Interpretable Architectures and Algorithms for Natural Language Processing

Rohan Kumar Yadav

UiA University of Agder

Doctoral Dissertations at the University of Agder 388

Rohan Kumar Yadav

Interpretable Architectures and Algorithms for Natural Language Processing

Doctoral Dissertation for the Degree *Philosophiae Doctor (PhD)* at
the Faculty of Engineering and Science, Specialisation in Artificial Intelligence

University of Agder
Faculty of Engineering and Science
2022

Doctoral Dissertations at the University of Agder 388
ISSN: 1504-9272
ISBN: 978-82-8427-101-9

Printed by 07 Media
Oslo

Preface

The foundation of this research was initially triggered when I first started to work on machine learning models during my master's degree. I then decided to move forward my research along machine learning. Due to immense amount of data available and future scope in the field of language, I decided to pursue my PhD in the field of Natural Language Processing. This dissertation is a result of the research work carried out at the Department of Information and Communication Technology (ICT), University of Agder (UiA), Grimstad, Norway, from September 2019 to June 2022. During my Ph.D. study, my main supervisor has been Associate Professor Lei Jiao, UiA, and my co-supervisor has been Professor Morten Goodwin, UiA, Norway.

Acknowledgments

First of all, I would like to express my intense gratitude to my supervisor, Associate Professor Lei Jiao. Without his assistance and guidance, this endeavor would not have been possible. His encouragement, enthusiasm, and vision always motivated me to do fruitful research. I am deeply indebted to him for his meticulous reviews of my manuscripts. His comments and feedback always helped to ensure the manuscripts with high quality. His contribution in my personal and professional development is enormous. During the ups and downs of the journey, he is the only person who always came with a solution and showed the next logical step.

I am grateful to my co-supervisor Professor Morten Goodwin for his support and guidance in my research. His insightful comments and feedback always helped to improve the quality of the manuscripts. Morten has been and still is one of my mentors in Deep Learning and Natural Language Processing. He is an expert in AI, whom I relied on during my time as a PhD student. Both my supervisors have not only played the role of advisors but also have been guardians.

I would also thank Ole-Christoffer Granmo who is the head of Centre for Artificial Intelligence Research (CAIR) for constantly supporting me with various aspects of NLP and Tsetlin Machine. He has been one of the important guidance during my research journey. His help and support with Tsetlin Machine has been immense that lead me to have high class papers. I wish to thank the Ph.D. coordinators at the Faculty of Engineering and Science, UiA, Kristine Evensen Reinfjord and Emma Elisabeth Horneman for their administrative support. I also would like to thank all the professors at the Department of ICT who always encouraged and motivated me.

I am also grateful to my teammates in CAIR for their feedback, help, and encouragements through these years. I really enjoyed the technical as well as non-technical discussions that we had in the CAIR group. Many thanks to Bimal Bhattarai, Rupsa Saha, Ahmed Abdulrahem Abouzeid, Svein Anders Tunheim, Saeed Gorji, Jivitesh Sharma, and Ayan Chatterjee for their support throughout the Ph.D. journey.

Although far away from my home country, Nepal, where my family lives, the love, encouragement, and support that I receive from my parents, Rudra Prasad Yadav, Shila Yadav, and my sister Sneha Yadav have been immense. Taking this opportunity, I express my eternal gratitude to them for their blessings, love, understanding, and support. Their confidence in me is the source that leads me to every tiny success in my life. I also take great pride in thanking my wife, Barkha Sah, who always supported, motivated, and encouraged me in this endeavor.

Dedicated to my parents

Rudra Prasad Yadav and Shila Yadav

And my loving wife

Barkha Sah

Abstract

Natural Language Processing (NLP) is one of the branches of Artificial Intelligence (AI) that teaches computers to understand, process, and generate language. Recently, deep neural networks-based (DNNs) NLP have gained huge popularity for their ability to accomplish various tasks in NLP, such as text classification, sentiment analysis, information retrieval, machine translation, and reading comprehension. Some transformer-based models such as BERT and GPTs are fine examples that achieve the state-of-the-art performance in many NLP tasks. However, these models being so huge are very difficult for humans to understand the underlying concept of the model, making them arguably BlackBox in nature. More specifically, for the downstream application of NLP, these models are far from perfect that can provide any logical explanation of the model. Moreover, the necessity of explanability in NLP comes into the picture when sensitive information needs to be evaluated out of the model. Such necessity of explainable NLP escalates mostly in the education domain, legal document analysis, and medical diagnosis using Electronic Health Records (EHR). As we know, these tasks are mostly solved based on accuracy by the big models in NLP and there is a big problem of lack of explainability. Even though recent attempt of explaining attention weights seems hopeful, it is merely a mathematical weight over input rationales rather than logical explanation of the model. Hence, in this thesis, we have designed various interpretable architectures and algorithms that are useful for the application of explainable NLP. Our proposed algorithms generally apply to most of the NLP tasks that require human-level explanation. The primary focus of this thesis is to enhance the state of the art not only in terms of accuracy but also for the transparency and explainability of NLP models, which have been addressed on a limited scale before.

This thesis has two parts: Firstly, we introduce the human level-interpretable models using Tsetlin Machine (TM) for NLP tasks. Secondly, we present an interpretable model using DNNs. The first part combines several architectures of various NLP tasks using TM along with its robustness. We use this model to propose logic-based text classification. We start with basic Word Sense Disambiguation (WSD), where we employ TM to design novel interpretation techniques using the frequency of words in the clause. We then tackle a new problem in NLP, i.e., aspect-based text classification using a novel feature engineering for TM. Since TM operates on Boolean features, it relies on Bag-of-Words (BOW), making it difficult to use pre-trained word embedding like Glove, word2vec, and fasttext. Hence, we designed a Glove embedded TM to significantly enhance the model's performance. In addition to this, NLP models are sensitive to distribution bias because of spurious correlations. Hence we employ TM to design a robust text classification against spurious correlations.

The second part of the thesis consists interpretable model using DNN where we design a simple solution for complex position dependent NLP task. Since TM's interpretability comes

with the cost of performance, we propose an DNN-based architecture using a masking scheme on LSTM/GRU based models that ease the interpretation for humans using the attention mechanism. At last, we take the advantages of both models and design an ensemble model by integrating TM's interpretable information into DNN for better visualization of attention weights.

Our proposed model can be efficiently integrated to have a fully explainable model for NLP that assists trustable AI. Overall, our model shows excellent results and interpretation in several open-sourced NLP datasets. Thus, we believe that by combining the novel interpretation of TM, the masking technique in the neural network, and the integrated ensemble model, we can build a simple yet effective platform for explainable NLP applications wherever necessary.

Sammendrag

Natural Language Processing (NLP) er en av grenene innen kunstig intelligens (AI) som lærer datamaskiner å forstå, behandle og generere språk. Nylig har løsninger basert på dype nevrale nettverk (DNN) fått stor popularitet pga. deres evne til å utføre ulike oppgaver i NLP, som tekstklassifisering, sentimentanalyse, informasjonshenting, maskinoversettelse og leseforståelse. Transformatorbaserte modeller som BERT og GPT er gode eksempler på dette og oppnår topp ytelse i mange NLP-oppgaver. Imidlertid er disse modellene så enorme at det er veldig vanskelig for mennesker å forstå det underliggende konseptet. Dette gir dem en BlackBox natur. Mer spesifikt, for nedstrømsapplikasjonen av NLP, er disse modellene langt fra perfekte når det gjelder å gi en logisk forklaring på modellen. Dessuten kommer nødvendigheten av forklarbarhet i NLP inn i bildet når sensitiv informasjon skal vurderes ut av modellen. Forklarlig NLP er særlig påkrevd innen utdanningsdomenet, juridisk dokumentanalyse og medisinsk diagnose ved bruk av elektroniske helsejournaler (EPJ). Som vi vet løses disse oppgavene for det meste basert på nøyaktigheten av de store modellene i NLP, og det er et stort problem med manglende forklarbarhet. Selv om nylige forsøk på å forklare via oppmerksomhetsvekter virker lovende, er dette bare en matematisk vekting over input-rasjonale snarere enn en logisk forklaring av modellen. Derfor har vi i denne oppgaven designet ulike tolkbare arkitekturer og algoritmer som er nyttige for bruk av forklarbar NLP. Våre foreslåtte algoritmer gjelder generelt for de fleste NLP-oppgavene som krever forklaring på menneskelig nivå. Hovedfokuset i denne oppgaven er å forbedre oppnåelig ytelse, ikke bare når det gjelder nøyaktighet, men også når det gjelder åpenhet og forklarbarhet av NLP-modeller. Dette har vært behandlet bare i begrenset skala tidligere.

Denne oppgaven har to deler: For det første introduserer vi menneske-tolkbare modeller ved å bruke Tsetlin Machine (TM) for NLP-oppgaver. For det andre presenterer vi en tolkbar modell ved bruk av DNN. Den første delen kombinerer flere arkitekturer for ulike NLP-oppgaver ved bruk av TM sammen med denne modellens robusthet. Vi bruker denne modellen til å foreslå logikkbasert tekstklassifisering. Vi starter med grunnleggende Word Sense Disambiguation (WSD), der vi bruker TM til å designe nye tolkningsteknikker ved å bruke hyppigheten til ordene i setningen. Vi takler deretter aspektbasert tekstklassifisering som er et nytt problem i NLP, ved å bruke en ny funksjonsteknikk for TM. Siden TM opererer på boolske funksjoner, er den avhengig av Bag-of-Words (BOW), noe som gjør det vanskelig å bruke forhåndstrente ordinnbyggingsmodeller som Glove, word2vec og fasttext. Derfor designet vi en Glove embedded TM for å forbedre modellens ytelse betydelig. I tillegg til dette er NLP-modeller følsomme for distribusjonsskjevhet på grunn av falske korrelasjoner. Derfor bruker vi TM for å designe en robust tekstklassifisering mot falske korrelasjoner.

Den andre delen av oppgaven består av en tolkbar modell ved bruk av DNN hvor vi designer en enkel løsning for komplekse posisjonsavhengige NLP-oppgaver. Siden TMs tolkbarhet kommer med kostnadene når det gjelder ytelse, foreslår vi en DNN-basert arkitektur ved å bruke et maskeringsskjema på LSTM/GRU-baserte modeller som letter tolkningen for mennesker ved å bruke oppmerksomhetsmekanismen. Endelig tar vi fordelene med begge modellene og designer en ensemblemodell ved å integrere TMs tolkbare informasjon i DNN for bedre visualisering av oppmerksomhetsvekter.

Vår foreslåtte modell kan integreres effektivt til en fullstendig forklarbar modell for NLP

som bidrar til pålitelig AI. Samlet sett viser modellen vår utmerkede resultater og tolkning i flere åpne NLP-datasett. Dermed tror vi at ved å kombinere den nye tolkningen av TM, maskeringsteknikken i det nevrale nettverket og den integrerte ensemblemodellen, kan vi bygge en enkel, men effektiv plattform for forklarbare NLP-applikasjoner der det er nødvendig.

Publications

The author of this dissertation is the first author and the principal contributor of all the included papers listed below. Papers A-F in the first set of the following list are selected to represent the main research achievements and are reproduced as Part II of this dissertation. Other papers which are listed in the second set are some other contributions towards Artificial Intelligence Research.

Papers Included in the Dissertation

Paper A **Yadav, R. K.**, Jiao, L., Granmo, O., and Goodwin, M., "Interpretability in Word Sense Disambiguation using Tsetlin Machine.", In *International Conference on Agents and Artificial Intelligence (ICAART)*, pp. 402-409. SciTePress, 2021.

Paper B **Yadav, R. K.**, Jiao, L., Granmo, O., and Goodwin, M., "Human-Level Interpretable Learning for Aspect-Based Sentiment Analysis.", In *35th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 14203-14212., 2021.

Paper C **Yadav, R. K.**, Jiao, L., Goodwin, M., and Granmo, O., "Positionless aspect based sentiment analysis using attention mechanism.", *Knowledge-Based Systems*, Volume 226, Elsevier, May, 2021.

Paper D **Yadav, R. K.**, Jiao, L., Granmo, O., and Goodwin, M., "Enhancing Interpretable Clauses Semantically using Pretrained Word Representation.", In *BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 265–274. ACL, 2021.

Paper E **Yadav, R. K.**, Jiao, L., Granmo, O., and Goodwin, M., "Robust Interpretable Text Classification against Spurious Correlations Using AND-rules with Negation.", In *International Joint Conference on Artificial Intelligence (IJCAI), 2022*.

Paper F **Yadav, R. K.**, Nicolae, D.C., "Enhancing Attention's Explanation Using Interpretable Tsetlin Machine.", *Algorithms*, 15, no. 5: 143, MDPI, 2022.

Other Publications

1. Yadav, R. K., Bhattarai, B., Jiao, L., Granmo, O., and Goodwin, M., “Indoor Space Classification Using Cascaded LSTM.”, In *IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 1110-1114. IEEE, 2020.

2. Abeyrathna, K.D., Bhattarai, B., Goodwin, M., Gorji, S.R., Granmo, O., Jiao, L., Saha, R., Yadav, R. K., “Massively Parallel and Asynchronous Tsetlin Machine Architecture Supporting Almost Constant-Time Scaling.”, In *International Conference on Machine Learning (ICML)*, PMLR, 2021.

3. Tunheim, S., Yadav, R. K., Jiao, L., Shafik, R., Granmo, O., “Cyclostationary Random Number Sequences for the Tsetlin Machine.”, In *IEA-AIE*, 2022.

4. Sharma, J., Yadav, R. K., Granmo, O., Jiao, L., “Drop Clause: Enhancing Performance, Interpretability and Robustness of the Tsetlin Machine.”, In *Arxiv*, 2021.

5. Nicolae, D.C., Yadav, R. K., Tufiş, D, “A Lite Romanian BERT: ALR-BERT.”, *Computers*, MDPI, 2022.

Contents

List of Figures

List of Tables

Part I

Main Chapters

Chapter 1

Introduction

Machine learning (ML) has recently obtained a lot of press for its ability to properly predict a wide range of complicated events. However, there is a growing recognition that, in addition to predictions, machine learning models can produce knowledge about domain relationships in data, which is referred to as "interpretations". In the absence of a well-defined concept of interpretability, a wide range of approaches and outputs (e.g., visualizations, natural language, mathematical equations) have been branded as interpretation. This has caused a great deal of misunderstanding about the concept of interpretability. It is not apparent what it means to interpret something, what common threads exist in various ways, or how to choose an interpretation method for a certain problem or audience. The term "interpretability" is a wide and ill-defined term. To interpret data in its broadest sense means to extract information (in a certain form) from it. Hence, there have been several studies looking into various parts of interpretation, which are frequently referred to as explainable AI.

Natural language processing (NLP) is one of the most fascinating domains in the application of ML. NLP deals with extracting the structure and meaning of the text, analyzing text, and extracting information about people, places, and events to better understand social media sentiment and customer conversations. Most of the NLP models have been substantially improved since the introduction of deep learning. As a result, the models are becoming more sophisticated, making the reasoning behind their predictions more difficult to comprehend. To deploy deep neural networks (DNNs) for generating high-stakes choices, interpretability must be ensured for the public to have faith in them. For example, users are unlikely to be persuaded of the decision made by the model with little to no explanation for the tasks such as legal document analysis, medical diagnosis, and education domain. As a result, an NLP system's capacity to explain the logic is critical. Many studies have been published in recent years to handle text categorization difficulties, but only a few have looked at the explainability of their systems. Some researchers employ a heatmap to see how much each concealed feature affects the expected outcomes. Although these systems are promising, they generally ignore fine-grained data that might be useful in evaluating model behavior. In order to design a transparent model based on ML algorithms, one has to be able to explain the operational concept of the model and build a trustworthy platform. Recently, due to the introduction of an interpretable model called Tsetlin Machine (TM) [1], it has opened a vast door in the area of explainable ML. Hence, to have a trustable NLP model, we here aim to establish interpretable architectures and algorithms by maintaining a trade-off in performance/interpretability continuum.

In this chapter, the motivation of this Ph.D. dissertation is discussed along with an overview of the research questions. Furthermore, the goals and approaches are explained, and the structure of the dissertation is also outlined.

1.1 Motivation and Research Questions

As the industry quickly embraces ML technology, the interpretation of ML models has attracted significant interest and has become important in practice. The model interpretation clarifies how models make decisions, which is especially important in mission-critical sectors where the decision-making process must be transparent and accountable. Recently, the interpretation of DNN-based models have relied heavily on attention weights. Attention-based models are employed in explaining arbitrary BlackBox models' predictions, by picking a set of input components to approximate the predictions, which are typically utilized to improve interpretability while simultaneously boosting performance. Recent research, on the other hand, suggests that the feature selection emphasized by attention processes may not always correspond with an intuitively larger emphasis on final predictions. For example, a surprising portion of BERT's attention is focused on insignificant tokens like "[SEP]", ",", and "." [2]. Furthermore, several studies have been conducted to support or contradict the interpretability of attention processes [3, 4]. There has been a lot of discussion over whether or not attention is an explanation.

There are several traditional methods such as Decision Trees, Random Forest, Support Vector Machine (SVM), and Logistic Regression that are arguably considered the interpretable models. However, most of them suffer from low accuracy. Though Logistic Regression has shown promising performance in various NLP tasks, it is difficult for a human to interpret the prediction made by it. One has to analyze mathematical weights to understand the model's prediction. In addition, bigger models based on DNN such as LSTM, CNN, and transformers are very complex to analyze and have a BlackBox nature that prevents humans to understand them. It has created a huge concern in the field of NLP and computer vision for explainable and interpretable Artificial Intelligence (AI). One of the recent advancements in explainable ML is TM which is an interpretable and transparent model that learns patterns based on propositional formula [1]. This model is less explored, but seems very promising in the field of human-level interpretation.

In this thesis, we introduce interpretable models using both TM and DNNs-based models, enlisting the research questions and answering them. By doing so, we aim to advance the state-of-the-art for interpretable ML models on a variety of NLP tasks.

Research Question 1: How can we design interpretable NLP models using Tsetlin Machine?

Motivation: Word Sense Disambiguation (WSD) is one of the branches of text classification where the sense of a particular word has to be classified. This is one of the benchmark tasks for analyzing interpretable models, where the context is very important. Hence, we design an interpretable model using TM and evaluate whether it captures the true context that is understandable for humans. Here we introduce a novel way of interpreting the TM architecture using the frequency of literals included in the clause. In addition, we explore another NLP task

different from traditional text classification known as position-dependent text classification. We adopt the popular evaluation framework called Aspect based Sentiment Analysis (ABSA) dataset.

Approach: To design a human-interpretable WSD model, we train TM on the WSDCoarsed Balanced dataset for various words and extract the information of clauses. The clauses in TM hold the information of features in their original and negated forms. We will utilize a frequency-based feature selection technique to classify important and non-important words in clauses. For the ABSA task, to encode position-dependent information into TM so that it does not lose human-level interpretability, we extract novel position-dependent information by splitting the sentence into two parts from the target word. We also encode the SentiWordNet information represented by 3 bits to give the model extra information. This information is then fed to TM so that the clause of TM can be employed to trace back the information of each word and its position for easier human-level interpretation.

Research Question 2: How can TM capitalize the pretrained word representation to enhance accuracy and interpretability of the model?

Motivation: Legacy TM adopts Boolean input features like bag-of-words (BOW) to provide human-level interpretability. The BOW representation makes it difficult to employ any pre-trained data, such as word2vec and GloVe word representations. This limitation becomes the bottleneck for the performance of TM in NLP when compared with DNNs.

Approach: To obtain a better performing TM model with a more general interpretation, we extract additional features using GloVe representation based on cosine similarity. We accomplish this by feeding TM with semantically related words extracted from pre-trained word representations. This preprocessing adds additional semantic information to the model, and the clauses include more information than the legacy TM, which also interprets better semantically apart from enhancing the accuracy.

Research Question 3: How can we tackle the problem of spurious correlation in text classification?

Motivation: DNNs-based NLP models have achieved state-of-the-art performance because of their capability to learn the non-linearity in the data. However, the main concern lies in the ability to learn genuine correlations in the data. Due to distributional bias in the training data, these models fail to perform better when the distribution shifts in the test data or the test data are out-of-distribution. Since such models are BlackBox in nature, we cannot force them not to learn spurious correlations and usually rely on counterfactual data augmentation. However, this is an expensive and time-consuming process. To solve this problem, we employ TM to design a text classification model that is robust to spurious correlation.

Approach: Since TM is an interpretable model whose learning is transparent and explainable, we have the benefit of observing how the model learns based on the features. In more detail, we can study the functions in TM that are responsible for learning spurious correla-

tions and then design it not to learn those features. The hyper-parameter s, i.e., specificity, is one of the contributors of TM that decides the probability of features taking part in classification. Thus, we fine-tune this parameter so that it includes only genuine correlation in the model.

Research Question 4: Can we enhance the performance and explanation of ABSA task using DNN by discarding of positional embedding?

Motivation: The interpretability of the non-traditional text classification ABSA comes at the cost of performance in TM. These tasks heavily rely on positional embedding, which creates ambiguity in interpreting the attention weights. To have a better performing model as well as retaining a certain level of interpretation, we use a traditional Gated Recurrent Unit (GRU)-based model with an attention layer and design a novel feature representation technique for an easier and better explanation of the model.

Approach: To obtain a better-performing model with decent interpretation, we adopt two Bi-GRU-based models with an attention layer on top for the visualization of weight for the prediction. We then design a novel feature representation called the masking technique that significantly improves the performance of ABSA and the visualization of attention weight. The masking technique is based on duplicating the input sentence and masking the target (aspect) word while keeping the original sentence as it is. Both input samples are fed to the respective Bi-GRU, where the attention layer captures the significant information from each representation layer.

Research Question 5: How can we integrate the information in TM and DNN together for better performance and interpretability?

Motivation: The state-of-the-art NLP models are highly dominated by DNN-based models such as Long-short term memory (LSTM)/GRUs and transformers. However, their BlackBox nature makes the interpretation ambiguous. On the other hand, TM offers an easy logic-based interpretation of the model, but it comes with a significant loss in performance. The trade-off between accuracy and interpretability is one of the main concerns of modern NLP. There has been an immense attempt at extracting a logical explanation from the attention layer of DNN. However, the change in attention weights based on various scenarios makes it arguably tough to establish a trustworthy model. Hence to mitigate the limitation of both models, we utilize the interpretable information from TM and integrate it into DNN so that model has prerequisite information on the distribution of the task, generating sensible attention weights.

Approach: In order to integrate both models, we first train a TM on a particular dataset and obtain the distribution of the task based on the clause score of each word. It is then used as the prerequisite information in DNN's initial layer. This helps to pre-weight the input layer so that the attention layer can easily concentrate on intended input rationales, thereby assigning more weight to important words. We then extract these weights for a better understanding of the model.

Limitations: In this thesis, the main aim is to develop interpretable algorithms using TM and

DNN without significantly sacrificing performance. We tried to fill the gap between performance and interpretability by a significant margin so that it helps in downstream applications of NLP. The main focus of this thesis lies in addressing the issue of the interpretation of the model for text classification rather than more sophisticated tasks such as question answering, summarization, and information retrieval. The methods proposed in the thesis have not been tested in the real world, and thus we do not claim to have applied them already to the real-world examples.

Overall, the whole system is disjointed and versatile and we believe that it can be easily adapted for real-world interpretable text classification. Furthermore, information exchange between distinct (disjoint) modules is simple and may be developed as required. Hence, we propose and employ novel interpretable architectures and algorithms for NLP that bridge the gap between accuracy and interpretability to set a new paradigm of NLP methods.

1.2 Publications

We solve each task of the thesis using TM and DNNs. We employ various areas of ML to solve the problems of interpretability in sentiment analysis, text classification, and their robustness. We list the contributions of this thesis below, each of which is described in detail in Chapter III, and the associated papers published are presented in Part II of the thesis. Here, we present a summary of our papers:

Paper A We understand that DNN-based NLP models do not provide a faithful explanation of the model due to its BlackBox nature. A weakness of the state-of-the-art supervised models, however, is that it can be difficult to interpret them, making it harder to check if they capture senses accurately or not. In this paper, we introduce a novel TM-based supervised model that distinguishes word senses by means of conjunctive clauses. The clauses are formulated based on contextual cues, represented in propositional logic. Our experiments on CoarseWSD-balanced dataset indicate that the learned word senses can be relatively effortlessly interpreted by analyzing the converged model of the TM. Additionally, the classification accuracy is higher than that of FastText-Base and similar to that of FastTextCommonCrawl. This paper addresses Research Question 1.

Paper B In this paper, we propose a human-interpretable learning approach for ABSA task, employing the recently introduced TMs. We attain interpretability by converting the intricate position-dependent textual semantics into binary form, mapping all the features into BOWs. The binary-form BOWs are encoded so that the information on the aspect and context words are retained for sentiment classification. We further adopt the BOWs as input to the TM, enabling learning of aspect-based sentiment patterns in propositional logic. To evaluate interpretability and accuracy, we conducted experiments on two widely used ABSA datasets from SemEval 2014: Restaurant 14 and Laptop 14. The experiments show how each relevant feature takes part in conjunctive clauses that contain the context information for the corresponding aspect word, demonstrating human-level interpretability. At the same time, the obtained accuracy is on par with existing DNN models, reaching 78.02% on Restaurant 14 and 73.51% on Laptop 14. This paper addresses Research Question 1.

Paper C We observed that several existing articles demonstrated promising ABSA accuracy using positional embedding to show the relationship between an aspect word and its context. In most cases, the positional embedding depends on the distance between the aspect word and the remaining words in the context, known as the position index sequence. However, these techniques usually employ both complex preprocessing approaches with additional trainable positional embedding and complex architectures to obtain state-of-the-art performance. In this paper, we simplify preprocessing by including polarity lexicon replacement and masking techniques that carry the information of the aspect word's position and eliminate the positional embedding. We then adopt a novel and concise architecture using two bidirectional GRU along with an attention layer to classify the aspect based on its context words. Experiment results show that the simplified preprocessing and the concise architecture significantly improve the accuracy of the publicly available ABSA datasets, obtaining 81.37%, 75.39%, 80.88%, and 89.30% in restaurant 14, laptop 14, restaurant 15, and restaurant 16 respectively. This paper addresses Research Question 4.

Paper D TM is an interpretable pattern recognition algorithm based on propositional logic that has demonstrated competitive performance in many NLP tasks, including sentiment analysis, text classification, and word sense disambiguation. To obtain human-level interpretability, legacy TM employs Boolean input features such as BOW. However, the BOW representation makes it difficult to use any pre-trained information, for instance, word2vec and GloVe word representations. This restriction has constrained the performance of TM compared with DNNs in NLP. To reduce the performance gap, in this paper, we propose a novel way of using pre-trained word representations for TM. The approach significantly enhances the performance and interpretability of TM. We achieve this by extracting semantically related words from pre-trained word representations as input features to the TM. Our experiments show that the accuracy of the proposed approach is significantly higher than the previous BOW-based TM, reaching the level of DNN-based models. This paper addresses Research Question 2.

