not Always What You Need: Towards Efficient Classification of Domain-Specific Text. In Proceedings of the 2023 Computing Conference, Springer, Cham (in press).

6. Wahba, Y., Madhavji, N. and Steinbacher, J., 2023. A Hybrid Continual Learning Approach for Efficient Hierarchical Classification of IT Support Tickets in A Real-World Scenario. Accepted in: 24th IEEE International Conference on Industrial Technology (ICIT).