Paper E We observed that distribution biases in the training and testing data greatly impact the performance of the models when exposed to out-of-distribution and counterfactual data. The root cause seems to be that many machine learning models are prone to learning shortcuts, modeling simple correlations rather than more fundamental and general relationships. As a result, such text classifiers tend to perform poorly when a human makes minor modifications to the data, which raises questions regarding their robustness. In this paper, we employ a rule-based architecture called TM that learns both simple and complex correlations by ANDing features and their negations. As such, it generates explainable AND-rules using negated and non-negated reasoning. Here, we explore how non-negated reasoning can be more prone to distribution biases than negated reasoning. We further leverage this finding by adapting the TM architecture to mainly perform negated reasoning using the specificity hyper-parameter s. As a result, the AND-rules become robust to spurious correlations and can also correctly predict counterfactual data. Our empirical investigation of the model's robustness uses the specificity s to control the degree of negated reasoning. Experiments on publicly available Counterfactually-Augmented Data demon-

strate that the negated clauses are robust to spurious correlations and outperform Naive Bayes, SVM, and Bi-LSTM by up to 20%, and ELMo by almost 6% on counterfactual test data. This paper addresses Research Question 3.

Paper F Attention mechanism has been a popular choice for such explainability recently by estimating the relative importance of input units. We noticed that such processes tend to misidentify irrelevant input units when explaining them. This is due to the fact that language representation layers are initialized by pre-trained word embedding that is not context-dependent. Such a lack of context-dependent knowledge in the initial layer makes it difficult for the model to concentrate on the important aspects of input. Usually, this does not impact the performance of the model, but its explainability differs from human understanding. Hence, in this paper, we propose an ensemble method to use logic-based information from the TM to embed it into the initial representation layer in the DNN to enhance the model in terms of explainability. We obtain the global clause score for each word in the vocabulary and feed it into the DNN layer as context-dependent information. Our experiments show that the ensemble method enhances the explainability of the attention layer without sacrificing any performance of the model and even outperforming it for certain datasets. This paper addresses Research Question 5.

Each of the articles listed above tries to solve a task in NLP in an interpretable way. A pictorial representation of the flow of the content of our contributions in accordance with the aim of this thesis is shown in Figure 1.1. Our papers address the problems in the two phases of interpretable algorithms using TM, which together give an overview of interpretable text classification, enhancing performance, and robustness against counterfactual samples. As can be seen from Figure 1.1, the modules in each phase are interconnected to each other, but also act as disjoint modules to provide separate information to the interpretable NLP. The first four boxes show the type of task in NLP using TM and the circles denote the models associated with each task. There are two models: TM and DNNs. Eventually, both are integrated for better interpretability of the NLP model.

1.3 Thesis Outline

The dissertation is organized into two parts. Part I contains an overview of the work carried out throughout this Ph.D. study and Part II includes a collection of six published or accepted papers, which are mentioned in the list of publications. In addition to the introduction chapter presented above, the following chapters are included.

- Chapter II presents the background and preliminary information of various techniques and methods used in this thesis such as Explainable ML, TM, and DNNs.
- Chapter III explains the contributions of this thesis in detail. It is divided into two sections that present the two stages of the interpretable NLP model. Each section describes our contributions to both stages, i.e., Interpretable Text Classification using TM, and Interpretable Text Classification using DNNs. The motivation, intuition, novelty, methodology, and results for each contribution are explained extensively.

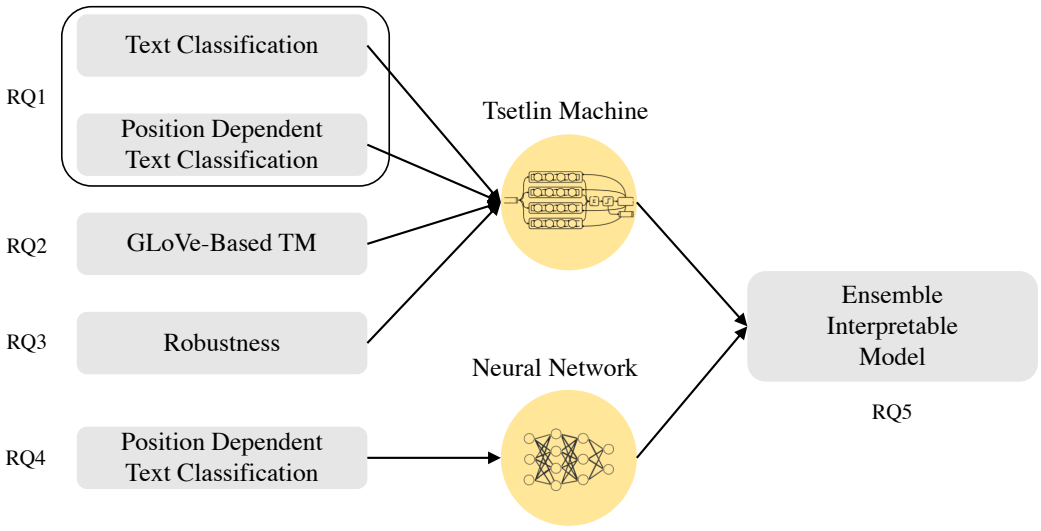


Figure 1.1: Organization of Contributions

- Chapter IV concludes the thesis and discusses the implications of the outcomes of the thesis. It also contains potential future research directions that can further improve the work presented in the thesis. This chapter also concludes Part I of the thesis.
- In Part II of the thesis, all publications associated with the thesis are presented entirety. There are six publications labeled as Paper A to F. The papers are listed in chronological order with the flow of the thesis and in order of published timeline.

Chapter 2

Background

In this chapter, we briefly describe the background and preliminary information needed to understand the thesis. Specifically, we explain concepts that have been used throughout this thesis. First, machine learning has been briefly introduced. Next, we move on to the TMs that have been extensively employed in this thesis and are a recurring theme in the papers associated with this thesis. Neural network has been an established model for NLP and has has also been explained here in this chapter. In addition, we present various NLP models that rely on neural networks such as RNN, LSTM, Attention, and Transformers.

2.1 Interpretable Machine Learning

The term "interpretability" is broad, and there are numerous definitions in the literature, some of which are borrowed from mathematical logic [5]. We could partially define it as a "relationship" between formal theories that expresses the potential of interpreting or translating one into the other. This concept clearly extends interpretability to various ML applications. Based on Cain's definition [6], mathematical models are not perfect, but mathematical representations of natural phenomena are highly reliable. The interpretability of model is a fundamental aspect of assessing how decisions are made as well as assessing the subsequent predictions of models. Although a machine learning model may make predictions with great precision and accuracy, what matters most to a decision-maker is how the choice was made, and therefore how the prediction was made or how a specific instance was categorized by the algorithm. Doshi-Velez et. al [7] answers this question by stating that "accuracy being an important metric is still an incomplete description of the model in most of the real-life tasks". There are certainly occasions in which it is vital to explain why one choice is superior to another, for instance, ML applications in the legal and medical domains.

Based on Tjoa and Guana [8], there are various types of interpretability:

1. **Perceptual interpretability:** This category considers human perception as interpretation. One of the subcategories is Saliency. Saliency is the method that assigns a mathematical value to the input components that reflects their importance and their contribution to the decision. Another subcategory includes the signal method, which examines the stimulation of the neurons or a set of neurons. Such activated values of neurons are one way to transform them into interpretable form. Verbal interpretability, on the other hand, is

another form of perceptual interpretability where it is assumed that some verbal structures are easily understandable to humans. The right concatenation of predicates and connectives can result in logical statements.

2. **Interpretability through mathematical structures:** The premise behind the study of predefined complex systems is that a parametric mathematical model can aid the explanation of the phenomena. For example, in a linear regression model, if the basic hypotheses are respected and the Ordinary Least Square estimates are consistent, the interpretation of the parameters is certainly clearer, and the model can be improved by adding more complex components. The idea of trade-off is the foundation of the reasoning that leads to the model's explainability in increasingly complicated models such as neural networks. In addition, Feature extraction is one of the most well-known techniques in the literature. It is a correlations-and-associations-based approach that allows one to distinguish which features are less important than others or to find internal patterns that can help explain and translate the models' complex components. Sensitivity is another basis of interpretation based on gradient analysis, perturbation, and localization. They are thus considered infinitesimal neighborhoods of successive points by analyzing how the function changes in a neighborhood, which is the concept behind the gradient analysis.

The interpretable model can be classified into two categories: Transparent Model and Post-Hoc Interpretability. Transparent ML models are transparent and do not need additional analysis to extract the explanation of the model. However, post-hoc interpretable models are neural networks that require extensive post-processing to obtain the desired interpretability. Here we have carefully selected the relatable, interpretable models that closely align with the field of NLP. The selection is also based on the comparison we have made with our proposed tasks in the thesis.

2.1.1 Linear Regression

Linear Regression model predicts the dependent variable based on the weighted sum of the independent variables also known as covariates [9]. The relationship's linearity makes it very easy to comprehend. These models are used to model a y (target) variable's dependency on certain x variables [10]. The data are supposed to be impacted by noise (of the Gaussian kind, with a mean μ and variance of σ), and the relationship may be described as follows:

$$y = \beta_0 + \beta_1 x_1 + \cdots \beta_p x_p + \epsilon, \tag{2.1}$$

where ϵ is the error that is the difference between actual and predicted value and β is the weight coefficient. These errors are considered to have a Gaussian distribution, which implies that errors exist in both the positive and negative directions, with many little errors and a few large ones. The well-known "ordinary least squares technique" (OLS) is one of the most commonly used approaches for estimating model parameters [11]. It is utilized to discover the weights β_j that minimize the squared differences between the actual and predicted results:

$$\hat{\beta} = \arg\min_{\beta_0,\cdots,\beta_p} \sum_{i=1}^{n} \left(y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_j^i \right) \right)^2. \tag{2.2}$$

In the medical field, for example, one of the important features is to quantify the impact of a medicine or therapy while also taking into consideration sex, age, and other factors in an interpretable manner [12]. Linear effects are simple to explain [13]. The assumption of normality ensures that the estimates are consistent and that the estimators have optimal qualities. If the assumptions are not met, the estimates will be warped and the findings will be invalidated. Another important properties are constant variance (homoschedasticity) and absence of multicollinearity. The type of the corresponding feature determines how a weight in a linear regression model is interpreted.

The feature importance can be obtained in following ways:

- **Interpretation of a Numerical Feature:** When all other feature values stay constant, increasing feature x_k by one unit improves the prediction for y by β_k units.
- **Interpretation of a Categorical Feature:** When all other characteristics stay constant, switching feature x_k from the reference to the other category increases the prediction for y by β_k.
- **Feature Importance:** The absolute value of a feature's t-statistic can be used to determine its relevance in a linear regression model. The predicted weight is scaled with its standard error in the t-statistic,

$$t_{\hat{\beta}_j} = \frac{\hat{\beta}_j}{SE(\hat{\beta}_j)}. \tag{2.3}$$

  With increased weight, the significance of a characteristic grows. The less essential the feature is, the greater the variation it has in its projected weight. SE is the standard error of the regression.

2.1.2 Decision Tress

The linear regression model fails in some circumstances where the relationship between results and individual features are nonlinear or where features interact with each other. Hence there is a necessity for tree-based model [14]. Tree-based models partition the data many times based on specified feature cutoff values. Different subsets of the dataset are formed as a result of the splitting, with each instance belonging to one of them. Terminal or leaf nodes are the final subsets, whereas internal nodes or split nodes are the intermediate subsets. The average outcome of the training data in this node is used to predict the outcome in each leaf node. Classification and regression may both be done with trees. A tree may be grown using a variety of algorithms. They differ in terms of the tree's potential structure (e.g., the number of splits per node), the criteria for locating splits, when to cease splitting, and how to estimate basic models within leaf nodes [15]. The relationship between the outcome y and the attributes x is described by the formula below,

$$\hat{y} = \hat{f}(x) = \sum_{m=1}^{M} c_m I_{x \in R_m}. \tag{2.4}$$

Each of the instances falls into exactly one leaf node (=subset R_m). $I_{x \in R_m}$ is known as identity function that returns 1 if in the subset R_m and 0 otherwise. The model takes a feature and calculates the cut-off point that minimizes the variance of y for a regression task. The

variance indicates how far a node's y values are distributed around its mean value. In addition, a Gini index c_m tells how impure the node is. When the data points in the nodes have relatively comparable y values, the variance and Gini index are reduced. As a result, the ideal cut-off point separates the two subgroups as much as possible in terms of the preferred outcome. The program aims to construct subsets for category features by experimenting with alternative groupings of categories. After determining the appropriate cutoff for each feature, the algorithm chooses the feature for splitting that would result in the best partition in terms of variance or Gini index, and then adds it to the tree. The algorithm repeats the search-and-split process in both new nodes until it reaches a stop condition.

The interpretation is straightforward: we start from the root node to the next node, and the edges indicate the subsets we are looking for. When we arrive at the leaf node, the node informs us of the expected result. ANDing all the possible edges gives us an explanation [16].

Feature importance: In a decision tree, the overall relevance of a feature can be calculated by observing all the splits that the feature used and evaluating its impact on variance and the Gini index compared to the parent node. The overall importance of all factors is multiplied by 100. This means that each value can be regarded as a percentage of the overall model value.

Tree decomposition Decomposing the decision path into one component per feature can explain individual predictions of a decision tree [17]. We can follow a choice through the tree and use the contributions added at each decision node to explain a prediction. The initial point in a decision tree is the root node. If root nodes were employed to make predictions, the mean of the training data's outcome would be predicted. To obtain the final prediction, we must trace the route of the data instance that we wish to explain and continue to contribute to the formula. A prediction of an instance is the mean of the target outcome plus the sum of all splits between the node at the root and the node at the terminal. The tree structure is ideal for capturing data interactions between features. The data is organized into distinct groups, which are frequently easier to understand than points on a multidimensional hyperplane, as in linear regression. The interpretation is arguably very simple.

2.1.3 Naive Bayes Classifier

Naive Bayes classifiers rely on the Bayes' theorem of conditional probabilities. Based on the value of each feature, it calculates the probability for a particular class. Naive Bayes classifiers calculate class probabilities independently for each feature, which is equivalent to the assumption that the features are independent of each other [18]. Naive Bayes is a conditional probability model that predicts the likelihood of a class C_k given an input x as follows:

$$P(C_k|x) = \frac{1}{Z} P(C_k) \prod_{i=1}^{n} P(x_i|C_k). \tag{2.5}$$

The term Z refers to a scaling parameter that ensures the sum of probabilities for all classes equals one (otherwise they would not be probabilities). A class's conditional probability is the class probability multiplied by the probability of each feature within the class, normalized by Z. x_i is the i^{th} input, where $i = 1, 2, \cdots n$. This formula is derived according to Bayes' theorem.

Because of the independence assumption, Naive Bayes is a model that can be interpreted. It can be interpreted on a modular level. Because we can interpret the conditional probability, it is

very clear how much each feature contributes to a specific class prediction [19].

2.1.4 Tsetlin Machine

TM is a recent ML model that has gained significant interest because of its human-understandable pattern recognition by composing patterns in propositional logic. It has surpassed many cutting-edge pattern recognition approaches in benchmarks involving pattern discrimination, image recognition, text classification, tabular data, and optimal move prediction for board games without sacrificing the crucial virtue of interpretability [1, 20, 20, 21, 22]. A revolutionary game theoretic system that manages decentralized teams of Tsetlin Automata (TA) is at the core of the TM. The orchestration directs the TA to cooperatively learn arbitrary complicated propositional formulations in conjunctive normal form, capturing the various aspects of the patterns seen. These equations have proven to be particularly well adapted for human interpretation while still allowing for the formation of complicated nonlinear patterns. The TM represents inputs, patterns, and outputs as bit sequences and is very straightforward to understand. Decentralized manipulation of those bits is then required for pattern recognition. This gives the TM a computing edge over approaches that rely on more intricate mathematical models.

The main important aspect of TM is that it learns both linear and nonlinear relationships in the data without losing the interpretability of the model. TM uses propositional logic to learn correlations between features and labels. A propositional logic formula in TM, namely a clause, is a conjunction of negated and non-negated forms of the input features. Such negated and non-negated (original) features are called literals and are controlled by TA.

The clause, which represents a specific sub-pattern among a specific set of patterns, is the most important component of TM. This sub-pattern is in propositional AND-form, which makes it highly interpretable and amendable for logical task understanding. To have a clear comprehension of what a clause looks like, let us consider a bag-of-words input $X = [x_1, \cdots, x_n]$, $x_k \in \{0, 1\}$, $k \in \{1, \ldots, n\}$, where $x_k = 1$ means the presence of a word in the sentence and n is the size of the vocabulary. Let us assume there are γ classes in total. If each class needs α clauses to learn the pattern, altogether the model is represented by $\gamma \times \alpha$ clauses C_ι^κ, $1 \leq \kappa \leq \gamma$, $1 \leq \iota \leq \alpha$, as:

$$C_\iota^\kappa = \left(\bigwedge_{k \in I_\iota^\kappa} x_k \right) \wedge \left(\bigwedge_{k \in \bar{I}_\iota^\kappa} \neg x_k \right), \tag{2.6}$$

where I_ι^κ and $\bar{I}_\iota^\kappa$ are non-overlapping subsets of the input variable indices, $I_\kappa^\iota, \bar{I}_\kappa^\iota \subseteq \{1, \cdots, n\}$, $I_\kappa^\iota \cap \bar{I}_\kappa^\iota = \emptyset$. I_ι^κ represents the set of indices of the features that the TAs include in original form, while the set $\bar{I}_\iota^\kappa$ contains the indices of the features that the TAs include in negated form.

Here, clauses with odd indexes in each class are allocated positive polarity (+), whereas those with even indexes are assigned negative polarity (-). Positive polarity clauses vote in favor of the target class, while negative polarity clauses vote against it. As demonstrated in Eq. (2.7), a summation operator aggregates them by subtracting the total number of negative votes from the total number of positive votes,

$$f^\kappa(X) = \Sigma_{\iota=1,3,\ldots}^{\alpha-1} C_\iota^\kappa(X) - \Sigma_{\iota=2,4,\ldots}^{\alpha} C_\iota^\kappa(X). \tag{2.7}$$

For γ classes, the final output $\hat{y}$ is given by the argmax operator to classify the input based

on the highest net sum of votes, as shown in Eq. (2.8).

$$\hat{y} = \text{argmax}_{\kappa} \left(f^{\kappa}(X) \right). \tag{2.8}$$

In a nutshell, each input feature corresponds to two TAs, *TA* and *TA'*. *TA* controls the literal's original (non-negated) form, whereas *TA'* controls its negation [22]. Each TA has two actions (Include/Exclude) with $2N$ states and decides whether to include or exclude the literal. When a TA transits from state 1 to state N, the action "Exclude" is executed. When a TA transits from state $N + 1$ to state $2N$, it executes the "Include" action. Feedback in the form of Reward, Penalty, or Inaction is adopted to trigger each TA move. The *TA* or *TA'* that received reward will move away from the center while those that received penalty will move towards the center. In this way, the *TA* (or *TA'*) can be trained to either "include" or "exclude" a word (or its negation), helping the clauses, which are composed by the literals, learn different subpatterns. Consequently, the TM, composed by clauses, will gradually converge to the intended pattern. The feedback (reward or penalty) given to the TM follows two types: Type I and Type II feedback. Based on these feedback types, rewards or penalties are fed to the TA for the training samples. Type I Feedback is activated when a given input feature is either correctly assigned to the target sentiment (true positive) or mistakenly ignored (false negative). This feedback provides two countering effects: (1) involving more literals from the sample to refine the clauses; (2) trimming of the clauses by a factor specificity s that makes all clauses eventually evaluate to 1. The s-parameter is also responsible for avoiding overfitting. Type II Feedback is activated when an input feature is wrongly assigned to the target sentiment (false positive). It is responsible for introducing literals that make the clause evaluate to false, every time a false positive occurs. Type I Feedback and Type II Feedback are summarized in Tables B.1 and B.2 respectively.

2.2 Deep Learning

Deep learning has a long and illustrious history, as well as numerous goals. Several recommended solutions have yet to bear fruit in full. For supervised learning, modern deep learning provides a robust foundation. A deep network can express functions of increasing complexity by adding more layers and units within each layer [23]. Given sufficiently large models and huge datasets of labeled training examples, deep learning can accomplish the task that consists of mapping an input vector to an output vector. The fundamental deep learning models are deep feedforward networks, commonly known as feedforward neural networks or multi-layer perceptrons (MLPs). These models are known as feedforward because of their flow of information through a computation being evaluated from input to yield output. For machine learning practitioners, feedforward networks are extremely important. Many important business applications are built upon them. Because feedforward neural networks are often depicted by combining several different functions, they are referred to as networks. The model is linked to a directed acyclic graph that shows how the functions are connected. Some of the sophisticated deep learning models are Recurrent Neural networks (RNN), Convolutional Neural Networks (CNN), and Transformers.

2.2.1 Deep Neural Networks

In recent years, neural networks have emerged as the preferred tool for NLP [24]. This subsection will provide an overview of the basic building blocks used in neural networks. Neural networks can be thought of as compositions of functions [25]. Indeed, the basic machine learning models described so far, e.g., linear regression, and logistic regression, can be viewed as simple instances of a neural network.

Let us consider a multiclass logistic regression given by:

$$f(x) = Wx + b,$$
$$g(y) = softmax(y),$$

where $W \in \mathbb{R}^{C \times d}$, $x \in \mathbb{R}^d$, $b \in \mathbb{R}^C$, $y \in \mathbb{R}^C$, and C is the total number of classes and d represents the dimension of the input. From here onward, we will use W to represent a matrix of the weights, b is the set of parameters of the model. Logistic regression here can be observed as the functions f and g: $g(f(x))$ where $f(\cdot)$ is referred to as affine function and $g(\cdot)$ is represented as activation function which is softmax function in this case.

In recent years, neural networks have emerged as the preferred tool for NLP. This subsection will provide an overview of the basic building blocks used in neural networks. Neural networks can be thought of as compositions of functions. Indeed, the basic machine learning models described so far, e.g., linear regression, and logistic regression can be viewed as simple instances of a neural network.

$$h = \sigma_1(W_1 x + b_1),$$
$$y = softmax(W_2 h + b_2),$$

where σ_1 is the first hidden layer's activation function. It's worth noting that each layer has its weight matrix W and bias vector b. Layers mostly have distinct parameters, but tying or sharing of such parameters allows different layers to set their parameters to be the same. Such sharing of parameters is an important aspect because it induces inductive bias that often helps model with generalization. Forward propagation is the process of computing the output of one layer, such as h, which is then fed as input to subsequent layers, and eventually produces the output of the entire network, y. Because a collection of linear functions can be expressed as another linear function, deep neural networks' expressiveness stems primarily from their non-linear activation functions.

Activation functions: Most commonly used activation functions include sigmoid, softmax and rectified linear unit (ReLU) [26]. Another activation function known as hyperbolic tangent or tanh function is less commonly used function that outputs the values in the range of $(-1, 1)$:

$$\sigma(x) = \frac{\exp^x - \exp^{-x}}{\exp^x + \exp^{-x}}. \tag{2.9}$$

2.2.2 Recurrent Neural Network

NLP usually adopts models that can process a sequence of inputs because the text is sequential. The RNN is one of the most basic neural networks for sequential input [27]. An RNN is a feed-forward neural network that has a dynamic number of hidden layers with the same parameters.

As it unrolls through time, the model becomes "wide" rather than "deep." However, unlike a traditional feed-forward neural network, it accepts a new input at each "layer" or time step. The RNN, in particular, keeps a hidden state h_t, which represents its "memory" of the sequence at each time step t. At a given time step, RNN performs the following operations:

$$\begin{aligned} h_t &= \sigma_h(W_h x_t + U_h h_{t-1} + b_h), \\ y_t &= \sigma_y(W_y h_t + b_y), \end{aligned}$$

where σ_h and σ_y are the activation functions. The RNN modifies the previous hidden state h_{t-1} by applying a transformation U_h and a transformation W_h to the current input x_t, resulting in the new hidden state h_t. The RNN also produces an output y_t at each time step. In practice, learning over a large number of time steps is difficult for the RNN.

2.2.3 Long-Short Term Memory (LSTM)

When dealing with sequential data, long-short term memory networks are preferred over RNNs because they can retain information for longer time spans, which is necessary for modeling long-term dependencies found in natural language [28]. The LSTM can be considered as a more advanced RNN cell with mechanisms for deciding what should be "remembered" and what should be "forgotten." The LSTM adds a forget gate f_t, an input gate i_t, and an output gate o_t to the RNN, all of which are functions of the current input x_t and the previous hidden state h_t. These gates allow the model to selectively retain or overwrite information by interacting with the previous cell state c_{t-1}, the current input, and the current cell state c_t. The overall operation of the model is given by:

$$\begin{aligned} f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f), \\ i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i), \\ o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o), \\ c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c), \\ h_t &= o_t \circ \sigma_h(c_t). \end{aligned}$$

Here σ_g is the activation function of type sigmoid, σ_c and σ_h are the tanh activation function, and $\circ$ represents element-wise multiplication which is also known as Hadamard product. Multiple layers of LSTM cells can be stacked. In [29], it is introduced a bidirectional LSTM that runs separate LSTMs forward and backward over the sequence in most NLP tasks. The concatenation of the hidden states from the forward and backward LSTMs at time step t is the hidden state h_t and is given by

$$h_t = [h_{fwd}; h_{bwd}]. \tag{2.10}$$

2.2.4 Transformers

The Transformer [30] is a sequence-to-sequence model that is made up of an encoder and a decoder, each of which is made up of a stack of L identical blocks. A multi-head self-attention module and a position-wise feed-forward network (FFN) are the most important components

of each encoder block. A residual connection [31] is applied in each module to build a deeper model, followed by the Layer Normalization [32] module. Decoder blocks, in contrast to encoder blocks, also add the cross-attention modules between multi-head self-attention modules and position-wise FFNs. In addition, the decoder's self-attention modules have been tweaked to prevent each position from paying attention to subsequent positions.

Attention Modules

Transformer is based on attention mechanism with Query-Key-Value (QKV) model. If the the query matrix representation is given by $Q \in \mathbb{R}^{N \times D_k}$, keys matrix is $K \in \mathbb{R}^{M \times D_k}$ and value matrix is $V \in \mathbb{R}^{M \times D_v}$, then the scaled dot-product attention is given by:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{D_k}}\right)V = AV, \tag{2.11}$$

where N and M represents the lengths of queries and the keys (or values) respectively. D_k and D_v represent the dimensions of keys (or queries) and values respectively, and $A = softmax(\frac{QK^T}{\sqrt{D_k}})$ is called attention matrix where softmax is applied in row-wise manner. To avoid the softmax function's gradient vanishing problem, the dot-products of queries and keys are divided by $\sqrt{D_k}$.

Transformer employs multi-head attention instead of a single attention function, in which the D_m-dimensional original queries, keys, and values are projected into D_k, D_k, and D_v dimensions, respectively, using H different sets of learned projections. The keys and values, as well as the output, are computed with attention for each of the projected queries. The model then joins all of the outputs together and projects them back to a D_m-dimensional representation, shown below.

$$\begin{aligned} MultiHeadAttn(Q, K, V) &= Concat(head_1, \cdots, head_H)W^0, \\ where\ head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V). \end{aligned}$$

Position-wise FFN

The position-wise feed forward network is a fully connected module that operates individually and identically on each position.

$$FFN(H') = ReLU(H'W_1 + b^1)W^2 + b^2, \tag{2.12}$$

where H' is the output of the previous layer, and $W^1 \in \mathbb{R}^{D_m \times D_f}$, $W^2 \in \mathbb{R}^{D_f \times D_m}$, $b^1 \in \mathbb{R}^{D_f}$, $b^2 \in \mathbb{R}^{D_m}$ are the trainable parameters. Usually, the intermediate dimension D_f of FFN is set to be larger than D_m.

Residual Connection and Normalization

Transformer employs a residual connection around each module, followed by Layer Normalization, to generate a deep model. A transformer encoder block, for example, can be written as:

$$\begin{aligned} H' &= LayerNorm(SelfAttention(X) + X), \\ H &= LayerNorm(FFN(H') + H'), \end{aligned}$$

where *SelfAttention(·)* stands for the self-attention module and *LayerNorm(·)* stands for the layer normalization operation.

Position Encodings

Transformer is unaware of positional information because it does not use recurrence or convolution (especially for the encoder). As a result, additional positional representation is required to reflect token ordering.

2.3 Text Representation

NLP is a technique for teaching computers to understand human language. In NLP domain, representation of text is one of the most important aspect which generally map a text to linguistic structures that encode its meaning [33]. There are various ways of representing the words or sentence/context in the NLP models. One of the simple ways to represent words is to utilize Bag-of-words (BOWs). Many sophisticated representations are known as word embedding or sentence/context embedding. These are trained mathematical weights that represent each word or sentence/context that holds the semantics and syntactic relationships with other words or sentence/context. Here we explain these techniques in detail.

2.3.1 Bag-of-words

It is one kind of text representation usually denoted by $x_i \in \mathbb{R}^{|V|}$, where V is the vocabulary used for the particular task [34]. The number of occurrences of the i^{th} word in the vocabulary in the text corresponds to each entry x_i. This is also known as Boolean BOWs which is the base for text representation in this thesis. This frequency can be weighted using the term frequency-inverse document frequency (tf-idf) metric, which also indicates the importance of a phrase in a corpus of texts. In a text, an n-gram is a continuous series of n words. Only unigrams, or one-word sequences, are considered in the typical BOW model. Grammar and word order are ignored in the BOWs. A sequence of words $w_1, \cdots, w_T$ is usually encoded as a sequence of the matching word embeddings $x_1, \cdots, x_T$ in order to capture compositionality and dependencies in the input.

2.3.2 Word2Vec Embedding

Word2Vec is a free and open-source word embedding prediction model that is trained using neural network [35]. This model creates a vector of each word using two hidden layers in a shallow neural network. Continuous BOWs (CBOWs) and Skip-gram models of word2vec capture word vectors that are expected to contain semantic and syntactic information. It is recommended that the corpus be trained using a large corpus in order to have a better representation of words. Word2Vec has been useful in a variety of NLP tasks [36]. Word2Vec was created to make embedding training more important, and it has subsequently become a standard for developing pre-trained word representations. Depending on the context, word2Vec predicts using one of two neural network models: CBOW or Skip-gram. In both models, the corpus and a predetermined length of a window are shifted together, and the training is done with words inside the window

in each step [37]. This feature presentation algorithm provides a powerful tool for unraveling corpus relationships and token similarity. For example, in the vector space, this technique would consider two words like "small" and "smaller" to be close to each other.

Continuous Bag of words: CBOWs predict words in current work based on context. The nearby words in the window communicate with CBOWs. The CBOWs technique employs three levels. The first layer is context, while the hidden layer corresponds to the estimation of each word from the input to the weight matrix, which is then estimated for the third layer, which is output. The final step in this strategy is to use backpropagation of the error gradient to correlate the output and the task itself in order to improve the representation. In the CBOWs approach, the middle word is predicted based on its context, whereas in the skip-gram method, the context word is predicted based on the center word [38].

Skip-Gram: Skip-Gram is the inverse of the CBOW model in which it predicts based on the central word after context training. The targeted word is associated with the input layer, and the context is associated with the output layer. In contrast to CBOWs, this approach seeks to estimate the context given the word. The correlation between output and every word of the context is the final phase of this model, which is used to adjust representation using backpropagation [38, 39]. When there is little training data and not frequently occurring words, skip-gram is effective. CBOWs, on the other hand, is faster and perform better with repeated words. Two techniques are proposed to handle the issue of learning the final vectors. The first is negative sampling, which limits the number of output vectors that need to be updated so that only a sample of them is updated based on a noise distribution (a probabilistic distribution used in the sample step). Furthermore, hierarchical softmax, which is based on the Huffman tree, is another option. It's a binary tree that displays all words in the order of their counts. Then each step from the root to the target is normalized. When the dimension of the vectors is low, negative sampling is effective, and it works well with repeated words. In contrast, hierarchical softmax works nicely with fewer words occurrence.

2.3.3 Global Vectors (GloVe)

Word embedding using Word2vec training will better capture word semantics and adjust word connections. Word2vec, on the other hand, focuses mostly on local context window knowledge, while global statistical data is underutilized. So the GloVe is introduced [40], which is a well-known technique based on the global co-occurrence matrix. In GloVe, each element X_{ij} shows the frequency with which the words w_i and w_j co-occur in a certain context window and is frequently used for text classification. GloVe is a word2Vec expansion for effectively learning word vectors, where the words prediction is based on the surrounding words. Glove is based on how many times a word appears in the corpus, which is done in two steps. The initial step is to create a co-occurrence matrix from the corpus, which is then factorized to obtain vectors. Word representation algorithms, such as Word2vec and GloVe, are simple, accurate, and they can learn semantic representations of words from vast data sets. However, they do not learn embedded words from terms that are out of their vocabulary, such as words that do not appear in the present training corpus or terms that do not appear in the current vocabulary. To solve this issue, a new

models is proposed known as FastText [41]. However, fasttext is not associated with this thesis.

2.3.4 Embedding from Language Models (ELMo)

ELMo [42] provides deep contextual word representations. Researchers agree that two issues should be considered in an appropriate word representation model: the dynamic nature of words used in semantics and grammar, and how these usages should change as the language environment evolves. To solve the two issues raised above, they develop a method of deep contextualized word representation. The bi-directional language model is applied to learn the final word vectors (forward and backward Language Model (LMs)). Instead of using solely the last layer representations like other contextual word representations, ELMo adopts a linear concatenation of representations learned via a bidirectional language model. ELMo gives alternative word representations for the same word in different sentences. It learns word representations based on the representations gained through the Bi-language model (BiLM). In both forward and backward LMs, the training phase of BiLMs utilizes the log-likelihood of sentences. After concatenating hidden representations from the forward $LM\overrightarrow{h}_{n,j}^{LM}$ and backward layers $LM\overleftarrow{h}_{n,j}^{LM}$, the final vector is produced, where $j = 1, 2, \cdots, L$, and it is given by

$$\begin{aligned} BiLM = \sum_{n=1}^{k} (&\log p(t_n|t_1, \cdots, t_{n-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) \\ &+ \log p(t_n|t_1, \cdots, t_{n-1}; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s)). \end{aligned}$$

The forward and backward directions share the token representation parameters and softmax parameters Θ_x and Θ_s, respectively. The forward $\overrightarrow{\Theta}_{LSTM}$ and backward $\overleftarrow{\Theta}_{LSTM}$ are parameters. ELMo extracts the representations acquired from BiLM from an intermediary layer and performs a linear combination for each token in a downstream task. As shown below, BiLM contains $2L + 1$ set representations.

$$\begin{aligned} R_n &= (X_x^{LM}, \overrightarrow{h}_{n,j}^{LM}, \overleftarrow{h}_{n,j}^{LM} | j = 1, \cdots, L) \\ &= (h_{n,j}^{LM} | j = 0, \cdots, L), \end{aligned}$$

where $h_{n,0}^{LM} = x_n^{LM}$ is the layer of token and $h_{n,j}^{LM} = [\overrightarrow{h}_{n,j}^{LM}, \overleftarrow{h}_{n,j}^{LM}]$ for each of the BiLSTM layer. ELMo is the combination of these given characteristics unique to the task, where all given layers in M are flattened to a single vector and is given by,

$$ELMo_n^{task} = E(M_n; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{h,j}^{LM}, \tag{2.13}$$

where s^{task} is a softmax normalized weight for the combining of representations from multiple layers, and γ^{task} is a hyper-parameter for optimization and scaling of the representation.

2.3.5 Bidirectional Encoder Representations from Transformers (BERT)

BERT [43] is trained on large amounts of free text and then fine-tuned separately without the use of custom network architectures. BERT is a contextualized word representation LM in which

the transformer DNN uses parallel attention layers instead of sequential recurrence [30]. BERT is trained on two concepts of unsupervised tasks that encourage bidirectional prediction and sentence-level understanding:

- Masked Language Model (MLM), where 15% of the tokens are masked randomly and trained to predict those masked tokens.
- Next Sentence Prediction (NSP), where a pair of sentences are given to the model and trained to identify if the two sentences are in context.

The purpose of the second assignment is to collect long-term or pragmatic information. BERT is trained based on the Books Corpus [44] dataset as well as English Wikipedia text passages. BERT-Base and BERT-Large are the two pre-trained models offered. BERT can be utilized to unannotated data or fine-tuned task-specific data obtained directly from the pre-trained model.

A growing amount of research is looking into what aspects of language BERT is able to learn from unlabeled data as a result of their recent success in NLP. Most recent investigation has focused on internal vector representations or model outputs (such language model surprise) (e.g., probing classifiers). Recent research has looked at this issue by analyzing the language model outputs on a set of carefully selected input sentences [45] or by looking into the internal vector representations of the model using techniques like probing classifiers [46].

A growing amount of research is looking into what aspects of language BERT is able to learn from unlabeled data as a result of their recent success in NLP. Most recent investigation has focused on internal vector representations or model outputs (e.g., probing classifiers). Recent research has looked at this issue by analyzing the language model outputs on a set of carefully selected input sentences [45] or by looking into the internal vector representations of the model using techniques like probing classifiers [46]. Regarding the interpretation of BERT, it was studied in [2] the structure of attention maps of the BERT. Specifically, they studied the behaviour of attention head and discovered recurring behaviors in them, such as paying particular attention to fixed positional offsets or paying attention extensively to the entire text. They also claimed that the deliminator token [SEP], which is used by the model as a sort of no-op, receives an unexpectedly large amount of attention from BERT.

2.4 Summary

In this chapter, we have discussed several background concepts underlining this thesis. This thesis studies different concepts of interpretable machine learning models, deep learning models and the models associated with NLP in detail. The preliminaries used in the literature form the basis for the proposed architecture in this thesis. This thesis combines the technology of each section described above to formulate interpretable architectures for NLP and to maintain the robustness of the models. We have employed TM in the majority part of the thesis (Paper A, B, D, E, F) and deep learning models as the support to TM and as the state of the art for comparison (Paper C, F).

Chapter 3

Contributions

In this chapter, we describe and elaborate on the contributions of this thesis. As discussed in the previous chapter, we divide our contributions into two aspects: Interpretable Text Classification using TM and Interpretable Text Classification using Neural Network. The main contribution of this thesis is to design interpretable architectures and algorithms for NLP where we first adopt the TM to model various NLP tasks along with its enhanced performance and robustness (to be detailed in 3.1). Thereafter, we design an interpretable model using DNN-based architecture where we remove positional embedding that includes ambiguity in the interpretation of the ABSA task. We then employ the information from both TM and DNN to enhance the attention's interpretation while maintaining the performance (to be detailed in 3.2). The final goal of the thesis is to have a fine-tuned balance between accuracy and interpretability.

3.1 Interpretable Text Classification Using TM

This chapter describes the NLP architectures and algorithms using TM. Specifically, we propose several architectures that deal with extracting novel interpreting methods with TM. For a better understanding of what the model's interpretation looks like, we adopt the WSD task that is based on BOW techniques (Subsection 3.1.1). We then extend it to design an algorithm that maps position-dependent features into TM, which helps TM to understand where to concentrate on a given context (Subsection 3.1.2). We employ a popular evaluation framework called ABSA task for the evaluations. Since the interpretation of the model comes with a decrement in accuracy and the Boolean nature of TM restricts us to use any pre-trained information, we design a feature augmentation technique that appends similar features that enhance the performance as well as interpretation of the model (Subsection 3.1.3). At last, we tackle one of the most important issues of NLP: Spurious Correlations. We extend our study on TM and design a robust text classification model against spurious correlations (Subsection 3.1.4). Hence this dis-jointly oriented architectures and algorithms, when put together, gives a robust and reliable interpretable NLP model.

3.1.1 Bag-of-words based Text Classification

The Boolean BOW is the base for text representation in TM. TM is a recent ML model that is interpretable because of its learning ability in the form of propositional logic. It is interpretable

in the sense that it decomposes problems into self-contained sub-patterns that can be interpreted in isolation. Each sub-pattern is represented as a conjunctive clause, which is a conjunction of literals, with each literal representing either an input bit or its negation. However, this logic is easily interpretable only if the combination of literals is in limited numbers, which is very subjective based on each individual. In the case of NLP, where the vocabulary can range to thousands, it is next to impossible to interpret it thereby making the claim questionable. Hence, we design a novel interpreting technique using the frequency of the literal appearing in the clause. This technique can compact the number of literals and can be obtained based on the count for easier interpretation.

WSD is one of the unsolved tasks in NLP [47] with rapidly increasing importance, particularly due to the advent of chatbots. WSD consists of distinguishing the meaning of homographs – identically spelled words whose sense or meaning depends on the surrounding context of words in a sentence or a paragraph. WSD is one of the main NLP tasks that still revolves around the perfect solution of sense classification and indication [48], and it usually fails to be integrated into NLP applications [49]. Many supervised approaches attempt to solve the WSD problem by training a model on sense annotated data [50]. However, most of them fail to produce interpretable models. Because word senses can be radically different depending on the context, interpretation errors can have adverse consequences in real applications, such as chatbots. It is therefore crucial for a WSD model to be easily interpretable for human beings, by showing the significance of context words for WSD.

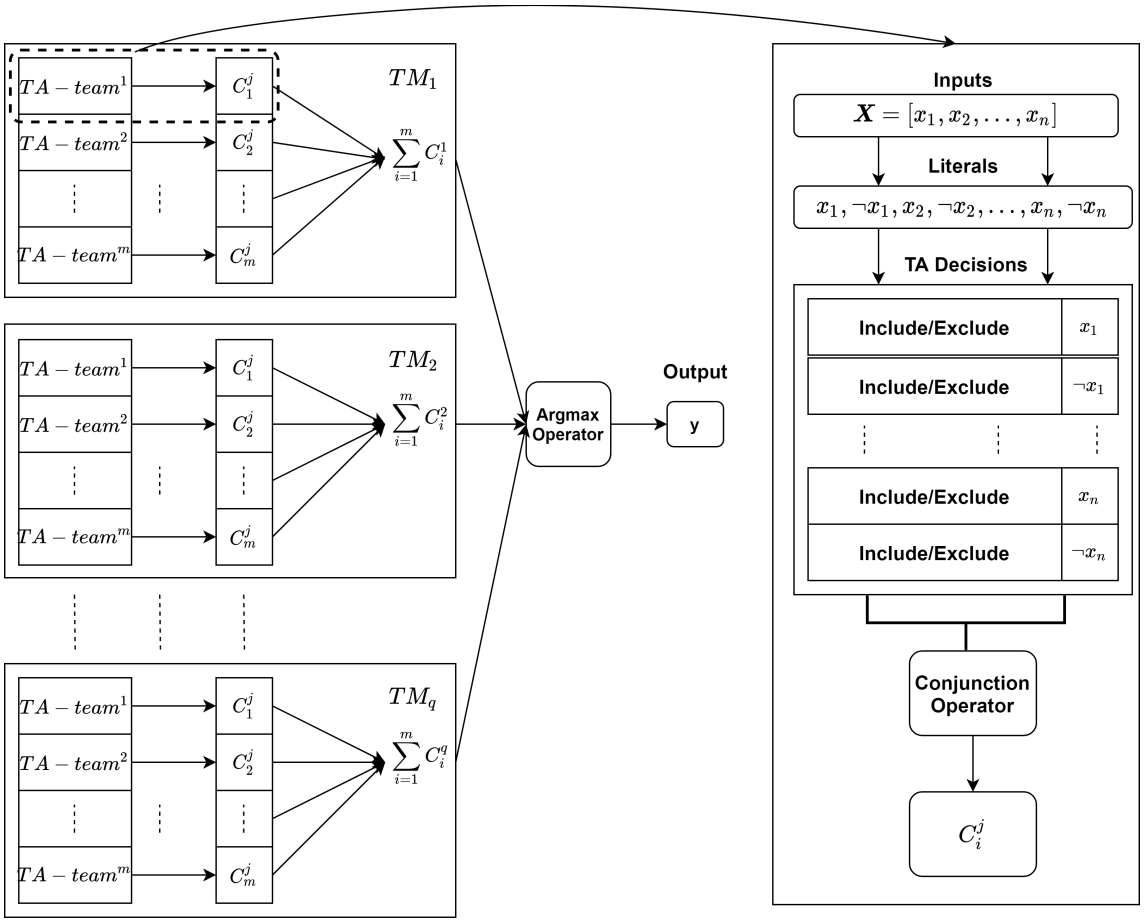
Although some of the rule-based methods, like decision trees, are somewhat easy to interpret, other methods are out of reach for comprehensive interpretation [51], such as DNNs. Despite the excellent accuracy achieved by DNNs, the "BlackBox" nature impedes their impact [52]. It is difficult for humans to interpret the decision-making process of artificial neurons. Weights and bias of deep neural networks are in the form of fine-tuned continuous values that make it intricate to distinguish the context words that drive the decision for classification. Some straightforward techniques such as Naive Bayes classifier, logistic regression, decision trees, random forest, and support vector machine are therefore still widely used because of their simplicity and interpretability. However, they provide reasonable accuracy only when the data is limited.

3.1.1.1 Basic Concept of TM for Classifying Word Senses

The first step in our architecture for WSD is to remove the stop-words from the text corpus, and then stem the remaining words[1]. Thereafter, each word is assigned a propositional variable $x_k \in \{0, 1\}$, $k \in \{1, 2, \ldots, n\}$, determining the presence or absence of that word in the context, with n being the size of the vocabulary. Let $\boldsymbol{X} = [x_1, x_2, ..., x_n]$ be the feature vector (input) for the TM.

The above feature vector is then fed to a TM classifier as explained in Section 2.1.4, whose overall architecture is shown in Fig. 3.1. Multiclass TM consists of multiple TM and each TM has several TA teams which is expanded in Fig. 3.1(b).

[1]In this work, we used the PortStemmer package.



(a) (b)

Figure 3.1: The architecture of (a) multiclass TM, (b) a TA-team forms the clause C_i^j, $1 \leq j \leq q$, $1 \leq i \leq m$.

3.1.1.2 Interpretable Classification Process

We now detail the interpretability once the TM has been trained. In brief, the interpretability is based on the analysis of clauses. Let us consider the noun "apple" as the target word. For simplicity, we consider two senses of "apple", i.e., Company as sense s_1 and Fruit as sense s_2. The text corpus for s_1 is related to the apple being a company, whereas for s_2 it is related to the apple being a fruit.

Let us consider a test sample $I_{test} = [\text{apple}, \text{launch}, \text{iphone}, \text{next}, \text{year}]$ and how its sense is classified based on the context words. This set of words is first converted to binary form based on a bag of words.

To extract the clauses that vote for sense s_1, the test sample I_{test} is passed to the model and the clauses that vote for the presence of sense s_1 are observed as shown in Fig. 3.2. The literals formed by TM are expressed in indices of the tokens. For ease of understanding, it has been replaced by the corresponding word tokens. The green box shows that the literal is non-negated whereas the red box denotes the negated form of the literal as shown in Fig. 3.2. For example, the sub-patterns created by clause $C_3 = \text{apple} \wedge \neg\text{orange} \wedge \neg\text{more}$. These clauses consist of included literals in conjunctive normal form (CNF). Since the clauses in the TM are trained

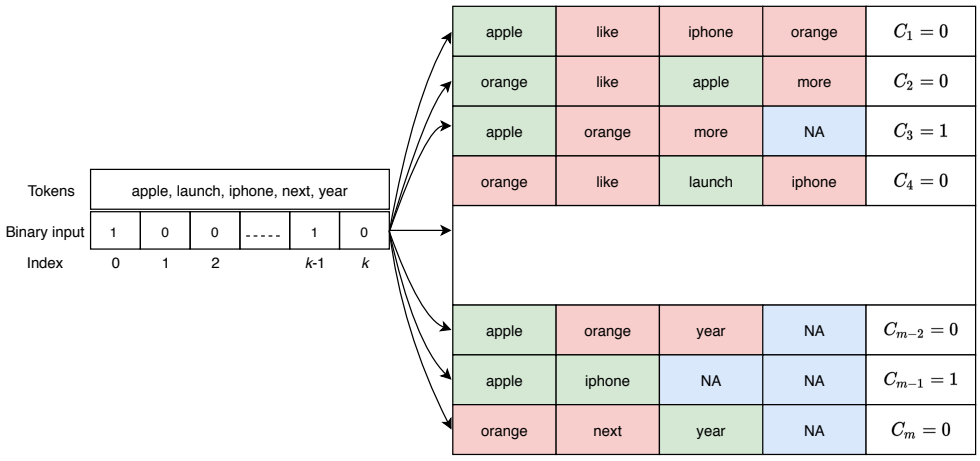


Figure 3.2: Structure of clauses formed by the combination of sub-patterns. Green color indicates the literals that are included as original, red color indicates the literals that are included as the negated form and the blue color boxes indicates that there are no literals because not all the clauses have the same number of literals.

sample-wise, there exist several randomly placed literals in each clause. These random literals just occur because of randomly picked words that do not effect the classification. These literals are assigned to be non-important literals and their frequency of occurrence is low. On the other hand, the literals that have a higher frequency among the clauses are considered to be important literals and hence makes significant impact on classification. Here, we emphasize on separating important and non-important literals for easy human interpretation. The general concept for finding the important words for a certain sense is to observe the frequency of appearances for a certain word in the trained clauses. To do that, in the above example, once the TM is trained, the literals in clauses that output 1 or votes for the presence of the class s_1 for I_{test} are collected first, as shown in Eq. (3.1):

$$L_t = \bigcup_{\substack{k,j, \\ \forall C_j=1}} \{x_k^j, \neg x_k^j\}, \tag{3.1}$$

where x_k^j is the k^{th} literal that appears in clause j and $\neg x_k^j$ is the negation of the literal. Note that a certain literal x_k may appear many times in L_t due to the multiple clauses that output 1. Clearly, L_t is a set of literals (words) that appears in all clauses that contribute to the classification of class s_1. The next step is to find frequently appearing literals (words) in L_t, which correspond to the important words. We define a function, $\beta(h, H)$, which returns the number of the elements h in the set H. We can then formulate a set of the numbers for all literals x_k and their negations $\neg x_k$ in L_t, $k \in \{1, 2, \ldots, n\}$, as shown in Eq. (3.2):

$$S_t = \Big\{ \bigcup_{k=1:n} \beta(x_k, L_t), \bigcup_{k=1:n} \beta(\neg x_k, L_t) \Big\}. \tag{3.2}$$

We rank the number of elements in set S_t in descending order and consider the first η percent in the rank as the important literals. Similarly, we define the last η percent in the rank as non-important literals. To distinguish the important literals more precisely, several independent experiments can be carried out for a certain sense. Following the same concept, the literals in M

Datasets	Micro-F1				Macro-F1			
	FTX-B	FTX-C	BRT-B	TM	FTX-B	FTX-C	BRT-B	TM
Apple	96.3	97.8	99.0	97.58	96.6	97.7	99.0	97.45
JAVA	98.7	99.5	99.6	99.38	61.1	84.1	99.8	99.35
Spring	86.9	92.5	97.4	90.78	78.8	96.4	97.2	90.76
Crane	87.9	94.9	94.2	93.63	88.0	94.8	94.1	93.62

Table 3.1: Results on the full CoarseWSD balanced dataset for 4 different models: FastText-Base (FTX-B), FastText-CommonCrawl (FTX-C), 1 Neural Network BERT-Base (BRT-B) and TM. Table cells are highlighted (dark blue to light blue) for better visualization of accuracy.

different experiments can be collected to one set $L_t(total)$ as shown in Eq. (3.3):

$$L_t(total) = \bigcup_{e=1}^{M} (L_t)_e, \tag{3.3}$$

where $(L_t)_e$ is the set of literals for the e^{th} experiment. Similarly, the counts of all literals in these experiments, stored in set $S_t(total)$ shown in Eq. (A.8), are again ranked and the top η percent is deemed as important literals and the last η percent is the non-important literals. The parameter η is to be tuned according to the level of human interpretation required for a certain task.

$$S_t(total) \quad = \quad \left\{ \bigcup_{k=1:n} \beta(x_k, L_t(total)), \bigcup_{k=1:n} \beta(\neg x_k, L_t(total)) \right\}. \tag{3.4}$$

3.1.1.3 Results

To illustrate the interpretability, let us take a sample as an example to extract the literals that are responsible for the classification of an input sentence: "**former apple ceo, steve jobs, holding a white iphone 4**". Once this input is passed through the model, TM predicts its sense as a company and we examine the clauses that output 1. We append all the literals that are presented in each clause and calculate the number of appearances for each literal. The number of appearances of a certain literal for the selected sample after one experiment (exp1) is shown in Figs. 3.3 and 3.4 by a blue line. After five experiments (exp5), the number for a certain literal is shown by a red line in Figs. 3.3 and 3.4. Clearly, it makes sense that the negated form of the mostly-appearing literals in Fig. 3.3, i.e., "not tree", "not fruit", " not cherries" etc. indicate that the word "apple" does not mean a fruit but a company. Nevertheless, as stated in the previous section, there are also some literals that are randomly placed in the clause and are non repetitive because the counts refuse to climb up for the same input, marking them not important literals, as shown in Fig. 3.4.

3.1.2 Position Dependent Text Classification

From the previous section, we have seen that a simple BOW technique performs decently using TM and the interpretation also makes sense. However, some NLP tasks are complicated and needs more information than words in the context. There are numerous tasks where the positions

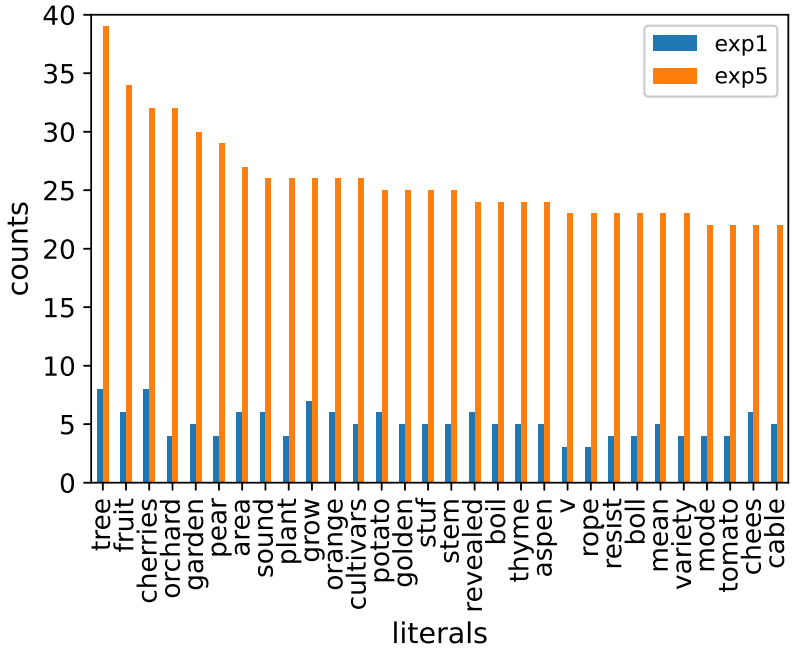


Figure 3.3: Count of first 30 literals that are in negated form for classifying the sense of apple as company. (considered as important literals)

of the words are important to be considered. One such task is ABSA where the sentiment of the aspect word has to be categorized based on the context where it has been used. DNN models that employ LSTM/GRU as language representation integrate positional embedding to use additional information about aspect words. Since TM works with Boolean input, there is no simple way to integrate this information in binary form. Hence, we propose a binary-form BOW that is encoded so that the information on the aspect word and context words is retained for sentiment classification. We further adapt the BOWs as input to the TM, enabling learning of aspect-based sentiment patterns in propositional logic.

Sentiment analysis, which identifies people's opinions on specific topics, is a classic problem in NLP. Under the umbrella of sentiment analysis, ABSA is a fine-grained evaluation framework for sentiment classification [53], and it has become a hot research topic [54]. Among various tasks in ABSA, this study focuses on the sentiment polarity (positive, neutral, negative) of a target word in given comments or reviews. For example, let us consider a review: *"Certainly not the best **sushi** in New York, however, it is always fresh and the **place** is very clean, sterile"*. The target word *"sushi"* is closely associated with its context words *"not best"*, assorting it as a negative polarity. The target word, *"place"*, is associated with its context words *"clean"* and *"sterile"*, classifying it as a positive sentiment. Such a complex form of sentiment classification is highly dependent on where the word appears in the sentence. To address this challenge, several recent approaches to ABSA have been based on attention mechanisms [55]. Although the accuracy of attention-based ABSA approaches are progressively improved, the interpretability of these models is still questionable, making them less trustworthy. Not surprisingly, little research has been done on ABSA learning techniques that are interpretable at a human level [56].

Recently, interpretable AI has taken a big leap in industrial application [57]. Indeed, the scientific community has performed extensive research on ways to interpret neural networks. In a

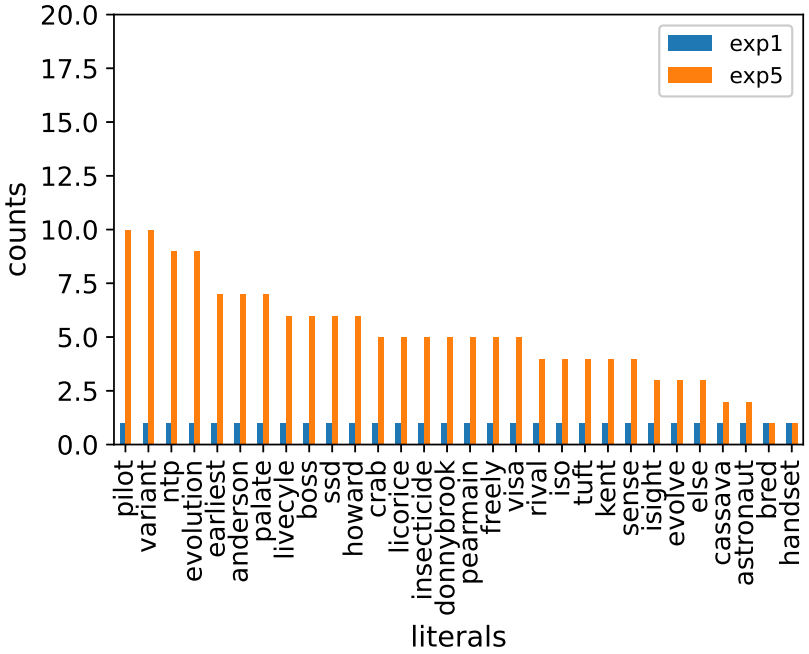


Figure 3.4: Count of last 30 literals that are in negated form for classifying the sense of apple as company. (considered as non-important literals)

modern neural network, one can use the fact that the variants of attention [58] assign soft weights to the input representations, and then extract highly weighted tokens as rationales. However, these attention weights do not provide faithful explanations for classification [3, 4, 59, 60]. On the other hand, certain classic models, like Decision Trees, are particularly easy to understand, yet still compromise on accuracy compared with neural networks. Hence, an effective trade-off between accuracy and interpretability has still not been achieved.

3.1.2.1 Input Binarization

Usually, the sentiment of the aspect word is reflected by its surrounding words in a sentence, as shown in Fig. 3.5. In this example, the aspect word “price” has positive sentiment due to the word “reasonable” in the context. Similarly, for “service”, the context word “poor” describes its negative sentiment. This reveals that the sentiment of aspect words heavily relies on its position in a sentence and thus position embedding [61] is necessary. Such embedding creates a probability distribution of the sentence based on the aspect word. Recently, position-aware modelling has shown promising results on ABSA tasks [62].

Since TM requires binary inputs, to utilize TM for interpretability, the inputs must be binarized. It is challenging to incorporate the required position-based word relations in binary form, to allow for ABSA. In particular, since a TM does not employ any world knowledge like Word2vec [35], Elmo [63] or BERT [43], so as to retain the interpretability of the model, we reduce the size of vocabulary by replacing the sentiment carrying words with a common token. Understandably, without pre-trained embeddings, a model cannot find the similarity between two semantically related words such as “excellent” and “good”. Hence, we adopt Opinion Lexicon [64], which is a list of English positive and negative sentiment words. In more details, we replace every possible word in the dataset by the common token “positive” or “negative”, as

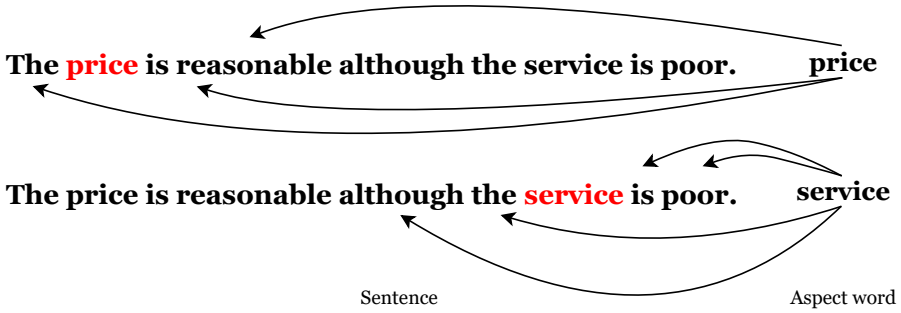


Figure 3.5: Representation of an aspect word and its surrounding words.

shown in Fig. 3.6. Such external knowledge also helps to reduce the vocabulary size thereby decreasing the sparsity of BOW representations.

The price is reasonable although the service is poor.

⇩

The price is positive although the service is negative.

Figure 3.6: Replacement of sentiment-carrying words with a common sentiment token using Opinion Lexicon.

Once the vocabulary size is determined, the context word and the aspect word can be converted into binary form, named as $BOW_{context}$ and BOW_{aspect} respectively. Since BOW in binary form does not consider the frequency of the replaced common tag (i.e., "positive" and "negative"), it becomes a rough representation of those tokens. In order to determine the location of these sentiment-carrying tokens, the sentence is split into two parts, divided by the aspect word. More specifically, we create additional binary vectors LOC^1_{vec} and LOC^2_{vec}, representing the location of the common tokens. The dimension of LOC^1_{vec} and LOC^2_{vec} is three (the 1^{st} bit: negative, the 2^{nd} bit: no sentiment, the 3^{rd} bit: positive) as shown in Fig. 3.7. LOC^1_{vec} represents the presence of the common tokens "positive" or "negative" in the first part. If there are no sentiment tags, this is represented by "no sentiment". Similarly, LOC^2_{vec} represents the presence of the common tokens in the second part.

After the pre-processing of inputs, we use SentiWordNet to obtain the sentiment score (SC) of the 1^{st} part and the 2^{nd} part of the split sentence. This involvement of such additional knowledge enrich the input information. We adopt the sentiment score in a 3-D binary form for each part of the sentence. The SC vector SC^1_{vec} for the 1^{st} part of the context is given by Eq. (3.5). Similarly, vector SC^2_{vec} is utilized for the second part of the context.

$$SC^1_{vec} = \begin{cases} [0,0,1](positive), & \text{if } SC > 0, \\ [1,0,0](negative), & \text{if } SC < 0, \\ [0,1,0](no\ sentiment), & \text{if } SC = 0. \end{cases} \tag{3.5}$$

After processing all these binary representations, we concatenate them all to make a final input vector of size $(2n + 12)$ as shown in Fig. 3.8.

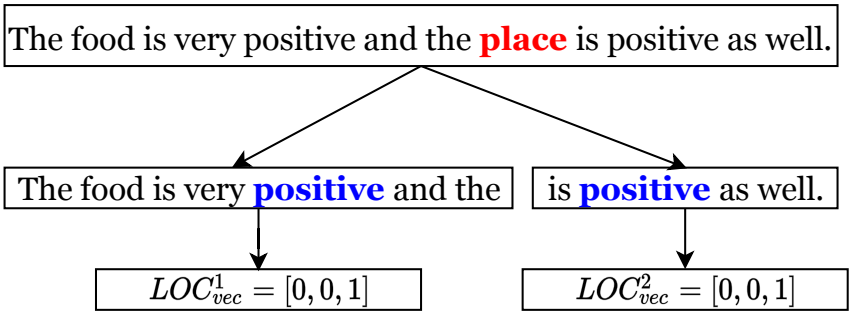


Figure 3.7: 3-bit input feature representing the location of common sentiment-carrying tokens: negative, no sentiment, and positive.

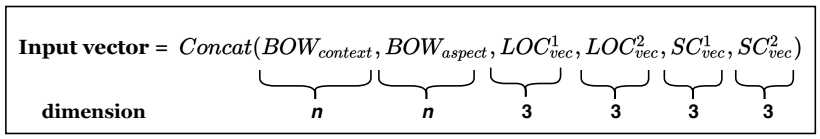


Figure 3.8: Construction of binary input by concatenating all the pre-processed features.

3.1.2.2 The TM based ABSA

TM has a novel game theoretic strategy that regulates a decentralized team of TAs. This strategy guides the TAs to learn an arbitrarily complex propositional formula by including or excluding certain literals. More specifically, the included literals, by the operation of conjunction, formulate clauses. Each clause, after training, is expected to capture a sub-pattern. The overall pattern is decided by summing up the output of all clauses for any unknown input. The architecture for ABSA using TM is shown in Figs. 3.9 and 3.10.

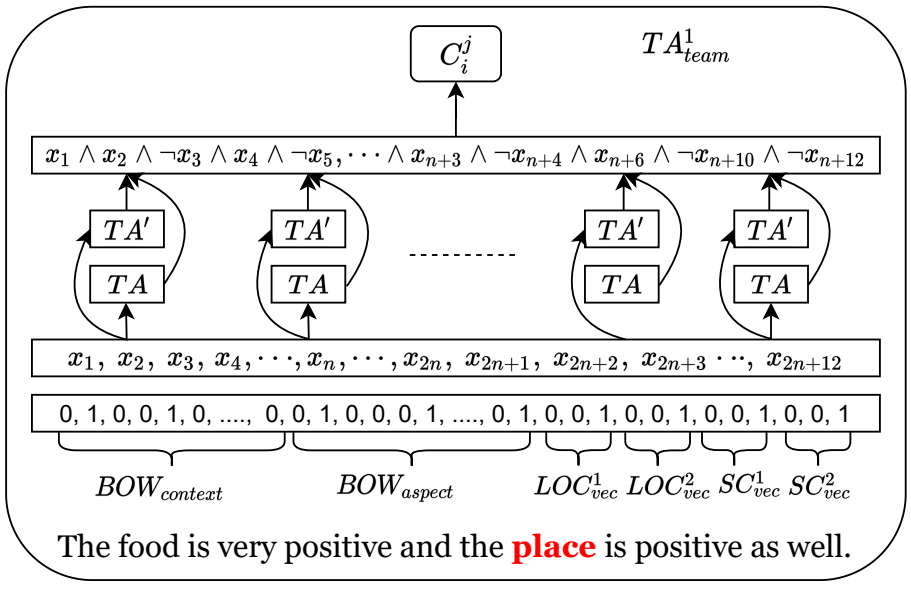


Figure 3.9: TA team forms a Clause C^j_i by either including or excluding the input features.

Let us consider the input feature as a vector with a vocabulary size of n words, which is represented in BOW as $X_s = [x_1, x_2, x_3, \cdots, x_n, \cdots, x_{2n}, x_{2n+1}, x_{2n+2}, \cdots, x_{2n+12}]$ with $x_k \in \{0,1\}$ and $k \in \{1, \ldots, 2n + 12\}$. Here, $[x_{2n+1}, x_{2n+2}, x_{2n+3}]$ and $[x_{2n+4}, x_{2n+5}, x_{2n+6}]$ represent

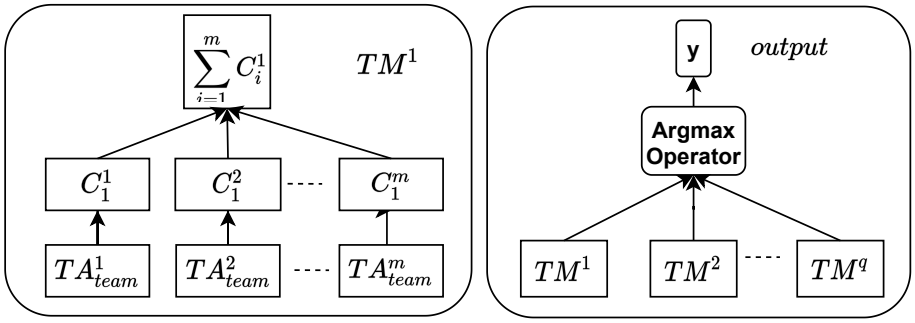


Figure 3.10: (a). The sum of the votes for the clauses offers a score for a particular class. (b). Argmax operator decides the output class based on the score of the clauses in each class.

LOC_{vec}^1 and LOC_{vec}^2 respectively. Similarly, $[x_{2n+7}, x_{2n+8}, x_{2n+9}]$ and $[x_{2n+10}, x_{2n+11}, x_{2n+12}]$ represent SC_{vec}^1 and SC_{vec}^2 respectively. We then feed this position dependent input feature X_s into the TM model for evaluation as explained in Section 2.1.4. The step-by-step learning process for ABSA-based TM is explained in Section B.3.3.

3.1.2.3 Results

For evaluation we used ABSA dataset from SemEval-2014 Task 4. The task has two domain-specific datasets, namely, Restaurant 14 (res14) and Laptop 14 (lap14). These datasets are provided with training and testing data. In our experiment, we evaluate the proposed method and compare it with related approaches for ABSA as baselines.

- **ContextAvg** averages the word embedding to form a context embedding [65].
- **LSTM** uses the last hidden vector of the LSTM for classification [28].
- **TD-LSTM** utilizes two LSTMs to learn the language model from the left and the right contexts of the aspect [65].
- **ATAE-BiLSTM** is an attention-based LSTM with Aspect Embedding model [66].
- **MemNet** integrates the content and the position of the aspect word into a deep neural network [65].
- **RAM** is a multi-layer architecture where each layer consists of attention-based aggregation of word features and a GRU cell [67].
- **IAN** is an Interactive Attention Network model that calculates the attention weights of the word in its sentiment and aspect interactively [55].
- **PRET+MULT** uses two approaches of transfer knowledge from document level using pretraining and multitask training [68].
- **HCSN** proposes a Human-like Semantic Cognition network for the ABSA task, motivated by the human beings' reading cognitive process [69]. We show that performance of our proposed scheme is quite similar to this technique with high interpretability.

- **TNet** employs a CNN layer instead of attention layer to extract features from the transformed word representations originated from a bi-directional RNN layer [70].
- **AGDT** is an Aspect-Guided Deep Transition model that uses the given aspect to direct the sentence encoding from scratch with specially designed deep transition architecture. This model generates the aspect-based sentence representation and hence predicts sentiment more accurately [71].

Methods	Restaurant 14		Laptop 14	
	Accuracy	Macro-F1	Accuracy	Macro-F1
ContextAvg	71.5	58.0	61.5	53.9
LSTM	74.3	63.0	66.5	60.1
TD-LSTM	75.6	64.5	68.1	63.9
ATAE-BiLSTM	77.6	65.3	68.7	64.2
MemNet	76.9	66.4	68.9	62.8
RAM	78.5	68.5	72.1	68.4
IAN	78.6	NA	72.1	NA
PRET+MULT	79.1	69.7	71.2	67.5
HCSN	77.8	70.2	76.1	72.5
TNet	80.79	70.84	76.01	71.47
AGDT	78.85	NA	71.50	NA
TM based ABSA	78.02 (76.40 ± 1.0)	67.85 (64.01 ± 0.8)	73.51 (71.47 ± 0.9)	70.82 (67.48 ± 1.5)

Table 3.2: Experiment results of various approaches for SemEval-2014 dataset. The upper results show the best reproducible accuracy and lower ones represent the mean and the standard deviation of the last 50 epochs when running the model for five times.

For pre-processing of text, we substitute the short form to its full form, such as "isn't" to "is not". Additionally, we stem the words to reduce the vocabulary size created due to spelling mistakes and variants of words[2]. The remaining pre-processing procedure has already been explained before. We train the TM model on both the datasets for 100 epochs each. Since the output sentiment label has imbalanced training samples, we use two evaluation metrics: Accuracy and Macro-F1 [72]. Following most of the related studied within the ABSA task, we report the best reproducible results by running the ABSA TM for 100 epochs, as shown in Table 3.2. We have reported the highest reproducible accuracy along with its mean and standard deviation obtained during 5 experiments. As we can see, Context2vec and LSTM perform quite poorly as they do not consider the aspect information when deciding the sentiment polarity. However, due to the consideration of left and right context information, TD-LSTM performs slightly better than LSTM. The variants of attention perform consistently better than LSTM and TD-LSTM. This is due to the fact that attention captures important information with regard to the aspect word. Other methods like RAM and MemNet perform slightly better because of the integrated memory in sentiment modeling. Another kind of the neural network-based

[2]In this work, we adopt the Porter Stemmer.

model is HCSN. HCSN utilizes a human-being-like cognitive network for ABSA, which is motivated by the principles of human beings' reading cognitive processes. Its pre-reading, active reading, and post-reading technique mimics the human behavior, which is then fed to the GRU network. As interesting as it seems, the involvement of the neural network still brings this below human-level interpretation on what drives the model to make the decision. Our model, which offers a transparent view of the learning process, obtains quite similar or higher accuracy compared to HCSN and PRET+MULT techniques. However, the TNet architecture with a CNN layer, which extracts salient features from transformed word representation, achieves higher accuracy compared to TM. AGDT is a model that uses Aspect guided GRU along with Max pooling to obtain Aspect Concatenated Embedding. It obtains quite similar accuracy compared to TM on Restaurant 14, whereas accuracy is lower on Laptop 14. Note that we do not use any pre-trained word2vec or GloVe embedding for TM and our model still performs better than LSTM, TD-LSTM as well as attention-based BilSTM for both datasets. The Macro-F1 score shows that TM does not only greedily learn a particular class but also creates a set of features for each and every class. Even though the performance of our proposed model does not outperform the state-of-the-arts models, it reaches to comparable accuracy and Macro-F1 with transparent learning and interpretable prediction.

In addition to the above comparisons, we demonstrate here the necessity of including both $LOCs$ and SCs vectors. First we only used the $LOCs$ in the model and observed that the accuracy of the model reaches 76.51%. Secondly, we replaced $LOCs$ with SCs and the performance of the model decreased to 75%. This shows that both vectors add useful information when employed together thereby reaching the stated accuracy of 78.02%.

To compare the performance of TM with classical interpretable models such as Logistic Regression (LR), we use our preprocessed BOW as input to LR. We observed that the TM performs better than LR in terms of accuracy. LR obtains the accuracy of 75.38% as compared with TM's 78.02% on the Restaurant 14 dataset. Indeed, those two approaches operate based on different concepts. LR is trained by adjusting weights and bias. TMs, on the other hand, relates words using propositional logic to represent a class. Employing propositional logic for knowledge representation provides rules rather than a mathematical computation. This crucial difference between a rule-based approach and regression methods is explored in [73]. One can analyze why a LR model assigns a particular class to an input by inspecting the weights and bias. However, assigning them meanings requires understanding of the mathematical computation that LR carries out. Since TM creates a list of patterns for a particular class based on the interaction of aspect words and the sentiment words in the context, its conjunctive clauses hold information of words in a rule-based form. It is well-known that evaluating a conjunctive clause is particularly easy for humans, making them natively interpretable and easier to explain than LR.

3.1.3 Enhancing Interpretable Clauses and Performance of TM

TM has shown promising results in above-mentioned NLP tasks, namely sentiment analysis, document classification, and Word Sense Disambiguation [20, 74]. To obtain human-level interpretability, legacy TM employs Boolean input features such as BOW. However, the BOW representation makes it difficult to use any pre-trained information, for instance, word2vec and GloVe word representations. This restriction has constrained the performance of TM compared

with DNNs in NLP. To reduce the performance gap, we propose a novel way of using pre-trained word representations for TM. The approach significantly enhances the performance and interpretability of TM. We achieve this by extracting semantically related words from pre-trained word representations as input features to the TM. Our experiments show that the accuracy of the proposed approach is significantly higher than the previous BOW-based TM, reaching the level of DNN-based models.

A key advantage of DNN models is distributed representation of words in a vector space. By using a single-layer neural network, Mikolov et al. introduced such a representation, allowing for relating words based on the inner product between word vectors [35]. One of the popular methods is skip-gram, an approach that learns word representations by predicting the context surrounding a word within a given window length. However, skip-gram has the disadvantage of not considering the co-occurrence statistics of the corpus. Later, Pennington et al. developed *GloVe* – a model that combines the advantages of local window-based methods and global matrix factorization [75]. The foundation for the above vector representation of words is the distributional hypothesis that states that "the word that occurs in the same contexts tend to have similar meanings" [76]. This means that in addition to forming a rich high-dimensional representation of words, words that are closer to each other in vector space tend to represent similar meanings. As such, vector representations have been used to enhance for instance information retrieval [77], name entity recognition [78], and parsing [79].

However, in the case of TM, such word representations cannot be directly employed because they consist of floating-point numbers. First, these numbers must be converted into a Boolean form for TM to use, which may result in information loss. Secondly, replacing the straightforward BOW of a TM with a large number of floating-point numbers in fine-grained Boolean form would impede interpretability. Here, we propose a novel pre-processing technique that evades the above challenges entirely by extracting additional features for the BOW. The additional features are found using the pre-trained distributed word representations to identify words that enrich the BOW, based on cosine similarity. In this way, TM can use the information from word representations for increasing performance, and at the same time retain the interpretability of the model.

3.1.3.1 Boosting TM BOW with Semantically Related Words

Here, we introduce our novel method for boosting the BOW of TM with semantically related words. The method is based on comparing pre-trained word representations using cosine similarity, leveraging distributed word representation. There are various distributional representations of words available, which are obtained from different corpora, using various techniques, such as word2vec, GloVe, and fastText. We here use GloVe because of its general applicability.

3.1.3.2 Input Feature Extraction from Distributed Word Representation

Distributed word representation does not necessarily derive word similarity based on synonyms but based on the words that appear in the same context. As such, the representation is essential for NLP because it captures the semantically interconnecting words. Our approach utilizes this property to expand the range of features that we can use in an interpretable manner in TM.

Consider a full vocabulary W of m words, $W = [w_1, w_2, w_3 \ldots, w_m]$. Further consider a particular sentence that is represented as a Boolean BOW $X = [x_1, x_2, x_3, \ldots, x_m]$. In a Boolean BOW, each element x_r, $r = 1, 2, 3, \ldots, m$, refers to a specific word w_r in the vocabulary W. The element x_r takes the value 1 if the corresponding word w_r is present in the sentence and the value 0 if the word is absent. Assume that n words are present in the sentence, i.e., n of the elements in X are 1-valued. Our strategy is to extract additional features from these by expanding them using cosine similarity. To this end, we use a GloVe embedding of each present word $w_r, r \in \{z | x_z = 1, z = 1, 2, 3 \ldots, m\}$. The embedding for word w_r is represented by vector $w_r^e \in \Re^d$, where d is the dimensionality of the embedding (typically varying from 25 to 300).

We next introduce two selection techniques to expand upon each word:

- Select the top k most similar words,
- Select words up to a fixed similarity angle $\cos(\theta) = \phi$.

For example, let us consider two contexts: "very good movie" and "excellent film, enjoyable". Figs. 3.11 and 3.12 list similar words showing the difference between top k words and words within angle $\cos(\theta)$, i.e., ϕ. In what follows, we will explain how these words are found.

excellent film, enjoyable					
excellent	similarity	film	similarity	enjoyable	similarity
good	0.703	movie	0.858	entertaining	0.688
superb	0.680	films	0.839	pleasurable	0.660
terrific	0.656	movies	0.716	exhilarating	0.647
quality	0.603	directed	0.683	fun	0.625
wonderful	0.584	documentary	0.656	exciting	0.619
best	0.550	starring	0.651	amusing	0.605
exceptional	0.550	cinema	0.637	satisfying	0.585
perfect	0.535	screenplay	0.632	engrossing	0.578
impressive	0.534	drama	0.622	enlightening	0.573
decent	0.532	comedy	0.615	informative	0.571

top k similar words

words within ϕ

Figure 3.11: Similar words for an example "excellent film, enjoyable" using 300d GloVe word representation.

3.1.3.3 Similar Words based on Top *k* Nearest Words

We first boost the Boolean BOW of the considered sentence by expanding X with $(k-1) \times n$ semantically related words. That is, we add $k-1$ new words for each of the n present words. We do this by identifying neighbouring words in the GloVe embedding space, using cosine similarity between the embedding vectors.

Consider the GloVe embedding vectors $W_G^e = [w_1^e, w_2^e, \ldots, w_m^e]$ of the full vocabulary W. For each word w_r from the sentence considered, the cosine similarity to each word w_t, $t = 1, 2, \ldots, m$, of the full vocabulary is given by Eq. (3.6),

top k similar words

very good movie					
very	similarity	good	similarity	movie	similarity
extremely	0.872	better	0.765	film	0.858
quite	0.858	really	0.736	movies	0.849
so	0.785	always	0.717	films	0.790
pretty	0.738	you	0.707	hollywood	0.679
too	0.731	well	0.704	starring	0.675
really	0.729	excellent	0.703	comedy	0.658
well	0.720	very	0.696	sequel	0.646
always	0.712	things	0.693	remake	0.624
especially	0.709	think	0.689	drama	0.608
but	0.707	way	0.683	actor	0.599

words within ϕ

Figure 3.12: Similar words for an example "very good movie" using 300d GloVe word representation.

$$\phi_r^t = \cos(w_r^e, w_t^e) = \frac{w_r^e \cdot w_t^e}{||w_r^e|| \cdot ||w_t^e||}. \tag{3.6}$$

Clearly, ϕ_r^t is the cosine similarity between w_r^e and w_t^e. By calculating the cosine similarity of w_r to the words in the vocabulary, we obtain m values: ϕ_r^t, $t = 1, 2, \ldots, m$. We arrange these values in a vector Φ_r:

$$\Phi_r = [\phi_r^1, \phi_r^2, \ldots, \phi_r^m]. \tag{3.7}$$

The k largest elements from Φ_r are then identified and their indices are stored in a new set A_r. Finally, a boosted BOW, referred to as X_{mod}, can be formed by assigning element x_t value 1 whenever one of the A_r contains t, and 0 otherwise:

$$X_{mod} = [x_1, x_2, x_3, \ldots, x_m], \tag{3.8}$$
$$x_t = \begin{cases} 1 & \exists r, t \in A_r \\ 0 & \nexists r, t \in A_r. \end{cases}$$

In addition, the vocabulary size for a particular task/dataset can be changed accordingly, which is usually less than m. Note that implementation-wise, the GloVe library provides the top k similar words of w_r without considering the word w_r itself, having similarity score 1. Hence, using the GloVe library, w_r must also be added to the boosted BOW.

3.1.3.4 Similar Words within Cosine Angle Threshold

Another approach to enrich the Boolean BOW of a sentence is thresholding the cosine angle. This is different from the first technique because the number of additional words extracted will vary rather than being fixed. Whereas the first approach always produces $k - 1$ new features for each given word, the cosine angle thresholding brings in all those words that are sufficiently similar. The cosine similarity threshold is given by $\phi = \cos(\theta)$, where θ is the threshold for vector angle, while ϕ is the corresponding similarity score.

As per Eq. (3.7), we obtain Φ_r, which consists of the similarity scores of the given word w_r in comparison to the m words in the vocabulary. Then, for each given word w_r, the indices of

those scores ϕ_r^t that are greater than or equal to ϕ ($\phi_r^t \geq \phi$) are stored in the set A_r. Similar to the first technique, the words in W with the indices in A_r are utilized to create X_{mod} as given by Eq. (3.8).

Model	R8	R52	MR
TF-IDF+LR	93.74	86.95	74.59
CNN-rand	94.02	85.37	74.98
CNN-non-static	95.71	87.59	77.75
LSTM	93.68	85.54	75.06
LSTM (pretrain)	96.09	90.48	77.33
Bi-LSTM	96.31	90.54	77.68
PV-DBOW	85.87	78.29	61.09
PV-DM	52.07	44.92	59.47
fastText	96.13	92.81	75.14
fastText (bigrams)	94.74	90.99	76.24
SWEM	95.32	92.94	76.65
LEAM	93.31	91.84	76.95
Graph-CNN-C	96.99	92.74	77.22
S^2GC	97.40	94.50	76.70
BERT	96.02	89.66	79.24
Lguided-BERT-1	97.49	94.26	81.03
Lguided-BERT-3	98.28	94.32	81.06
TM	96.16± 1.52	84.62± 1.8	75.14± 1.2
TM with k	97.50± 1.12	88.59± 1.2	77.51± 0.6
TM with ϕ	96.39± 1.0	89.14± 1.5	76.55± 0.9

Table 3.3: Comparison of feature extended TM with the state of the art for R8, R52 and MR. Reported accuracy of TM is the mean of last 50 epochs of 5 independent experiments with their standard deviation.

3.1.3.5 Distributed Word Representation in TM

Consider two contexts for sentiment classification: "Very good movie" and "Excellent film, enjoyable". Both contexts have different vocabularies but some of them are semantically related to each other. For example, "good" and "excellent" have similar semantics as well as "film" and "movie". Such semantics are not captured in the BOW-based input. However, as shown in Fig. D.4, adding words to the BOWs that are semantically related, as proposed in the previous section, makes distributed word representation available to the TM.

3.1.3.6 Results

In this subsection, we evaluate our TM-based solution with the input features enhanced by distributed word representation. We have selected various types of datasets to investigate how broadly our method is applicable: R8 and R52 of Reuters, Movie Review (MR), and TREC-6.

• **Reuters** 21578 dataset include two subsets: R52 and R8 (all-terms version). R8 is divided into 8 sections while there are 52 categories in R52.

•**MR** is a movie analysis dataset for binary sentiment classification with just one sentence per

review [80]. In this study, we used a training/test split from [81][3].
•**TREC-6** is a question classification dataset [82]. The task entails categorizing a query into six distinct categories (abbreviation, description, entity, human, location, numeric value).
Here we use GloVe pretrained word vector that is trained using CommonCrawl with the configuration of 42B tokens, 1.9M vocab, uncased, and 300d vectors. We here compare our proposed model with selected text classification- and embedding methods.We have selected representative techniques from various main approaches, both those that leverage similar kinds of pre-trained word embedding and those that only use BOW. The selected baselines are:

- **TF-IDF+LR**: This is a bag-of-words model employing Term Frequency-Inverse Document Frequency (TF-IDF) weighting. Logistic Regression is used as a softmax classifier.
- **CNN**: The CNN-baselines cover both initialization with random word embedding (CNN-rand) as well as initialization with pretrained word embedding (CNN-non-static) [83].
- **LSTM**: The LSTM model that we employ here is from [84], representing the entire text using the last hidden state. We tested this model with and without pre-trained word embeddings.
- **Bi-LSTM**: Bi-directional LSTMs are widely used for text classification. We compare our model with Bi-LSTM fed with pre-trained word embeddings.
- **PV-DBOW**: PV-DBOW is a paragraph vector model where the word order is ignored. Logistic Regression is used as a softmax classifier [85].
- **PV-DM**: PV-DM is also a paragraph vector model, however with word ordering taken into account. Logistic Regression is used as a softmax classifier [85]
- **fastText**: This baseline is a simple text classification technique that uses the average of the word embeddings provided by fastText as document embedding. The embedding is then fed to a linear classifier [86]. We evaluate both the use of uni-grams and bigrams.
- **SWEM** : SWEM applies simple pooling techniques over the word embeddings to obtain a document embedding [87].
- **Graph-CNN-C**: A graph CNN model uses convolutions over a word embedding similarity graph [88], employing a Chebyshev filter.
- S^2GC: This technique uses a modified Markov Diffusion Kernel to derive a variant of Graph Convolutional Network (GCN) [89].
- **LguidedLearn**: LguidedLearn is a label-guided learning framework for text classification. This technique is applied to BERT as well [90], which we use for comparison purposes here.
- **Feature Projection (FP)**: This is a novel approach to improve representation learning through feature projection. Existing features are projected into an orthogonal space. [91].

Model	TREC
LSTM	87.19
FP+LSTM	88.83
Transformer	87.33
FP+Transformer	89.51
BAE: BERT	97.6
TM [92]	87.20
TM	88.05± 1.52
TM with k	89.82± 1.18
TM with ϕ	90.04± 0.94

Table 3.4: Comparison of feature extended TM with the state of the art for TREC. Reported accuracy of TM is the mean of last 50 epochs of 5 independent experiments with their standard deviation.

From Table 3.3, we observe that the TM approaches that employ either of our feature extension techniques outperform several word embedding-based Logistic Regression approaches, such as PV-DBOW, PV-DM, and fastText. Similarly, the legacy TM outperforms sophisticated models like CNN and LSTM based on randomly initialized word embedding. Still, the legacy TM falls of other models when they are initialized by pre-trained word embeddings. By boosting the Boolean BOW with semantically similar features using our proposed technique, however, TM outperforms LSTM (pretrain) on the R8 dataset and performs similarly on R52 and MR. In addition, our proposed approach achieves quite similar performance compared with BERT, even though BERT has been pre-trained on a huge text corpus. However, it falls slightly short of sophisticated fine-tuned models like Lguided-BERT-1 and Lguided-BERT-3. Overall, our results show that our proposed feature extension technique for TMs significantly enhances accuracy, reaching the state-of-the-art accuracy. Importantly, this accuracy enhancement does not come at the cost of reduced interpretability, unlike DNNs. The state of the art for the TREC dataset is different from the other three datasets, hence we report the results separately in Table 3.4. These results clearly show that although the basic TM model does not outperform the recent DNN- and transformer-based models, the feature-boosted TM outperforms all of those models except understandably BAE:BERT [93].

3.1.4 Robust Text Classification against Spurious Correlations

In recent years, state-of-the-art NLP models have raised the bar for excellent performance on a variety of tasks. However, concerns are rising over their primitive sensitivity to distribution biases that reside in the training and testing data. This issue greatly impacts the performance of the models when exposed to out-of-distribution and counterfactual data. The root cause seems to be that many machine learning models are prone to learning shortcuts, modeling simple correlations rather than more fundamental and general relationships. As a result, such text classifiers tend to perform poorly when a human makes minor modifications to the data, which

[3]https://github.com/mnqu/PTE/tree/master/data/mr.

raises questions regarding their robustness. Here, we use a rule-based architecture called TM that learns both simple and complex correlations by ANDing features and their negations. As such, it generates explainable AND-rules using negated and non-negated reasoning. Here, we explore how non-negated reasoning can be more prone to distribution biases than negated reasoning. We further leverage this finding by adapting the TM architecture to mainly perform negated reasoning using the specificity parameter s. As a result, the AND-rules become robust to spurious correlations and can also correctly predict counterfactual data. Our empirical investigation of the model's robustness uses the specificity s to control the degree of negated reasoning.

Despite impressive advances in DNN architectures for NLP, their implementations still suffer from various challenges. One of the challenges is associated with DNN's capability of learning simple correlations and ignoring more complex ones [94]. This behavior of DNN becomes questionable when the simple correlation is spurious, absent from the test data, or occurs in an unfitting context. For instance, in the sentence *Nolan's films are always great mostly because of his excellent direction*, the influential word for predicting a positive sentiment should be "great" and "excellent" instead of "Nolan's" and "direction". However, due to the majority of samples consist of "Nolan having a great movie", it makes the classifier learn that "Nolan" corresponds to a positive sentiment word [95]. Similarly, a toxicity classifier learns that "gay" corresponds to a toxic comments [96] and a medical diagnosis classification system learns the disease associated with the patient ID [97]. The issue of spurious patterns also moderately impacts the out-of-distribution (OOD) generalization of models that are trained on independent identical distribution (IID) data, resulting in performance degradation when the distribution shifts.

Researchers recently have found that the decay in model performance, as well as social bias in NLP, appear out-of-domain due to sensitivity towards spurious signals. One of the solutions to deal with such vulnerability in NLP models is data augmentation with counterfactual samples [98], which can help the model learn real causal correlations between input and labels. For instance, a man-made counterfactual sample of the last example could be *Nolan's films are always* ***boring*** *mostly because of his* ***poor*** *direction.* Inserting such counterfactual data into the original training sets has shown to be beneficial for learning real causal correlation, thereby improving the robustness of the model [98]. However, augmentation with counterfactual data usually relies on a human-in-the-loop system to generate sentiment-flipped samples. For this process, humans are asked to make minimal and believable edits to generate counterfactual samples. Even though such an addition of data makes the model robust against spurious correlations, completing a human-in-the-loop process is costly and time-consuming.

The main reason behind the failure of DNNs on counterfactual data during inference is still unclear because of their BlackBox nature [52]. What they learn from the data that limits the models' robustness against different distribution samples is currently an open research question. Some researchers argue that the attention mechanism provides an explanation of DNN models, which assign soft weights to the input representations, and then extract highly weighted tokens as rationales [58]. However, these attention weights do not provide faithful explanations for classification [3, 59]. In addition, DNNs fail to use logical reasoning in various tasks. Logical reasoning is one of the most important prerequisites in NLP that supports various practical applications such as legal assistants, medical decision support, and personalized recommender systems. Due to these issues, DNNs have failed to demonstrate their robustness on counterfactual

data. On the other hand, a rule-based knowledge system is a powerful tool that offers logical reasoning because of its explainability. However, most rule-based systems rely on static rules that are handcrafted. Without learning capability, performance and generalization is limited. Keeping these two challenges in consideration, we employ a recent architecture called TM, which is an interpretable rule-based model that learns both simple and complex correlations via conjunctive clauses [1]. Unlike DNNs and simple rule-based architectures, TM learns rules with logical reasoning as a human does and it also offers a transparent and interpretable learning [99].

3.1.4.1 Learning Rule-based Clauses for Counterfactual Inference

The step-by-step explanation for the learning process of TM can be found in [100]. Here we explain briefly the learning of the rule-based clauses in TM for counterfactual inference via an example. Let the sentence "*Long, boring, blasphemous. Never have I been so glad to see ending credits roll.*" be the training sample that has negative sentiment. Each of the input words in the sentence is controlled by two TAs where *TA* controls non-negated literal such as "Long", and *TA'* controls the negated form such as "¬Long". The input that represents this particular sample is a sparse Boolean bag-of-words. All the vocabulary words that are present in the given sentence get the truth value 1, while those absent get the truth value 0. By explicitly representing missing words in vector form like $[0, 0, 0, 1, 0, \cdots, 0, 1, 0, 0, 0, 1]$, the representation becomes sparse. However, logically, such representation not only captures the presence of a particular word, but also equally well represents those words that are *not* present. This explicit bag of-words representation is ideal for TMs. This is because the TM can then pick informative negated features in the very first hundred iterations of learning using the selection parameter specificity s. We detail the role of s next.

In TM, each TA that controls a literal decides whether to take the action "Include" or "Exclude" based on the feedback it receives. There are two types of feedback: Type I Feedback and Type II Feedback [101]. Type I Feedback is activated when a given input feature is either correctly assigned to the target label (true positive) or mistakenly ignored (false negative), while Type II Feedback is activated when an input feature is wrongly assigned to the target label (false positive). The parameter s, $s \geq 1$, plays a very important role in the learning process, as it controls how strongly the model favours the action "Include". It also determines how many "fine-grained" sub-patterns the clauses will acquire. The greater the value of s, the more the TAs are encouraged to include literals in their clauses. Since s decides which literals take part in the clause for classification, it is vital to fine-tune it for reducing the vulnerability against spurious correlation. For the above-mentioned training example, when s is large, the states for the corresponding TAs in a clause after training are shown in Fig. 3.13. As seen, the high value of s enforces TA to include many literals in the clause, such as including "ending", "boring", "credits", "¬friendly", "¬good", and "¬like". Among the included literals, spurious correlations that do not carry sentiment information, such as "ending" and "credits", indeed influence the model's prediction on counterfactual data.

When we have a small s, as shown in Fig. 3.14, the number of included literals is reduced and the majority, if not all, of the included literals are in the negated form. One can see from the figure that the non-negated literals are now not enforced to be included in the clause. The states in *TA* for "ending", "boring", and "credits" have not reached to action "Include". Nevertheless,

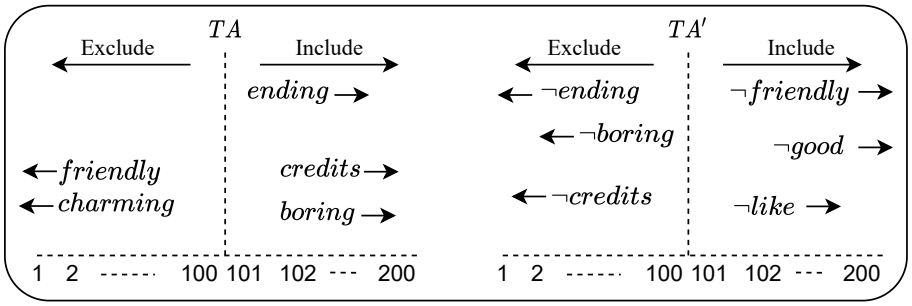


Figure 3.13: States of TAs when s is high for a particular clause.

TM still learns negated features easily in contrast to non-negated features due to sparse input representation thereby not affecting the states of "¬friendly", "¬good", and "¬like".

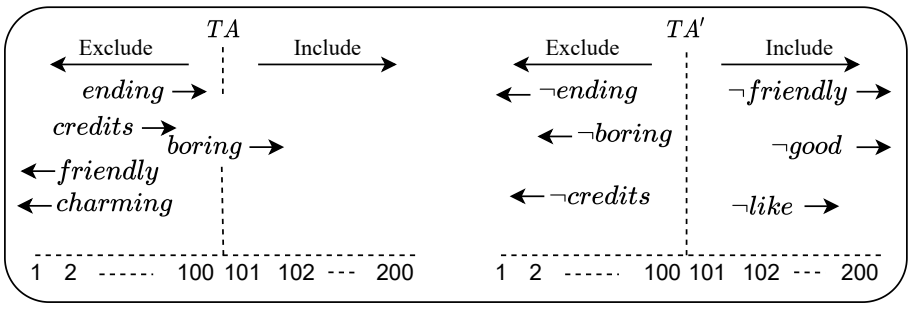


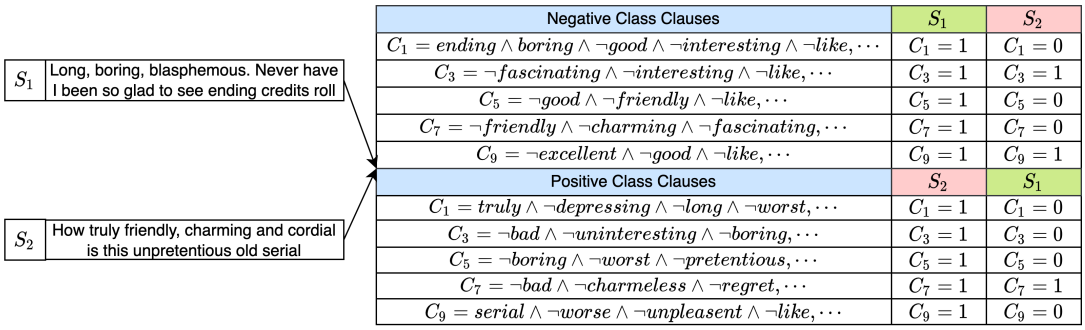
Figure 3.14: States of TAs when s is low for a particular clause.

3.1.4.2 Robustness against Counterfactual Sample

In this subsection, we will detail the reason why a trained TM model is robust and insusceptible to spurious correlations. Let us consider a model trained with a low value of $s = 2$ and two sentences with different sentiment labels: S_1 with positive and S_2 with negative sentiment. From Fig. 3.15, we can see the behavior of trained clauses for the negative class and the positive class for the original samples. The rule-based logic that is formulated by TM is in propositional form, ANDing several literals. The clause associated with propositional logic becomes 1 if an input satisfies the conjunction.

When context S_1 is received by the model, it correctly predicts negative sentiment because it triggers all the five clauses in the negative class, whereas only one clause for the positive class. Similarly, when S_2 is given, it predicts positive sentiment because the input triggers all five clauses in the positive class compared to only one clauses in the negative class.

Now consider two human generated counterfactual samples S_1^{cf} for S_1 and S_2^{cf} for S_2 as shown in Fig. 3.16. For S_1, the word "boring" is replaced by "fascinating"; "blasphemous" is replaced by "soulful"; and "glad" is replaced by "sad". Similarly, for S_2, the word "friendly" is replaced by "depressing"; "charming" is replaced by "charmless"; and "unpretentious" is replaced by "pretentious". This means that the labels for the corresponding counterfactual samples are now flipped. When S_1^{cf} is sent to the trained TM model with $s = 2$, it only triggers two clauses from the negative class and three clauses in the positive class. Similarly, when S_2^{cf} is given to the model, it triggers four clauses in the negative class but only one clause in the

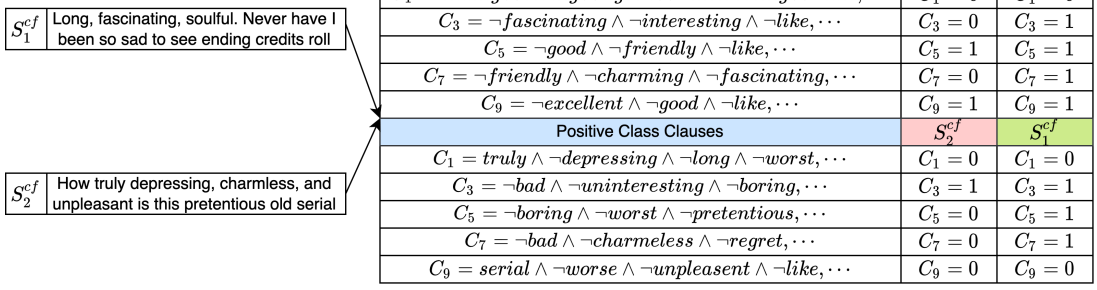


Negative Class Clauses	S_1	S_2
$C_1 = ending \wedge boring \wedge \neg good \wedge \neg interesting \wedge \neg like, \cdots$	$C_1 = 1$	$C_1 = 0$
$C_3 = \neg fascinating \wedge \neg interesting \wedge \neg like, \cdots$	$C_3 = 1$	$C_3 = 1$
$C_5 = \neg good \wedge \neg friendly \wedge \neg like, \cdots$	$C_5 = 1$	$C_5 = 0$
$C_7 = \neg friendly \wedge \neg charming \wedge \neg fascinating, \cdots$	$C_7 = 1$	$C_7 = 0$
$C_9 = \neg excellent \wedge \neg good \wedge \neg like, \cdots$	$C_9 = 1$	$C_9 = 1$
Positive Class Clauses	S_2	S_1
$C_1 = truly \wedge \neg depressing \wedge \neg long \wedge \neg worst, \cdots$	$C_1 = 1$	$C_1 = 0$
$C_3 = \neg bad \wedge \neg uninteresting \wedge \neg boring, \cdots$	$C_3 = 1$	$C_3 = 0$
$C_5 = \neg boring \wedge \neg worst \wedge \neg pretentious, \cdots$	$C_5 = 1$	$C_5 = 0$
$C_7 = \neg bad \wedge \neg charmeless \wedge \neg regret, \cdots$	$C_7 = 1$	$C_7 = 1$
$C_9 = serial \wedge \neg worse \wedge \neg unpleasent \wedge \neg like, \cdots$	$C_9 = 1$	$C_9 = 0$

Figure 3.15: Clause triggered by original samples S_1 and S_2 on both classes when $s = 2$.

positive class. Even though the probability of being in a class decreases due to the reduction in clause score, it still manages to predict such counterfactual samples correctly.

Since most of the entries in the sparse bag-of-word representation are zeros, the majority of literals presented in the clause will be in the negated form after a few iterations. With a comparatively small number of included literals due to the small s, the majority of clauses most likely become monotone in the negated form. Negated literals provide a more general form of the features that are not presented in a particular input sample thereby being less sensitive to spurious correlations as compared with the non-negated literals. We can clearly observe from Fig. 3.16 that the non-monotone clauses that have non-negated features are the ones that fail to capture counterfactual reasoning. This means monotonous clauses that have only negated features are more insusceptible to such modified data.



Negative Class Clauses	S_1^{cf}	S_2^{cf}
$C_1 = ending \wedge boring \wedge \neg good \wedge \neg interesting \wedge \neg like, \cdots$	$C_1 = 0$	$C_1 = 0$
$C_3 = \neg fascinating \wedge \neg interesting \wedge \neg like, \cdots$	$C_3 = 0$	$C_3 = 1$
$C_5 = \neg good \wedge \neg friendly \wedge \neg like, \cdots$	$C_5 = 1$	$C_5 = 1$
$C_7 = \neg friendly \wedge \neg charming \wedge \neg fascinating, \cdots$	$C_7 = 0$	$C_7 = 1$
$C_9 = \neg excellent \wedge \neg good \wedge \neg like, \cdots$	$C_9 = 1$	$C_9 = 1$
Positive Class Clauses	S_2^{cf}	S_1^{cf}
$C_1 = truly \wedge \neg depressing \wedge \neg long \wedge \neg worst, \cdots$	$C_1 = 0$	$C_1 = 0$
$C_3 = \neg bad \wedge \neg uninteresting \wedge \neg boring, \cdots$	$C_3 = 1$	$C_3 = 1$
$C_5 = \neg boring \wedge \neg worst \wedge \neg pretentious, \cdots$	$C_5 = 0$	$C_5 = 1$
$C_7 = \neg bad \wedge \neg charmeless \wedge \neg regret, \cdots$	$C_7 = 0$	$C_7 = 1$
$C_9 = serial \wedge \neg worse \wedge \neg unpleasent \wedge \neg like, \cdots$	$C_9 = 0$	$C_9 = 0$

Figure 3.16: Clauses triggered by counterfactual samples S_1^{cf} and S_2^{cf} on both classes when $s = 2$.

3.1.4.3 Results

In this subsection, we present experimental results for analyzing the performance of TM on counterfactual data. As we have already discussed the significance of s for inheriting robustness in the model, we experiment with different values of s on the dataset designed by [98]. This dataset has been developed using IMDB reviews that consist of $50k$ samples divided equally across train/test splits after removing 20% of reviews. Among them, $2.5k$ reviews have been split into training, validation, and testing of 1707, 245, and 488 respectively. These reviews are modified using Amazon's Mechanical Turk crowdsourcing so that the labels are flipped to

generate counterfactual samples [98]. In addition, to evaluate the out-of-domain performance of the proposed model, we used Amazon reviews [102] on data aggregated over six domains, i.e., *beauty, fashion, appliances, gift cards, magazines, and software*, SemEval Twitter sentiment analysis [103], and Yelp challenge dataset.

We used the original $1.7k$ samples as the training dataset to evaluate the robustness of the model on human-generated counterfactual test data of size 488. We also train the model using counterfactual data of size $1.7k$ and evaluate it on the original test samples of size 488. The performance of the model for various values of s is shown in Table 3.5. Other parameters of TM are the same for all the training datasets selected in the paper, with 3000 clauses per class and the threshold (T) value of 80×16. These parameters are selected by trial and error. For evaluating the behavior of s, we only validate on the test samples that are not from the same training data, and the complete performance evaluation is detailed later in the paper. Here, we use the features extension technique as the preprocessing as in [99]. As seen in Table 3.5, the accuracy of the model trained on original training samples achieves 72.1% on counterfactual test data when $s = 2$, and it decreases as s increases. Similarly, the accuracy of the model trained on counterfactual training samples achieves 65.20% when $s = 2$ and decreases as s increases. This indicates that lowering the value of s fine grains the pattern in the clause with negated literals, which confirms the robustness against counterfactual data as discussed earlier.

Training Data	s = 2		s = 3		s = 5		s = 10		s = 15		s = 20		s = 30		s = 50	
	Orig	CF	Orig	CF	Orig	CF	Orig	CF	Orig	CF	Orig	CF	Orig	CF	Orig	CF
Orig ($1.7k$)	-	**72.1**	-	71.1	-	68.87	-	65.53	-	64.73	-	60.64	-	58.63	-	54.31
CF ($1.7k$)	**65.20**	-	63.92	-	62.45	-	62.92	-	61.01	-	59.01	-	57.70	-	54.27	-

Table 3.5: Accuracy of TM on Counterfactual (CF) test data using Original (Orig) training samples and vice-versa for various values of s.

Training Data	SVM		NB		ELMo		Bi-LSTM		BERT		TM	
	Orig	CF	Orig	CF	Orig	CF	Orig	CF	Orig	CF	Orig	CF
Orig ($1.7k$)	80.0	51.0	74.9	47.3	81.9	66.7	79.3	55.7	87.4	82.2	85.65	73.56
											(84.30 $\pm$ 0.78)	(72.1 $\pm$ 0.40)
CF ($1.7k$)	58.3	91.2	50.9	88.7	63.8	82.0	62.5	89.1	80.4	90.8	65.98	92.20
											(65.20 $\pm$ 0.80)	(91.09 $\pm$ 0.55)
Orig ($19k$)	87.8	60.9	84.3	42.8	86.5	64.3	86.3	68.0	93.2	88.3	88.14	73.77
											(87.94 $\pm$ 0.16)	(72.46 $\pm$ 0.70)
Orig + CF ($3.4k$)	83.7	87.3	86.1	91.2	85.0	92.0	81.5	92.0	88.5	95.1	84.22	91.2
											(83.45 $\pm$ 0.42)	(89.95 $\pm$ 0.75)

Table 3.6: Experiment results of various models trained using Original and Counterfactual training dataset on their respective opposite test data. The upper results show the best reproducible accuracy and lower ones represent the mean and standard deviation of the last 50 epochs when running the model for five times.

To compare the performance of our model with the state of the art, extensive experiments have been carried out. Since $s = 2$ performs the best against counterfactual samples, we utilize this value for performance comparison. In addition to DNN based models, we also include typical interpretable linear models in our comparison. The models are mainly taken from [98], as:

- **Standard Methods:** We train linear standard model such as SVM and Naive Bayes (NB) for sentiment classification using "scikit-learn" [104].
- **Bi-LSTM:** For training Bi-LSTM, Kaushik et al. [98] restricted vocabulary of $20k$, replacing out-of-vocabulary as *UNK* tokens. The model consists of bidirectional LSTM with hidden dimesion 50, recurrent dropout 0.5, and global max pooling following the embedding layer.
- **ELMo:** Kaushik et al. [98] computed contextualized word representation (ELMo) using character based word representation and bidirection LSTM [63] using weighted sum of representation of 1024 dimensions.
- **BERT:** Kaushik et al. [98] used an off-the-shelf uncased BERT Base model to fine tune each task. In order to consider the BERT's sub tokenization, token length is set at 350 and trained for 20 epochs.

As we can see from Table 3.6, when the original data is used as the training samples, SVM's accuracy on CF test data drops to 51.0% compared with that of the original test data, i.e., 80%. A similar trend is observed for NB, Bi-LSTM, and ELMo. Interestingly, the performance of BERT suffers less perhaps due to the benefit of large pretrained information. However, disregarding the pre-trained language model of BERT, our proposed TM reaches 73.56% and outperforms all of the remaining models including 66.7% of ELMo. In the case of CF data as the training samples, the accuracy on original test samples by previous best model ELMo is 63.8% except BERT. Again, our proposed TM model outperforms all of them except BERT, achieving 65.98%. Although the main aim of this part of the thesis is to evaluate TM on different/counterfactual distribution and it is not necessary to augment both original and CF data, we still show the performance using augmented data as well as the remaining IMDB data of size $19k$ as training samples, and it can be seen that the performance of TM is on par with the other models.

3.2 Interpretable Text Classification Using Neural Network

In this section, we tackle the problem of interpretability in the DNN. DNN, being a BlackBox in nature, still finds a way to explain its model using the attention weights on top of various language representation layers such as RNN, LSTM, or GRU. These attention weights assign soft weights to the input rationales that can be used as the interpretation of the model. As good as it looks, it only makes sense in the scenario when the input text is either BOW or Sequential BOW. However, for some peculiar tasks such as ABSA, there is a need for positional embedding in the input layer. Even though LSTM/GRU are well known to capture the sequential information of the context, they still fail to concentrate on the target word and its nearby context. Hence, the majority of the task needs an additional positional embedding integrated into the initial layer of DNN. As a result, it not only increases the trainable parameters of the model but also disorients the interpretation of the model because the input rationales now consist of a combination of input and positional embedding. Since the attention weights give the impact of each input rationales, using the combined input rationales in the interpretation seems ambiguous. Hence the main focus of this work is to remove the positional embedding with simple architecture and yet maintain the

state-of-the-art performance so that attention weight justifies only input rationales (Subsection 3.2.1). In addition, we use the information from the previous TM-based model to enhance the explainability of the DNN-based model. The intuition behind this task is that the attention layer has been argued for being explainable because of its dynamic distinct attention distribution. Hence we use the information from interpretable TM and enrich the input representation layer of DNN for attention explanation (Subsection 3.2.2).

3.2.1 Position Dependent Text Classification without Positional Embedding

Positional embedding, in most cases, depends on the distance between the aspect word and the remaining words in the context, known as the position index sequence. However, these techniques usually employ both complex preprocessing approaches with additional trainable positional embedding and complex architectures to obtain state-of-the-art performance. Here, we simplify preprocessing by including polarity lexicon replacement and masking techniques that carry the information of the aspect word's position and eliminate the positional embedding. We then adopt a novel and concise architecture using two Bidirectional GRU along with an attention layer to classify the aspect based on its context words.

Various neural network architectures, from simple to complex ones, have been developed for position-aware sentiment classifications with a focus on the aspect word [61, 105]. A position encoding vector developed in [106] has been a popular choice for embedding positional information in LSTM based models. There the position index of the surrounding words is represented by the relative distance to the aspect word. Such position embedding creates a probability distribution among the context that is then embedded along with the word embedding of each word for classification of sentiments. However, a sophisticated neural network architecture is required for good performance because of the lack of sentiment lexicon knowledge with the integration of positional embedding [107]. Even a slight increment of accuracy in ABSA task usually requires a more complex architecture [108].

Here, we propose a very simple preprocessing of ABSA task by using sentiment lexicons and a masking technique that removes complex positional embedding thereby requiring a very straightforward architecture to obtain the state-of-the-art performance. As we know, human being usually makes sentiment classification of a particular word based on the surrounding words. Besides, human being, most probably, understands the meaning of each word and the sentiment associated with it as a priori. On the contrary, a neural network does not have this inbuilt knowledge. Even though various pre-trained word embedding captures the semantic relationship among the words, they are usually complicated. Therefore, it is important to find an efficient way to offer the model necessary knowledge, as priori, as much as possible. To give the model extra knowledge about sentiment in a simple way, we employ Opinion Lexicon [64] that has a list of positive and negative sentiment words. We use these lexicons and replace all the possible positive words with the "positive" tokens and negative words with "negative" tokens. The words that are not in the Opinion Lexicon will be left as they are. Additionally, to avoid complex positional embedding, the aspect word is masked with a common token, making it a Masked Aspect Embedding and the original sentence as Sentence Embedding. Then, we adopt the Attention-based BiGRU to train both the input to classify the sentiment of the masked

aspect word. To evaluate the performance of the proposed methodology, we experiment with all available restaurant and laptop datasets of the ABSA task [109, 110, 111]. The numerical results show that the proposed scheme obtains either similar or higher accuracy compared with the state-of-the-art solutions that use positional embedding and a complex architecture.

3.2.1.1 Preprocessing

As mentioned earlier, the sentiment of aspect word highly depends on the context words surrounding it. Human beings can understand the meaning and the sentiment of context words that describe the aspect word. That is why human being can easily extract the sentiment of any particular word. On the contrary, a neural network does not have the knowledge that shows the semantic and syntactic relationship between words. Word2vec [35] and GloVe embedding [75] capture the semantic relationship between the words but they are still far from human efficiency. Hence, we try to reduce the gap between the human knowledge and word embedding by making the semantically related word as the same token. To simplify the problem so that the neural networks can solve it better, we replace the sentiment-carrying words with the tag "positive" or "negative" based on Opinion Lexicon [64] as shown in Fig. 3.17. Opinion Lexicon is a list of English words with positive or negative sentiment. Such use of external resource in preprocessing not only integrates sentiment knowledge but also reduces the vocabulary size that is a substantial concern by itself in NLP [112]. Altogether this process replaces around 550 words with the token "positive" and "negative".

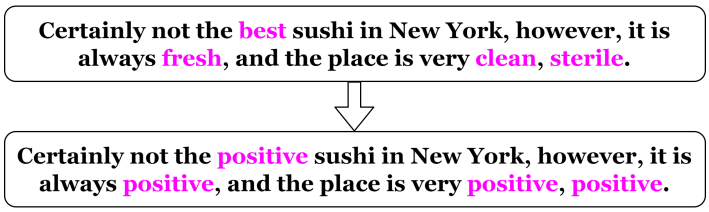


Figure 3.17: Replacement of sentiment carrying word with a common tag using Opinion Lexicon.

Another important aspect of the preprocessing is to embed the position information of the aspect word. Traditional positional embedding considers the relative distance between aspect words and the context words in a sentence. Such embedding creates a probability distribution over the sentence with respect to the aspect word. However, such position embedding integrated with the input sentence is often initialized with trainable weights that increase the complexity of the model [113]. To mitigate this problem, we propose a simple masking technique that is based on the pattern learning behavior of the neural network. Usually, an ABSA task has two inputs: Sentence Embedding carrying the original sentence where position information is integrated and Aspect Embedding carrying aspect or aspect word. Here, we modify Aspect Embedding as Masked Aspect Embedding that carries the sentence with the aspect word masked by a common tag (here we call the common tag as "MASK"). We propose this preprocessing to remove the positional embedding required by Sentence Embedding. The modification between existing positional embedding and proposed masking technique is shown in Fig. 3.18, where $pos(w^1)$ is the relative positional encoding of the first word with respect to the aspect word and the total number of words in the sentence is n. We hypothesize that the masked token is a common token

present in every sample at a different location, which creates a positional pattern. Since any machine learning model tries to capture the repetitive patterns, we hypothesize that the model will pick the masked token and its necessary context words around it to classify the sentiment. In all brevity, we propose a model that learns sentiment patterns for the position of the masked token. The overall preprocessed input is shown in Fig. 3.19.

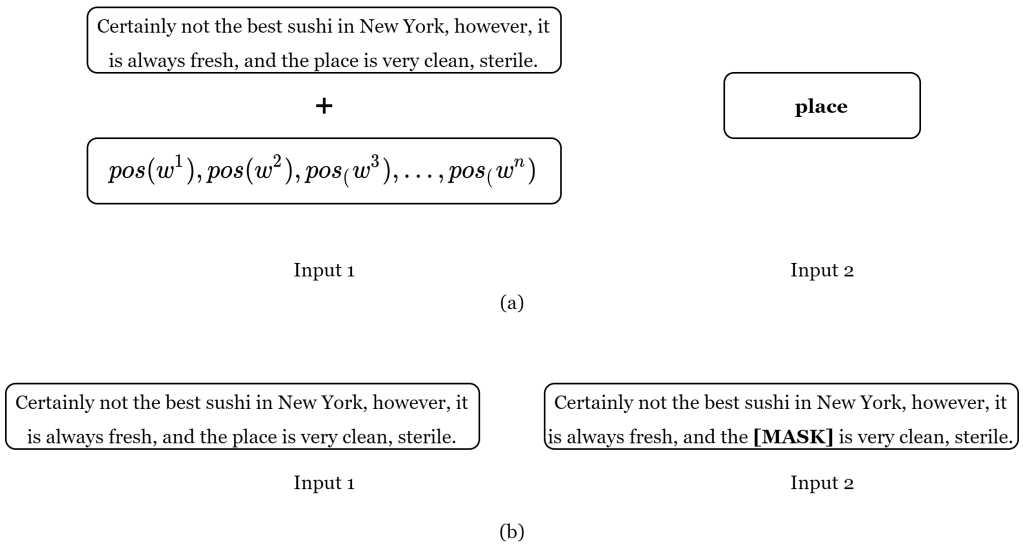


Figure 3.18: (a) Existing approach of position embedding. (b) Proposed masking technique to learn pattern for the position.

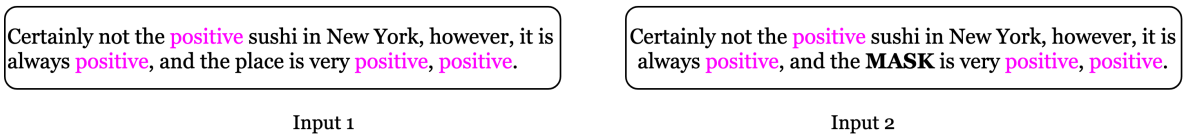


Figure 3.19: Proposed preprocessed input.

3.2.1.2 Architecture description

The overall architecture of proposed model is shown in Fig. 3.20, which consists 3 sections: Input Embedding, Bi-GRU, and Attention Layer. As the input embedding has been explained in the preprosessing part, we will focus on the latter two in the following paragraphs.

RNNs [114] have been the baseline for NLP recently, where the internal states are utilized to process data sequentially. However, RNNs have certain limitations that lead to the development of their variants, such as LSTM and GRU. Here, we have explored both LSTM and GRU for sequencing modeling. Since we aim at developing a very concise and efficient model, we opt for GRU in our final architecture. The GRU controls the flow of information like the LSTM unit without employing a memory unit, which makes it more efficient with uncompromised performance compared to LSTM [115]. In addition, GRU mitigates the problem of vanishing gradients and gradient explosions in vanilla RNN.

Our proposed model consists of two Attention-based Bi-GRUs: GRU_1 for Sentence Embedding and GRU_2 for Masked Aspect Embedding. Both of them are identical in architecture

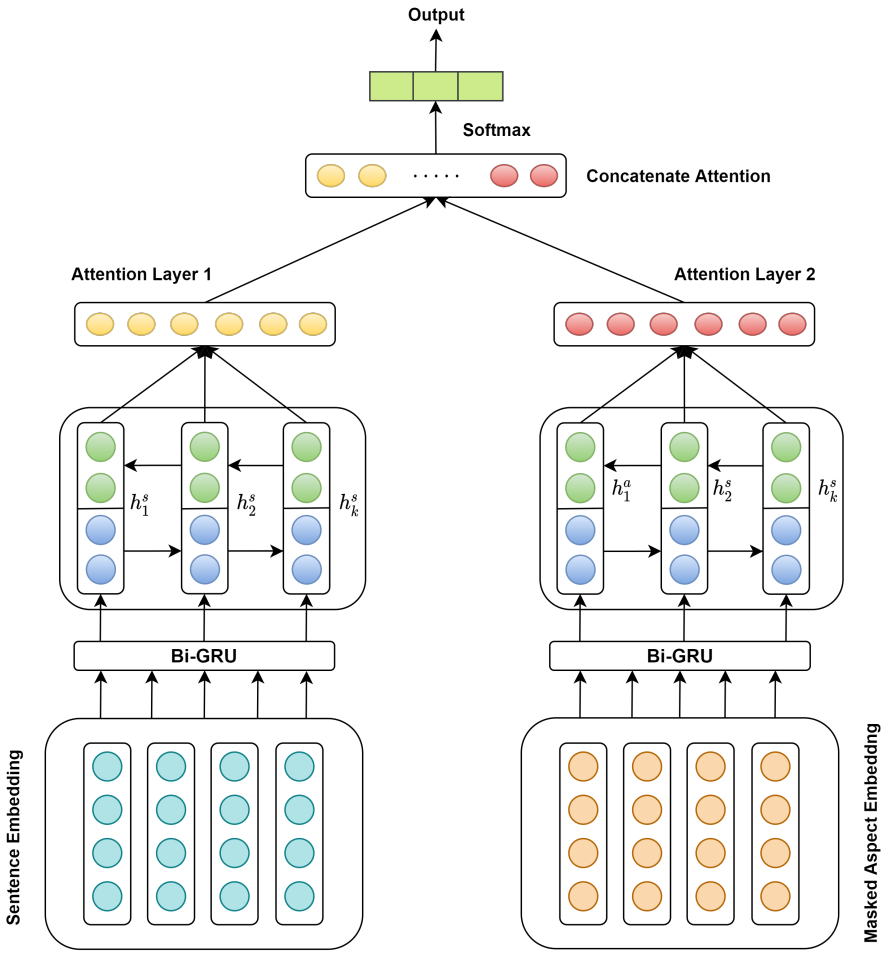


Figure 3.20: Proposed attention-based Bi-GRU architecture.

that has similar learning pattern with the same hyperparameters. The only difference is how the preprocessed input data is passed to these two separate Bi-GRUs. We assumed that GRU_2 captures the position of the masked token. Additionally, attention layer 2 gives the highest weightage to the masked token wherever it presents in the sentence. Similarly, GRU_1 is supposed to capture the context features from Sentence Embedding with attention layer 1, assigning higher weightage to the necessary context words. This hypothesis seems quite similar to how human being operates to understand the aspect-based sentiment.

Define $X = [x_1, x_2, x_3, \cdots, x_k]$ the Sentence Embedding (or Input 1), where k is the padded length of the sentence embedding to the forward layer of the GRU. There are two kinds of gates in GRU: the update gate and the reset gate. The update gate decides the amount of past information that needs to be brought into the current state and how much the new information is added. On the other hand, the reset gate takes care of how much information about the previous steps is written into the current candidate state h_t. Here, h_t is the output of the GRU at time step t and z_t represents the update gate. At a particular time step t, the new state h_t is given by:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t, \tag{3.9}$$

where $\odot$ is the element-wise multiplication and $\hat{h}_t$ is candidate activation. To update z_t, we have

$$z_t = \sigma \left(W_{z_t} x_t + U_{z_t} h_{t-1} + b_{z_t} \right). \tag{3.10}$$

Here, x_t is the word of the sentence at time step t that is plugged into the network unit and it is multiplied with its own weight W_{z_t}. Similarly, h_{t-1} holds the information of the previous unit and is multiplied with its own weight U_{z_t} and b_{z_t} is the bias associated with update state. The current state h_t can be updated using reset gate r_t by

$$\hat{h}_t = \tanh \left(W_h x_t + r_t \odot (U_h h_t) + b_h \right), \tag{3.11}$$

where W_h and U_h are weights associated with the candidate activation along with bais b_h. At r_t, the candidate state of step t can get the information of input x_t and the status of h_{t-1} of step $t-1$. The update function of r_t is given by

$$r_t = \sigma \left(W_{r_t} x_t + U_{r_t} h_{t-1} + b_{r_t} \right), \tag{3.12}$$

where W_{r_t} and U_{r_t} are the weights associated with the reset state and b_{r_t} is the bias.
The Bi-GRU contains the forward GRU layer $(\overrightarrow{h_t})$ that reads the input sentence from step 0 to t and the backward GRU $(\overleftarrow{h_t})$.

$$\overrightarrow{h_t} = \overrightarrow{GRU}(x_t), \ t \in [1, T], \tag{3.13}$$

$$\overleftarrow{h_t} = \overleftarrow{GRU}(x_t), \ t \in [T, 1], \tag{3.14}$$

$$h_t = \left[\overrightarrow{h_t}, \overleftarrow{h_t} \right]. \tag{3.15}$$

As we know that not all the words in the context have equal contribution for sentiment classification, an attention layer is assigned to prioritize important words in the context. Attention

layer 1 is wrapped on top of GRU_1 to learn a weight α_t^1 for each hidden state h_t obtained at time step t. Since there are k inputs in the padded sequences, time step t will be from 1 to k. The weighting vector for attention layer 1, $\alpha_t^1 = [\alpha_1^1, \alpha_2^1, \alpha_3^1, \cdots, \alpha_k^1]$ is calculated based on the output sequence $H = [h_1, h_2, h_3, \cdots, h_k]$. The attention vector s_1 for attention layer 1 is calculated based on the weighted sum of these hidden states, as:

$$s_1 = \sum_{t=1}^{k} \left(\alpha_t^1 h_t\right), \tag{3.16}$$

where the weighted parameter α_t^1 is calculated by:

$$\alpha_t^1 = \frac{\exp\left(u_t^T u_w\right)}{\sum_t \exp\left(u_t^T u_w\right)}, \tag{3.17}$$

and $u_t = \tanh\left(W_w h_t + b_w\right)$. Here W_w and h_t are the weight matrices and b_w represents the bias. The parameter u_w represents context vector that is different at each step, which is randomly initialized and learned jointly during the training process.

Similarly, the attention layer 2 is wrapped on top of GRU_2 for assigning weightage to the masked token based on its position. The attention vector s_2 for attention layer 2 is given by:

$$s_2 = \sum_{t=1}^{k} \left(\alpha_t^2 h_t\right). \tag{3.18}$$

Finally, both of the attention layers are concatenated

$$s = Concatenate\left(s_1, s_2\right). \tag{3.19}$$

The concatenated layer is then sent to a fully connected layer and the softmax function generates a probability over c class labels.

3.2.1.3 Results

Our experiments are conducted on four publicly available ABSA datasets. Each sample of every dataset is a single sentence of a product review with aspect word and the corresponding sentiment label associated. While the datasets are given in the laptop domain by SemEval 2015 and SemEval 2016, they only contain the aspect category without the aspect word. The "null" aspect terms are excluded from the datasets, and the "dispute" or more than one sentiment labels are also excluded from the aspect terms in the analysis. The remaining sentences contain at least one aspect of the word with a {positive, neutral, negative} sentiment tag. The numerical details of the datasets are shown in Table 3.7.

To evaluate the performance of the proposed model on ABSA datasets, we consider the following approaches as comparative models. These models are proper baselines for ABSA and they are as close as possible to this work. Additionally, we have used the models that have exactly been evaluated in these specific four datasets of ABSA.

- **Feature+SVM** extracts n-gram as a feature, parse feature, and lexicon features to train the classifier [116].

Dataset	Train			Test		
	Pos	Neu	Neg	Pos	Neu	Neg
Rest 14	2164	637	807	728	196	196
Lap 14	994	464	870	341	169	128
Rest 15	948	34	269	432	38	257
Rest 16	1289	63	457	474	29	123

Table 3.7: Details of ABSA datasets.

- **ContextAvg** averages the word embedding to form a context embedding and then it is feed to the softmax function along with aspect vector [65].
- **LSTM** uses the last hidden vector information of the LSTM as a sentence representation for classifying aspect level sentiment [28].
- **TD_LSTM** utilizes two LSTMs to learn the language model from left and right contexts of the aspect respectively [65].
- **ATAE_BiLSTM** This model is similar to our approach, which is an Attention-based LSTM architecture with Aspect Embedding. It computes the aspect-specific weighted score of each word according to the representation of the aspect. The sums of the LSTM hidden outputs based on the attention weights are utilized to generate the sentence representation for ABSA classification [66].
- **IAN** is an Interactive Attention Network model that calculates the attention weights of the word in sentiment and aspect interactively to generate aspect and sentence representations [55].
- **MemNet** integrates the content and the position of the aspect word into deep neural network [65].
- **RAM** is a multi-layer architecture where each layer consists of attention-based aggregation of word features. A GRU cell is used to learn the sentence representation [67].
- **Ont+LCR-Rot-hop** uses a lexicon domain ontology and a rotatory attention mechanism to predict the sentiment of the aspect word [117].
- **PBAN** is a position-aware bidirectional attention network on bidirectional GRU. It also uses the mean pool and dot product to embed the position information of aspect word into sentence representation. It performs on par with the state of the art [61].
- **PAHT** is a position-aware hierarchical transfer model that models the position information from multiple levels to enhance the ABSA performance by transferring hierarchical knowledge from the resource-rich sentence-level sentiment classification (SSC) dataset [62].
- **MTKFN** is a Multi-source Textual Knowledge Fusing Network that incorporates knowledge from multiple sources to enhance the performance of ABSA. It uses pre-trained layers to extract contextual features and predicts the sentiment polarities. Additionally, it uses the information of conjunctions that captures the relationship between clauses and provides additional sentiment features [118].

- **BERTADA-base** is further trained on a domain-specific dataset and evaluated on the test set from the same domain [119].
- **XLNetADA-base** model is like BERTADA-base except for adopting XLNet [119].

Dataset	Restaurant 14		Laptop 14		Restaurant 15		Restaurant 16	
	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1
Majority	65.00	26.26	53.45	23.22	54.74	23.58	72.36	29.99
Feature+SVM	80.16	-	70.49	-	-	-	-	-
ContextAvg	71.53	58.02	61.59	53.92	73.79	47.43	79.87	55.68
LSTM	74.49	59.32	66.51	59.44	75.40	53.30	80.67	54.53
TD_LSTM	78.00	68.43	71.83	68.43	76.39	58.70	82.16	54.21
ATAE_BiLSTM	77.63	64.97	69.61	63.04	77.40	54.29	86.01	60.32
IAN	77.35	64.77	69.58	61.08	78.07	51.89	85.44	56.51
MemNet	78.16	65.83	70.33	64.09	77.89	59.52	83.04	57.91
RAM	78.48	68.54	72.08	68.43	79.98	60.57	83.88	62.14
Ont+LCR-Rot-hop	-	-	-	-	80.60	-	88.00	-
PBAN	81.16	-	74.12	-	-	-	-	-
PAHT	79.29	68.49	75.71	69.55	80.86	60.76	85.81	67.11
MTKFN	79.47	68.08	73.43	69.12	80.67	58.38	88.28	66.15
BERTADA-base	84.92	76.93	77.69	72.60	-	-	-	-
XLNetADA-base	85.84	78.35	79.89	77.78	-	-	-	-
Proposed Model	81.37	72.06	75.39	70.50	80.88	62.48	89.30	66.93

Table 3.8: The state-of-the-art performance of ABSA on four datasets.

The comparison of our proposed model with recent similar studies is shown in Table 3.8. These state-of-the-art studies are selected for comparison because they mostly depend on positional embedding as well as complex architecture, which are more relevant to our proposed model.

Among the baseline models that depend on language modeling, LSTM performs poorly on all four datasets. Above this lies ATAE_BiLSTM that has poor performance considering the fact that it utilizes the attention mechanism to model the aspect word. However, TD_LSTM performs better than the two models mentioned above because it considers both the left and the right context as an aspect rather than the entire sentence. Similarly, MemNet performs better than IAN but it is not as good as RAM, because it does not use multiple attention mechanisms. Moreover, PAHT and MTKFN that utilize the hierarchical transfer model and external knowledge fusion respectively exhibit quite similar results as both of them are position-aware models. However, our proposed model surpasses both of these recent models with a significant margin using a much simpler architecture.

Among PBAN, PAHT and MTKFN, PBAN has higher accuracy than PAHT and MTKFN, which is a quite similar model to our proposed architecture. Hence, we focus on PBAN more than other listed models for comparison. PBAN that adopts two BiGRUs still falls behind in performance compared with our model. As shown in Table 3.8, PBAN achieves 81.16% accuracy on restaurant 14 and 74.12% on laptop 14 dataset. However, it uses traditional embedding with a more complex model than ours. On the other hand, by employing the Opinion Lexicon and masked aspect embedding, our model gives 81.37% and 75.39% accuracy on restaurant 14 and laptop 14 datasets respectively. Additionally, the macro-F1 score also outperforms the

above-mentioned models with a significant margin except for the restaurant 16 dataset where the macro-F1 score is slightly below PAHT.

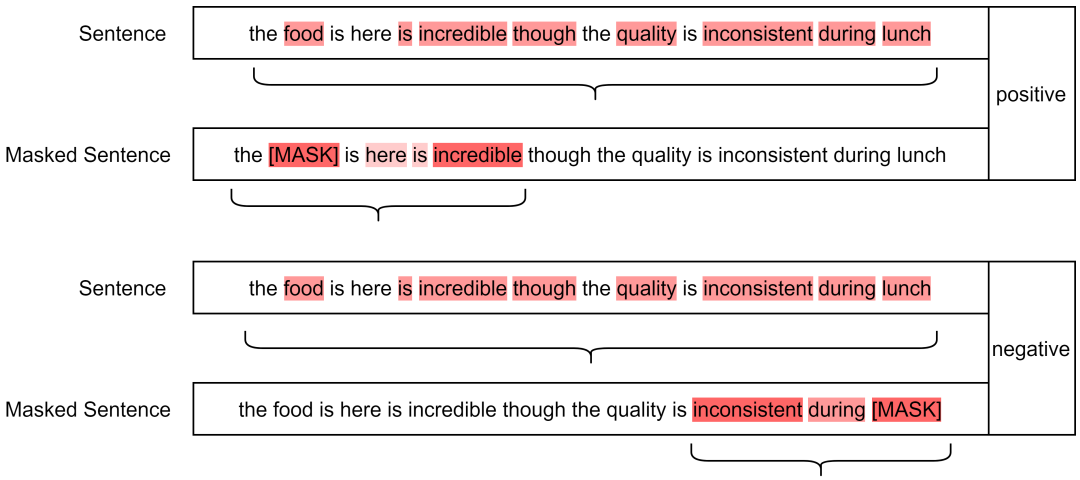


Figure 3.21: Visualization of two typical examples. The red color represents the attentive weight of the word. A deeper color indicates a larger weight value.

To have a detailed insight into why our proposed model performs better than the baselines with a straightforward architecture, we sample two examples from the restaurant 16 dataset and visualize attention heatmaps based on the trained model. Here we have two inputs for two separate BiGRUs: one being the sentence itself and the other being the masked sentence. As we have already discussed, the aspect words are masked with a common token, say "**MASK**". From Fig. 3.21, we can see that the sentiment of the aspect word "food" is classified as positive sentiment from our model. The original sentence is fed to GRU_1 that has an attention layer 1 and the Masked sentence is fed to GRU_2 that has an attention layer 2. Here, attention layer 1 captures the context words throughout the sentence whereas attention layer 2 pays high attention to the masked token with weight narrowing down to important context words as shown in Fig. 3.21. In the first example, attention layer 2 shifts the attention weightage towards the masked token "food" (the first half of the sentence) that highly depends on the context "incredible" for positive sentiment. For the second example, attention layer 2 shifts the attention weightage towards the masked token "lunch" (the second half of the sentence) whose sentiment depends on context "inconsistent" for negative sentiment. This is a clear validation of our hypothesis that the masked token will hold the position information without using any additional trainable positional embedding. In all brevity, attention layer 1 assigns weightage to words in the sentence, and attention layer 2 narrows down the weightage from these selected words to important context words, carrying sentiment of the aspect word.

3.2.2 Enhancing Attention's Explanation Using TM

Based on the study so far, we have observed that the TM-based models are interpretable with decent performance. On the other hand, DNN-based models are better in terms of accuracy but their explainability based on attention weights tends to differ with distinct attention distribution. This is due to the fact that language representation layers are initialized by pre-trained word embedding that is not context-dependent. Such a lack of context-dependent knowledge in the

initial layer makes it difficult for the model to concentrate on the important aspects of input. Usually, it does not impact the performance of the model significantly, but the explainability differs from human understanding. Hence, we propose an ensemble method to use logic-based information from the TM to embed it into the initial representation layer in the neural network to enhance the model in terms of explainability. We obtain the global clause score for each word in the vocabulary and feed it into the neural network layer as context-dependent information.

The attention mechanism is a prominent technique among current explainability approaches that identify essential sections of the input for the prediction job by offering a distribution across attended-to-input units [58]. Many NLP tasks, such as text categorization, question answering, and entity identification, have shown outstanding results using attention-based models [58, 120, 121]. In particular, in many NLP systems, the self-attention mechanism that underpins the Transformer design has played a key role [30, 122]. Despite this, recent research has revealed that learned attention weights are frequently unrelated to the relevance of input components as judged by various explainability approaches [123], and that alternative attention distributions can provide identical predictions [124, 4].

One efficient way to deal with the above-mentioned problem is to use prerequisite knowledge to enhance the input layer for better interpretation. Integrating human rationales as supplementary supervision information for attention learning is a promising way to enhance the explainability of attention-based models. Human rationales have previously been found to be useful inputs for increasing model performance and discovering explainable input in model prediction [125, 126]. However, obtaining such human rationales is an expensive and time-consuming process. Hence, to make it easier and more efficient, we use a logic-based model TM that mimics human-level understanding to generate prerequisite information to initialize the input layer of the neural network. Since TM can be explained by logic and rules, the information it provides can be easily explained to make the attention layer focus on important input tokens.

3.2.2.1 Clause Score from TM Architecture

In regards to NLP, TM heavily relies on the Boolean BOW given by $X = [x_1, x_2, x_3, \ldots, x_n]$. Let l be the number of clauses that represent each class of the TM, covering q classes altogether. Then, the overall learning problem is solved using $l \times q$ clauses. Each clause C_i^j, $1 \leq j \leq q$, $1 \leq i \leq l$ of the TM is given by :

$$C_i^j = \left(\bigwedge_{k \in I_i^j} x_k \right) \wedge \left(\bigwedge_{k \in \bar{I}_i^j} \neg x_k \right), \tag{3.20}$$

where I_i^j and $\bar{I}_i^j$ are non-overlapping subgroup of the input variable indices, $I_j^i, \bar{I}_j^i \subseteq \{1, \ldots, m\}$, $I_j^i \cap \bar{I}_j^i = \emptyset$. The subgroup decides that which of the input variables to participate in the clause, and whether they are in the original form or the negated. The indices of input features in I_j^i represent the literals that are included as original form of the literals, while the indices of input features in $\bar{I}_j^i$ correspond to the negated ones. Among the q clauses of each class, clauses that are indexed with odd number are assigned positive polarity (+) whereas those with even indexed are assigned negative polarity ($-$). The clauses with positive polarity vote for the true target class and those with negative polarity vote against it. A summation operator aggregates the votes by

subtracting the total number of negative votes from positive votes, as shown in Eq. (3.21).

$$f^j(X) = \Sigma_{i=1,3,\ldots}^{l-1} C_i^j(X) - \Sigma_{i=2,4,\ldots}^{l} C_i^j(X). \tag{3.21}$$

For q number of classes, the predicted output y is given by the argmax operator which classifies the input features based on the highest sum of votes obtained, as shown in Eq. (3.22).

$$\hat{y} = \text{argmax}_j \left(f^j(X) \right). \tag{3.22}$$

Once the model is trained with a particular dataset, we can explore the clauses that hold information of combination of literals in propositional form. Such information is humanly interpretable and can be used for downstream applications of NLP. Here, we explore the weightage of each word in the model. We pass each word in the vocabulary into the TM and obtain the clause score. The clause score is calculated by:

$$SC_{x_k} = |f^{\kappa=tp}(X_{x_k=1}) - \Sigma f^{\kappa=fp}(X_{x_k=1})|. \tag{3.23}$$

Here *tp* refers to true prediction, *fp* refers to false prediction, $|.|$ refers to the absolute value, and $k = 1, 2, \ldots, n$, where n is the number of vocabularies. We then create the input map for each input sentence with the score obtained for each word, which will be fed to the neural network's initial embedding layer.

3.2.2.2 Attention-based Neural Network

This subsection extends the attention-based neural network for text classification, where we use conventional Bi-GRU as the language representation layer and attention on top of it.

Because of its linked hidden layers, where the internal states are used to process data in a sequential fashion, RNNs [114] have lately become the standard for NLP. RNNs, on the other hand, have several drawbacks that have led to the creation of versions like LSTM and GRU. The GRU, like the LSTM unit, regulates the flow of information without using a memory unit, making it more efficient with near-lossless performance [127]. GRU also overcomes the issue of vanishing gradients and gradient explosions in vanilla RNN. Our selected model consists of a Bi-GRU layer on top of embedding layer initialized with GloVe embedding. This layer consists of an attention layer on top of Bi-GRU. The overall architecture of proposed model is shown in Fig. 3.22.

Consider the sentence "This is a wonderful movie", which is fed to the embedding layer initialized by GloVe embedding as shown in Fig. 3.22. On the other hand, we obtain the clause score for each word in the sentence and feed to the embedding layer to match the dimension of input sentence embedding. Then both the embedding layer is passed to multiplication layer, where both are multiplied element wise. The output of the multiplication layer is then fed to the Bi-GRU having multiple hidden layers. Let us assume that the input to Bi-GRU is given by $X = [x_1, x_2, x_3, \ldots, x_k]$ where k is the padded length of the input sentence. This information is passed to Bi-GRU layer which is explained in Section 3.2.1.2. However, in this case, we use a single structure Bi-GRU instead of two Bi-GRUs.

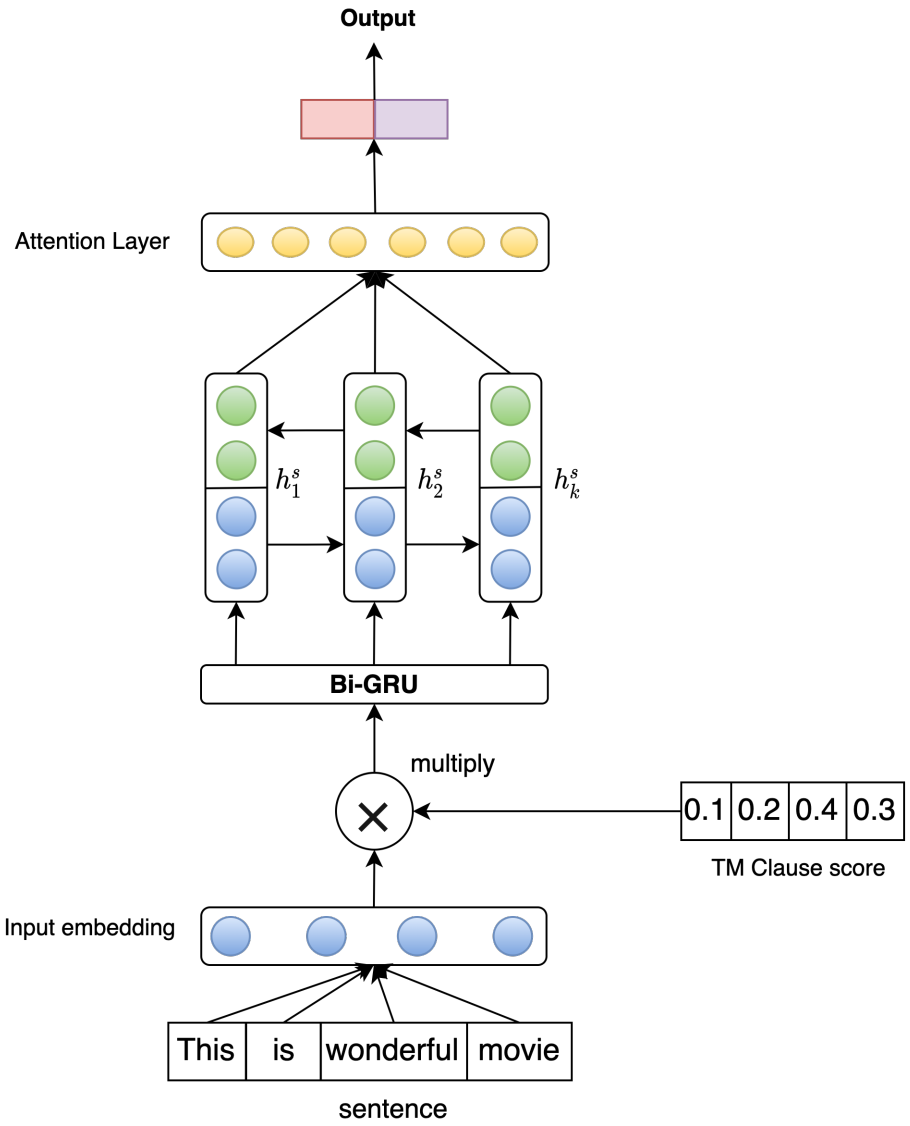


Figure 3.22: The two-action TA and its transition in TM.

3.2.2.3 Results

Here, we demonstrate the experiments and the results on the proposed model for enhancing the explanation of attention layer in text classification. We use two sentiment classification datasets for evaluation. They are:

- **MR** is a movie review dataset for binary sentiment classification with just one sentence per review [80]. There are 5331 positive reviews and 5331 critical reviews in the corpus. In this study, we used a training/test split from [81] (`https://github.com/mnqu/PTE/tree/master/data/mr`).
- **Reuters** The Reuters 21,578 dataset has two subsets: R52 and R83 (all-terms version). R8 is divided into eight categories, including 5485 training and 2189 exam papers. R52 is divided into 52 categories and 6532 training and 2568 test papers.

Here, we explore the proposed model's explainability by visualizing the respective attention

weights. The attention weight usually gives the impact of each individual feature on a particular prediction. However, such weight usually indicates the relationship between the input and the output, this method of interpreting the model can be beneficial for the system to understand the impact of each feature. As neural networks are already established BlackBox models, one can instead apply this interpretation to generate explainability for understanding the context of prediction. We define interpretation of the model as the weights obtained from the attention layer, and explainability as a use-case of interpretation to design the reasoning for a particular prediction that is easily understandable to humans. For ease of illustration, we visualize the attention weight of the Bi-GRU model and the attention weight of the Bi-GRU model initialized with TM's word score. We use the red color gradient to demonstrate the weight of each input word in the context. Dark color represents higher weight, with light color representing lower weight. As we can see from Fig. 3.23, only using Bi-GRU, the model recognizes mostly important words for predicting correct sentiment class. However, it is not perfect at the human level. However, Fig. 3.24 shows the visualization of attention weight using Bi-GRU and TM's score.

Here we can see that the model focuses on more significant words than the previous model. For instance, in the first example, the later model captures "look", "away" with higher weight, which is an important context for negative sentiment than "directing" and "attempt". This is more clearly seen in the third sample as the first model focus on "easily", "best", and "film" however our proposed model shifts the higher weightage to "best", "Korean", "film" for predicting the positive sentiment. One of the most peculiar cases where there are ambiguities in the context consisting of both positive and negative sentiment words is shown in the last example. Here using only Bi-GRU, the model captures "forgettable", "rip", and "work" as high-impact words. However, it does not give high weightage to the word "cheerful" which is also a sentiment carrying word. However, using our proposed model, the weightage changes drastically, and the model assigns higher weightage to "forgettable", "cheerful", "but", and "earlier". This makes more sense to human understanding because the context has the word "cheerful" and it is contradicted with the word "but" which eventually leads to a negative sentiment carrying word "forgettable" thereby making the whole context negative.

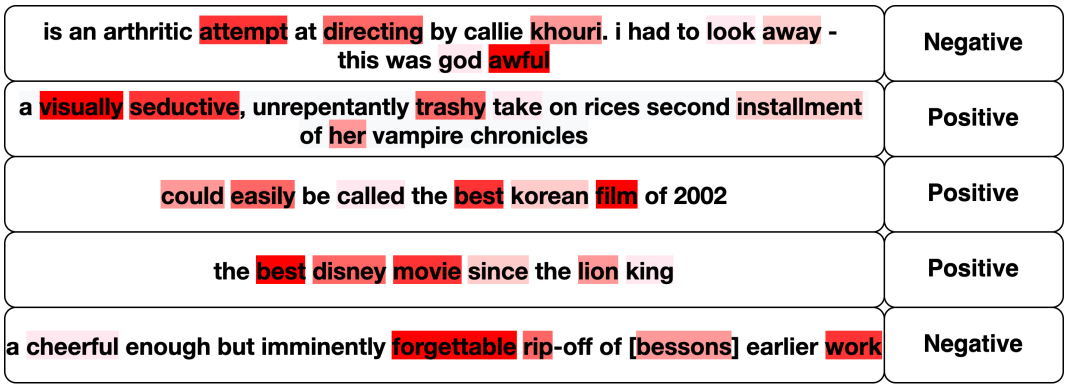


Figure 3.23: Visualization of attention weights with Bi-GRU only. Dark red to light red color represents the color gradients based on the attention weights in descending order.

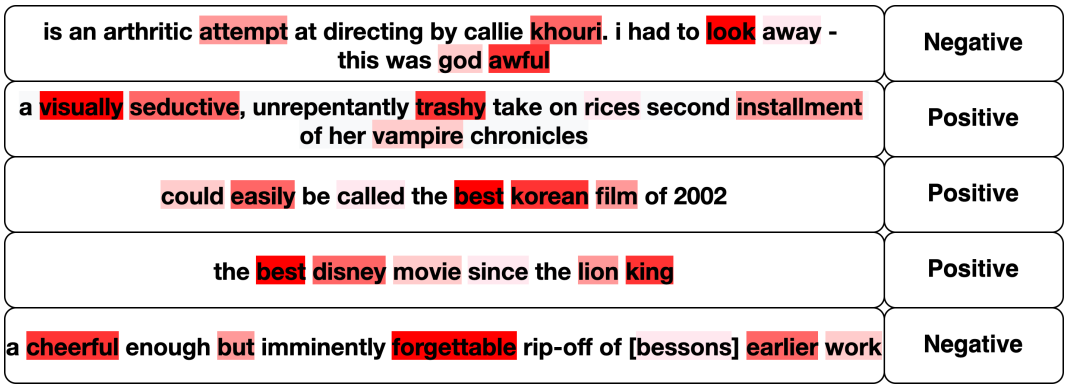


Figure 3.24: Visualization of attention weights with Bi-GRU and TM Score. Dark red to light red color represents the color gradients based on the attention weights in descending order.

3.3 Summary

In this chapter, we presented the various architectures and algorithms proposed for interpretable NLP. We presented the methods for interpretable NLP models for different NLP problems. The first section of this chapter consisted of the detailed framework of interpretable NLP models using TM. We demonstrated a technique based on the frequency of the literals in the clause to interpret an NLP model using TM on the WSD task. The outcome of this task showed that despite the huge vocabulary size, how the frequency of words appearing in the clause helps to understand the concept behind the prediction made by TM. We then extended this task further to non-traditional text classification (i.e., position-dependent text classification) for the ABSA task. There we designed a feature extraction technique where the position information was encoded into Boolean BOW so that TM can identify the context for the target word. Since TM suffered from lower accuracy compared with the state-of-the-art DNN models due to the fact that it cannot use the pre-trained word representation, we proposed feature augmentation to append a similar feature based on GloVe embedding in order to improve the performance. The result demonstrated that the model achieves significantly higher accuracy compared with vanilla TM. Now since most of the boxes were covered for a rule-based interpretable NLP model, there existed a serious problem of robustness of TM on spurious correlations. Hence we did an intensive analysis to understand the concept of the word involved in the classification. We discovered that the hyper-parameter called specificity (s) decides the probability of words and their negation to be included or excluded in the clause. Since TM is a transparent model, we visualized the learning way of TM based on the parameter s. We then proposed a novel way of using s in order to minimize the impact of spurious correlations in text classification.

In addition to TM-based approaches, we solved a major problem in text classification for easier interpretation of the model using the DNN-based attention model. Position-dependent text classification heavily relies on positional embedding in the initial layer of DNN which creates ambiguity in explaining the attention weights. This is because attention weights are directed not only to input rationales but also to positional embedding associated with them. In order to get rid of positional embedding, we designed a masking technique using two Bi-GRUs models each of which learns the representation of the original sentence and masked sentence. The result showed that the proposed model performs better than the positional embedding and the relationship

between attention weights to input rationales is no longer ambiguous. At last, we designed an ensemble architecture using DNN and TM to compensate for their limitation to have a trade-off between accuracy and interpretability. Since there are numerous studies showing that attention weights are not reliable, we propose a novel way of assigning prerequisite information to the DNN model using the clause score of each word from TM. This helps the model to understand the underneath concept of data distribution and helps the attention layer to converge towards the intended pattern in various distributions.

All the contributions give solutions to the research questions mentioned earlier in the thesis. Combining each and every technique and model gives a powerful interpretable architecture and algorithms for NLP that outperform existing solutions in terms of performance and/or interpretability, and in some cases, set up a completely new paradigm.

Chapter 4

Conclusions and Future Work

In this thesis, Interpretable Architectures and Algorithms for NLP are proposed, balancing the interpretability and performance of NLP applications. We divide the thesis into two parts. The first part deals with interpretable NLP models using TM and the second one focuses on the interpretation of the NLP model using DNN. We then extend this later model to design an ensemble approach to integrate information of TM into DNN as a prerequisite representation.

4.1 Conclusions to the Research Questions

In this section, we conclude the findings of our proposed methods in accordance to the research questions in Chapter 1.

Research Question 1: We design a novel technique to interpret populated clauses using the frequency of the literals. We used the WSD task as the dataset where the model has to classify the sense of a specific word given a particular context. Due to a large vocabulary in the NLP domain, clauses are over-populated with the literal and the conventional method of analyzing propositional logic seems to be a difficult way to interpret the model. Hence we design a frequency-based interpretation where we count the occurrence of each literal in the clause and filter the interpretability based on user requirements. The proposed model not only provides the interpretability for NLP but also achieves the state-of-the-art accuracy on WSD task.

We extend the traditional text classification to position-dependent text classification with a standard evaluation framework known as the ABSA task. We map the position as well as additional Sentiwordnet feature into Boolean BOW. The extracted features, when applied on TM, achieve accuracy on par with other position-dependent interpretable model on selected ABSA datasets of domain “Restaurant” and “Laptop”.

Research Question 2: TM is an interpretable model that operates on Boolean data, which restricts TM to use any pre-trained information such as word2vec and Glove embedding. For this reason, the performance of TM usually falls short of DNN models that are initialized by such word embedding. To boost the performance of TM, we adopt feature extraction using similar words from pre-trained word embedding to initialize TM’s BOW. Experiments show that the

proposed model achieves up to a 4% accuracy boost among the selected datasets.

Research Question 3: The state-of-the-art NLP models have raised the bar for excellent performance on a variety of tasks in recent years. However, concerns are rising over their primitive sensitivity to distribution biases that reside in the training and testing data. This issue hugely impacts the performance of the models when exposed to out-of-distribution and counterfactual data. Here, we employ TM that learns both simple and complex correlations by ANDing features and their negations. Specifically, we explore how non-negated reasoning can be more prone to distribution biases than negated reasoning. Experiments demonstrate that the negated clauses are robust to spurious correlations and outperform Naive Bayes, SVM, and Bi-LSTM by up to 20%, and ELMo by almost 6% on counterfactual test data.

Research Question 4: The interpretability of non-tradition text classification (ABSA) comes with the cost of performance in TM. This task heavily relies on positional embedding that creates ambiguity in interpreting the attention weights. Here we design a masking technique to remove positional embedding and maintain the performance. The proposed model also offers a direct relationship between attention weights and input rationales.

Research Question 5: The state-of-the-art NLP models are highly dominated by DNN-based models such as LSTM/GRUs and transformers. However, their BlackBox nature makes the interpretation ambiguous. On the other hand, TM offers an easy logic-based interpretation of the model but it comes with performance loss. The trade-off between accuracy and interpretability is one of the main concerns of this study. There has been an immense attempt of extracting a logical explanation from the attention layer of DNN. However, the change in attention weights based on various scenarios makes it arguably tough to establish a trustful model. Hence to mitigate the limitation of both models, we use the interpretable information from TM and integrate it into NN so that the model has prerequisite information about the distribution of the task thereby generating sensible attention weights. The experiments shows that the proposed ensemble method retains or outperforms the relatable state-of-art models thereby enhancing the attention weights towards more sensible input rationales.

4.2 Interpretable Text Classification Using TM

4.2.1 Bag-of-Words (BOW) based Text Classification

We have proposed a sense categorization approach based on TM. Although there are various methods for sense classification on a CoarseWSD-balanced dataset with good accuracy, many machine learning algorithms fail to provide a human interpretation that can explain the procedure of a particular classification. To overcome this issue, we present a TM-based sense classifier that learns the formulae from text corpus utilizing conjunctive clauses to demonstrate a particular feature of each category. Numerical results indicate that the TM-based approach is human-interpretable and it achieves a competitive accuracy, showing its potential for further WSD studies. In conclusion, we believe that the novel TM-based approach can have a significant impact on sense identification that is a very important factor in a chatbot or other WSD tasks.

4.2.2 Position Dependent Text Classification

Here, we aim to reduce the gap between the interpretability and the accuracy of position-dependent text classification known as ABSA by employing the TM. Our proposed model embeds the aspect-based inputs into binary form for classifying the sentiment of a particular word in a sentence. Such binary representations are then fed to a TM architecture where the learning process is transparent, which give a clear picture of what drives the TM to learn the particular sentiment for a given input. Additionally, we show the involvement of words carrying the sentiment for the aspect words in a case study. In short, the proposed model successfully provides a human-interpretable learning approach on ABSA task with comparable accuracy.

4.2.3 Enhancing Performance of TM

We aim to enhance the performance of TMs by introducing a novel way to exploit distributed feature representation. Given that a TM relies on Bag-of-words (BOW), it is not possible to introduce pre-trained word representation into a TM directly without sacrificing the interpretability of the model. To address this intertwined challenge, we extended each word feature by using cosine similarity on the distributed word representation. We proposed two techniques for feature extension: (1) using the k nearest words in embedding space and (2) including words within a given cosine angle (θ). Through this enhancement, the TM BOW can be boosted with pre-trained world knowledge in a simple yet effective way. Our experiment results showed that the enhanced TM not only achieves competitive accuracy compared with the state-of-the-art solutions but also outperforms some of the sophisticated DNN models. In addition, our BOW boosting also improves the interpretability of the model by increasing the scope of each clause, and semantically relating more samples. We thus believe that our proposed approach significantly enhances the TM in the accuracy/interpretability continuum, establishing a new standard in the field of explainable NLP.

4.2.4 Robust Text Classification against Spurious Correlations

Here, we employ TM to design a robust text classification against spurious correlations. TM learns the pattern using a set of clauses that are in the form of propositional logic. Such propositional logic is a combination of features in either non-negated or negated form. Since the propositional logic is human interpretable, it is easy to extract rule-based reasoning from TM. Our methods demonstrate that such a rule can be controlled or fine-tuned by modifying the hyper-parameter specificity s. We show that by keeping the value of s small, we can filter the clause from non-monotone to monotone where a majority of features are in the negated form thereby removing spurious correlations and forcing the model to rely on genuine correlations. Experiment results have shown that the proposed s-controlled TM outperforms various existing models on counterfactual test data. In addition, unlike DNNs, the human-level interpretation obtained from the rule-based reasoning of TM gives a complete understanding of how the model achieves its robustness.

4.3 Interpretable Text Classification Using Neural Network

4.3.1 Position Dependent Text Classification without Positional Embedding

In this part, we propose an efficient preprocessing scheme with an attention-based GRU model for aspect-based sentiment analysis. We first explore sentiment knowledge called Opinion Lexicon that is a list of positive, neutral, and negative sentiment words. In more detail, we replaced the words in ABSA dataset with a common tag, such as "positive" for positive sentiment words. This external input helps to bridge the gap from semantically related words to a certain extent and reduces the task's vocabulary. Since the ABSA is a position-dependent task, it requires the information of position along with sentence embedding or aspect embedding. The extra trainable weights for position information increase the complexity of the model. To reduce the complexity, we proposed a masking technique that masks the aspect word in the sentence with a common token "MASK". This masked embedding is separately sent to the model along with the sentence embedding. Experimentally, we have shown that the proposed scheme outperforms several position-aware methods with very straightforward attention-based BiGRUs architecture.

4.3.2 Enhancing Attention's Explanation Using TM

Recently, attention weights have been a great tool for visualizing the weight of input rationales in the model. However, their weight sometimes gives higher weightage to unwanted tokens that do not make sense to humans. This lead to the requirement of human-annotated rationales that are embedded into the models. Even if such human annotators are not a very extensive task to obtain while annotating new datasets, the problems come with annotating human rationales to existing datasets. It takes more time and is costly to re-annotate human rationales for interpretability. To solve this problem, we propose an alternative approach to obtain human interpretable rationales using TM. Since TM can be explained via logical rules, it provides human-level interpretation and is considered a prerequisite annotation of input rationales. The proposed model shows that embedding such information in attention-based models not only increases the accuracy but also enhances the weightage of the attention layer for each input rationale thereby making the explanation more sensible to humans. The visualization shows that the proposed model is capable of capturing the ambiguity of the context more efficiently than traditional models.

4.4 Future Works

The models proposed in this thesis pertain to various NLP tasks in TM separately. Even though these tasks are combined progressively, each of these tasks is independent in structure, which also applies to the models proposed using DNN in the second part of the thesis. However, to design a complete interpretable model for NLP, we would like to combine all these disjoints models into a large NLP framework that can apply to any type of dataset.

We would like to work on evaluating the explainability provided by TM and the ensemble model with human explainability. There are huge numbers of datasets that come with human rationales collected while labeling. Hence, there is a necessity of designing a TM model for NLP

that has significantly few propositional logic along with its evaluation with human rationales. However, reducing the propositional logic to a level where the human can easily comprehend the explanation is a challenging task because there is limited control over the size of propositional logic that TM creates. We would also like to design a weak supervision text classification using TM. The construction of a dataset is still a huge task that takes significant time and cost. Since TM is trained using a sample-wise method unlike DNN, it can be possible to design a text classification using weakly labeled data.

In addition to the above mentioned aspects, our ensemble approach in the second part of the thesis is only evaluated in one type of dataset. Hence, we would like to work on extending it to some sophisticated NLP tasks such as information retrieval, contextual text classification, and question answering. We believe that the interpretable information from TM can be helpful to attention-based DNN to enhance the focus on the respective part of the input representation.

Bibliography

[1] O.-C. Granmo, "The Tsetlin machine - a game theoretic bandit driven approach to optimal pattern recognition with propositional logic," *ArXiv*, vol. abs/1804.01508, 2018.

[2] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What does BERT look at? An analysis of BERT's attention," in *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, (Florence, Italy), pp. 276–286, Association for Computational Linguistics, 2019.

[3] S. Serrano and N. A. Smith, "Is attention interpretable?," in *ACL*, (Florence, Italy), pp. 2931–2951, ACL, 2019.

[4] S. Wiegreffe and Y. Pinter, "Attention is not not explanation," in *Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (Hong Kong, China), pp. 11–20, Association for Computational Linguistics, Nov. 2019.

[5] G. Boolos, "The logic of provability," 1993.

[6] J. W. Cain, "Mathematical models in the sciences," 2014.

[7] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv: Machine Learning*, 2017.

[8] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (XAI): Toward medical XAI," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 4793–4813, 2021.

[9] E. L. Zanutto, "A comparison of propensity score and linear regression analysis of complex survey data," *Journal of Data Science*, 2021.

[10] P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler, "Benign overfitting in linear regression," *Proceedings of the National Academy of Sciences*, vol. 117, pp. 30063 – 30070, 2020.

[11] D. LeJeune, H. Javadi, and R. Baraniuk, "The implicit regularization of ordinary least squares ensembles," in *AISTATS*, 2020.

[12] H. J. Kan, H. Kharrazi, H.-Y. Chang, D. P. Bodycombe, K. W. Lemke, and J. P. Weiner, "Exploring the use of machine learning for risk adjustment: A comparison of standard and penalized linear regression models in predicting health care costs in older adults," *PLoS ONE*, vol. 14, 2019.

[13] E. J. C. Priego, A. V. Olivares-Nadal, and P. R. Cobo, "Integer constraints for enhancing interpretability in linear regression," *Sort-statistics and Operations Research Transactions*, vol. 44, pp. 69–78, 2020.

[14] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 2004.

[15] J. J. Lin, C. Zhong, D. Hu, C. Rudin, and M. I. Seltzer, "Generalized and scalable optimal sparse decision trees," in *ICML*, 2020.

[16] M. Moshkovitz, Y.-Y. Yang, and K. Chaudhuri, "Connecting interpretability and robustness in decision trees through separation," in *ICML*, 2021.

[17] Y. D. Dhebar and K. Deb, "Interpretable rule discovery through bilevel optimization of split-rules of nonlinear decision trees for classification problems," *IEEE Transactions on Cybernetics*, vol. 51, pp. 5573–5584, 2021.

[18] A. McCallum and K. Nigam, "A comparison of event models for naive Bayes text classification," in *AAAI*, 1998.

[19] M. Loor and G. D. Tré, "Contextualizing naive Bayes predictions," *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, vol. 1239, pp. 814 – 827, 2020.

[20] B. Bhattarai., O. Granmo., and L. Jiao., "Measuring the novelty of natural language text using the conjunctive clauses of a Tsetlin machine text classifier," in *Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*, pp. 410–417, INSTICC, SciTePress, 2021.

[21] K. D. Abeyrathna, O.-C. Granmo, X. Zhang, L. Jiao, and M. Goodwin, "The regression Tsetlin machine: A novel approach to interpretable nonlinear regression," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 378, 2019.

[22] R. Saha, O.-C. Granmo, and M. Goodwin, "Mining interpretable rules for sentiment and semantic relation analysis using Tsetlin machines," in *Artificial Intelligence XXXVII*, (Cham), pp. 67–78, Springer International Publishing, 2020.

[23] I. J. Goodfellow, Y. Bengio, and A. C. Courville, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[24] W. Wang, L. Wang, R. Wang, Z. Wang, and A. Ye, "Towards a robust deep neural network in texts: A survey," *arXiv: Computation and Language*, 2019.

[25] A. Albarghouthi, "Introduction to neural network verification," *ArXiv*, vol. abs/2109.10317, 2021.

[26] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *ArXiv*, vol. abs/1710.05941, 2018.

[27] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, pp. 179–211, 1990.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[29] A. Graves, N. Jaitly, and A. rahman Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 273–278, 2013.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 30, (California, USA), Curran Associates, Inc., 2017.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[32] J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *ArXiv*, vol. abs/1607.06450, 2016.

[33] N. A. Smith, "Linguistic structure prediction," in *Synthesis Lectures on Human Language Technologies*, 2011.

[34] D. Yan, K. Li, S. Gu, and L. Yang, "Network-based bag-of-words model for text classification," *IEEE Access*, vol. 8, pp. 82641–82652, 2020.

[35] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems (NIPS), Nevada, USA*, vol. 26, pp. 3111–3119, Curran Associates, Inc., 2013.

[36] A. Bermingham and A. Smeaton, "On using Twitter to monitor political sentiment and predict election results," in *Proceedings of the Workshop on Sentiment Analysis where AI meets Psychology (SAAIP 2011)*, (Chiang Mai, Thailand), pp. 2–10, Asian Federation of Natural Language Processing, 2011.

[37] E. Altszyler, M. Sigman, and D. F. Slezak, "Comparative study of LSA vs word2vec embeddings in small corpora: A case study in dreams database," *ArXiv*, vol. abs/1610.01520, 2016.

[38] M. Naili, A. H. Chaïbi, and H. H. B. Ghézala, "Comparative study of word embedding methods in topic segmentation," in *KES*, 2017.

[39] Á. Elekes, A. Englhardt, M. Schäler, and K. Böhm, "Toward meaningful notions of similarity in NLP embedding models," *International Journal on Digital Libraries*, vol. 21, pp. 109–128, 2018.

[40] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, (Baltimore, Maryland), pp. 55–60, Association for Computational Linguistics, 2014.

[41] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[42] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (New Orleans, Louisiana), pp. 2227–2237, Association for Computational Linguistics, 2018.

[43] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, 2019.

[44] Y. Zhu, R. Kiros, R. S. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 19–27, 2015.

[45] T. Linzen, E. Dupoux, and Y. Goldberg, "Assessing the ability of LSTMs to learn syntax-sensitive dependencies," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 521–535, 2016.

[46] Y. Adi, E. Kermany, Y. Belinkov, O. Lavi, and Y. Goldberg, "Fine-grained analysis of sentence embeddings using auxiliary prediction tasks," *ArXiv*, vol. abs/1608.04207, 2017.

[47] E. Agirre and P. Edmonds, "Word sense disambiguation: Algorithms and applications," in *Springer, Dordrecht*, 2007.

[48] A. Raganato, J. Camacho-Collados, and R. Navigli, "Word sense disambiguation: A unified evaluation framework and empirical comparison," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, (Valencia, Spain), pp. 99–110, Association for Computational Linguistics, 2017.

[49] O. Lopez de Lacalle and E. Agirre, "A methodology for word sense disambiguation at 90% based on large-scale CrowdSourcing," in *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, (Denver, Colorado), pp. 61–70, Association for Computational Linguistics, 2015.

[50] K. Liao, D. Ye, and Y. Xi, "Research on enterprise text knowledge classification based on knowledge schema," in *2010 2nd IEEE International Conference on Information Management and Engineering*, pp. 452–456, April 2010.

[51] Y. Wang, L. Wang, M. Rastegar-Mojarad, S. Moon, F. Shen, N. Afzal, S. Liu, Y. Zeng, S. Mehrabi, S. Sohn, and H. Liu, "Clinical information extraction applications: A literature review," *Journal of Biomedical Informatics*, vol. 77, pp. 34 – 49, 2018.

[52] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, pp. 206–215, 2018.

[53] L. Zhang and B. Liu, *Sentiment Analysis and Opinion Mining*, pp. 1152–1161. Boston, MA: Springer US, 2017.

[54] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar, "SemEval-2014 task 4: Aspect based sentiment analysis," in *International Workshop on Semantic Evaluation (SemEval 2014)*, (Dublin, Ireland), pp. 27–35, ACL, 2014.

[55] D. Ma, S. Li, X. Zhang, and H. Wang, "Interactive attention networks for aspect-level sentiment classification," in *IJCAI*, (Melbourne, Australia), pp. 4068–4074, 2017.

[56] H. H. Do, P. Prasad, A. Maag, and A. Alsadoon, "Deep learning for aspect-based sentiment analysis: A comparative review," *Expert Systems with Applications*, vol. 118, pp. 272–299, 2019.

[57] W. Samek, G. Montavon, A. Vedaldi, L. Hansen, and K. Müller, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019.

[58] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, (California, USA), 2015.

[59] G. Brunner, Y. Liu, D. Pascual, O. Richter, M. Ciaramita, and R. Wattenhofer, "On identifiability in transformers," in *ICLR*, (Addis Ababa, Ethiopia), 2020.

[60] S. Vashishth, S. Upadhyay, G. S. Tomar, and M. Faruqui, "Attention interpretability across NLP tasks," *arXiv*, vol. 1909.11218, 2019.

[61] S. Gu, L. Zhang, Y. Hou, and Y. Song, "A position-aware bidirectional attention network for aspect-level sentiment analysis," in *COLING*, (Santa Fe, New Mexico, USA), pp. 774–784, ACL, 2018.

[62] J. Zhou, Q. Chen, X. Huang, Q. Hu, and L. He, "Position-aware hierarchical transfer model for aspect-level sentiment classification," *Information Sciences*, vol. 513, pp. 1–16, 2020.

[63] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *NAACL*, (New Orleans, Louisiana), pp. 2227–2237, ACL, 2018.

[64] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *ACM SIGKDD*, (New York, NY, USA), p. 168–177, ACM, 2004.

[65] D. Tang, B. Qin, and T. Liu, "Aspect level sentiment classification with deep memory network," in *EMNLP*, (Austin, Texas), pp. 214–224, ACL, 2016.

[66] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based LSTM for aspect-level sentiment classification," in *EMNLP*, (Austin, Texas), pp. 606–615, ACL, 2016.

[67] P. Chen, Z. Sun, L. Bing, and W. Yang, "Recurrent attention network on memory for aspect sentiment analysis," in *EMNLP*, (Copenhagen, Denmark), pp. 452–461, ACL, 2017.

[68] R. He, W. S. Lee, H. T. Ng, and D. Dahlmeier, "Exploiting document knowledge for aspect-level sentiment classification," in *ACL*, (Melbourne, Australia), pp. 579–585, ACL, 2018.

[69] Z. Lei, Y. Yang, M. Yang, W. Zhao, J. Guo, and Y. Liu, "A human-like semantic cognition network for aspect-level sentiment classification," in *AAAI*, (Hawaii, USA), 2019.

[70] X. Li, L. Bing, W. Lam, and B. Shi, "Transformation networks for target-oriented sentiment classification," in *ACL*, (Melbourne, Australia), pp. 946–956, ACL, 2018.

[71] Y. Liang, F. Meng, J. Zhang, J. Xu, Y. Chen, and J. Zhou, "A novel aspect-guided deep transition model for aspect based sentiment analysis," in *EMNLP-IJCNLP*, (Hong Kong, China), pp. 5569–5580, ACL, 2019.

[72] C. D. Manning and H. Schütze, "Foundations of statistical natural language processing," in *SGMD*, 2002.

[73] M. Haghighi, S. Johnson, X. Qian, K. Lynch, K. Vehik, and a. T. S. G. S. Huang, "A comparison of rule-based analysis with regression methods in understanding the risk factors for study withdrawal in a pediatric study," *Scientific Reports*, vol. 6, 2016.

[74] R. K. Yadav, L. Jiao, O.-C. Granmo, and M. Goodwin, "Interpretability in Word Sense Disambiguation using Tsetlin Machine," in *13th International Conference on Agents and Artificial Intelligence (ICAART), Vienna, Austria*, INSTICC, 2021.

[75] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, (Doha, Qatar), p. 1532–1543, ACL, 2014.

[76] Z. S. Harris, "Distributional structure," *WORD*, vol. 10, no. 2-3, pp. 146–162, 1954.

[77] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[78] J. Turian, L.-A. Ratinov, and Y. Bengio, "Word representations: A simple and general method for semi-supervised learning," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, (Uppsala, Sweden), pp. 384–394, Association for Computational Linguistics, 2010.

[79] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng, "Parsing with compositional vector grammars," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Sofia, Bulgaria), pp. 455–465, Association for Computational Linguistics, 2013.

[80] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *ACL*, (Michigan, USA), p. 115–124, ACL, 2005.

[81] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, (Sydney, NSW, Australia), p. 1165–1174, Association for Computing Machinery, 2015.

[82] X. Li and D. Roth, "Learning question classifiers," in *COLING*, 2002.

[83] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1746–1751, ACL, 2014.

[84] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," in *IJCAI*, p. 2873–2879, 2016.

[85] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning*, vol. 32 of *Proceedings of Machine Learning Research*, (Bejing, China), pp. 1188–1196, PMLR, 22–24 Jun 2014.

[86] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *EACL: Volume 2, Short Papers*, (Valencia, Spain), pp. 427–431, ACL, 2017.

[87] D. Shen, G. Wang, W. Wang, M. R. Min, Q. Su, Y. Zhang, C. Li, R. Henao, and L. Carin, "Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms," in *ACL (Volume 1: Long Papers)*, (Melbourne, Australia), pp. 440–450, ACL, 2018.

[88] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[89] H. Zhu and P. Koniusz, "Simple spectral graph convolution," in *International Conference on Learning Representations*, 2021.

[90] X. Liu, S. Wang, X. Zhang, X. You, J. Wu, and D. Dou, "Label-guided learning for text classification," *ArXiv*, vol. abs/2002.10772, 2020.

[91] Q. Qin, W. Hu, and B. Liu, "Feature projection for improved text classification," in *ACL*, (Online), pp. 8161–8171, ACL, 2020.

[92] Dragoș, C. Nicolae, and dragosnicolae, "Question classification using interpretable Tsetlin machine," in *International Workshop of Machine Reasoning*, ACM International Conference on Web Search and Data Mining, 2021.

[93] S. Garg and G. Ramakrishnan, "Bae: Bert-based adversarial examples for text classification," *arXiv*, vol. abs/2004.01970, 2020.

[94] A. Sauer and A. Geiger, "Counterfactual generative networks," in *ICLR*, (Online), 2021.

[95] Z. Wang and A. Culotta, "Identifying spurious correlations for robust text classification," in *Findings of the EMNLP 2020*, (Online), pp. 3431–3440, ACL, 2020.

[96] E. Wulczyn, N. Thain, and L. Dixon, "Ex machina: Personal attacks seen at scale," in *International Conference on World Wide Web*, (Perth, Australia), p. 1391–1399, WWW, 2017.

[97] S. Kaufman, S. Rosset, C. Perlich, and O. Stitelman, "Leakage in data mining: Formulation, detection, and avoidance," *ACM Trans. Knowl. Discov. Data*, vol. 6, 2012.

[98] D. Kaushik, E. Hovy, and Z. Lipton, "Learning the difference that makes a difference with counterfactually-augmented data," in *ICLR*, (Online), 2020.

[99] R. K. Yadav, L. Jiao, O.-C. Granmo, and M. Goodwin, "Enhancing interpretable clauses semantically using pretrained word representation," in *Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, (Punta Cana, Dominican Republic), pp. 265–274, Association for Computational Linguistics, 2021.

[100] R. K. Yadav, L. Jiao, O.-C. Granmo, and M. Goodwin, "Human-Level Interpretable Learning for Aspect-Based Sentiment Analysis," in *AAAI, Vancouver, Canada*, AAAI, 2021.

[101] O.-C. Granmo, S. Glimsdal, L. Jiao, M. Goodwin, C. W. Omlin, and G. T. Berge, "The convolutional Tsetlin machine," *arXiv*, vol. 1905.09688, 2019.

[102] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *EMNLP-IJCNLP*, (Hong Kong, China), pp. 188–197, ACL, 2019.

[103] S. Rosenthal, N. Farra, and P. Nakov, "SemEval-2017 task 4: Sentiment analysis in Twitter," in *Proceedings of the 11th SemEval-2017*, (Vancouver, Canada), pp. 502–518, ACL, 2017.

[104] B. Kim, S. Ryu, and G. Lee, "Two-stage multi-intent detection for spoken language understanding," *Multimedia Tools and Applications*, vol. 76, pp. 11377–11390, 2016.

[105] B. Xu, X. Wang, B. Yang, and Z. Kang, "Target embedding and position attention with LSTM for aspect based sentiment analysis," in *International Conference on Mathematics and Artificial Intelligence*, ICMAI, (New York, NY, USA), p. 93–97, ACM, 2020.

[106] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao, "Relation classification via convolutional deep neural network," in *COLING, Dublin, Ireland*, p. 2335–2344, 2014.

[107] Y. Song, J. Wang, T. Jiang, Z. Liu, and Y. Rao, "Attentional encoder network for targeted sentiment classification," *ArXiv*, vol. abs/1902.09314, 2019.

[108] K. Xu, H. Zhao, and T. Liu, "Aspect-specific heterogeneous graph convolutional network for aspect-based sentiment classification," *IEEE Access*, vol. 8, pp. 139346–139355, 2020.

[109] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar, "SemEval-2014 task 4: Aspect based sentiment analysis," in *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval Dublin, Ireland*, pp. 27–35, Aug. 2014.

[110] M. Pontiki, D. Galanis, H. Papageorgiou, S. Manandhar, and I. Androutsopoulos, "SemEval-2015 task 12: Aspect based sentiment analysis," in *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval, Denver, Colorado, USA*, pp. 486–495, 2015.

[111] M. Pontiki, D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, M. AL-Smadi, M. Al-Ayyoub, Y. Zhao, B. Qin, O. De Clercq, V. Hoste, M. Apidianaki, X. Tannier, N. Loukachevitch, E. Kotelnikov, N. Bel, S. M. Jiménez-Zafra, and G. Eryiğit, "SemEval-2016 task 5: Aspect based sentiment analysis," in *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval, San Diego, California, USA*, pp. 19–30, 2016.

[112] W. Chen, Y. Su, Y. Shen, Z. Chen, X. Yan, and W. Y. Wang, "How large a vocabulary does text classification need? A variational approach to vocabulary selection," in *NAACL*, (Minneapolis, MN, USA), pp. 3487–3497, ACL, June 2019.

[113] C. W. Wu, "Prodsumnet: reducing model parameters in deep neural networks via product-of-sums matrix decompositions," *arXiv*, vol. abs/1809.02209, 2019.

[114] T. Mikolov, M. Karafi, and S. Khudanpur, "Recurrent neural network based language model," in *INTERSPEECH, Makuhari, Chiba, Japan*, 2010.

[115] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Workshop on Deep Learning@NIPS*, (Montréal, Canada), 2014.

[116] S. Kiritchenko, X. Zhu, C. Cherry, and S. Mohammad, "NRC-Canada-2014: Detecting aspects and sentiment in customer reviews," in *International Workshop on Semantic Evaluation (SemEval 2014)*, (Dublin, Ireland), pp. 437–442, ACL, 2014.

[117] O. Wallaart and F. Frasincar, "A hybrid approach for aspect-based sentiment analysis using a lexicalized domain ontology and attentional neural models," in *ESWC, Portoroz, Slovenia*, 2019.

[118] S. Wu, Y. Xu, F. Wu, Z. Yuan, Y. Huang, and X. Li, "Aspect-based sentiment analysis via fusing multiple sources of textual knowledge," *Knowledge-Based Systems*, vol. 183, p. 104868, 2019.

[119] A. Rietzler, S. Stabinger, P. Opitz, and S. Engl, "Adapt or get left behind: Domain adaptation through BERT language model finetuning for aspect-target sentiment classification," *ArXiv*, vol. abs/1908.11860, 2020.

[120] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," in *Conference on Empirical Methods in Natural Language Processing*, (Austin, Texas), pp. 2249–2255, Association for Computational Linguistics, Nov. 2016.

[121] W. Wang, N. Yang, F. Wei, B. Chang, and M. Zhou, "Gated self-matching networks for reading comprehension and question answering," in *55th Annual Meeting of the Association*

for Computational Linguistics (Volume 1: Long Papers), (Vancouver, Canada), pp. 189–198, Association for Computational Linguistics, July 2017.

[122] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019.

[123] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *CoRR*, vol. abs/1312.6034, 2014.

[124] S. Jain and B. C. Wallace, "Attention is not Explanation," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 3543–3556, Association for Computational Linguistics, June 2019.

[125] O. Zaidan, J. Eisner, and C. Piatko, "Using annotator rationales to improve machine learning for text categorization," in *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, (Rochester, New York), pp. 260–267, Association for Computational Linguistics, Apr. 2007.

[126] Y. Zhang, I. Marshall, and B. C. Wallace, "Rationale-augmented convolutional neural networks for text classification," in *Conference on Empirical Methods in Natural Language Processing*, (Austin, Texas), pp. 795–804, Association for Computational Linguistics, Nov. 2016.

[127] J. Chung, Çaglar Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *ArXiv*, vol. abs/1412.3555, 2014.

Part II

Appended Papers

Appendix A

Paper A

Title:	Interpretability in Word Sense Disambiguation Using Tsetlin Machine
Authors:	Rohan Kumar Yadav, Lei Jiao, Ole-Christoffer Granmo, and Morten Goodwin
Affiliation:	University of Agder, Faculty of Engineering and Science, 4879, Grimstad, Norway
Conference:	*13th International Conference on Agents and Artificial Intelligence (ICAART)*, Vienna, Austria, Feb. 2021.
DOI:	10.5220/0010382104020409.

A

Interpretability in Word Sense Disambiguation Using Tsetlin Machine

Rohan Kumar Yadav, Lei Jiao, Ole-Christoffer Granmo, and Morten Goodwin
Department of Information and Communication Technology
Faculty of Engineering and Science, University of Agder
4879, Grimstad, Norway
E-mails: {rohan.k.yadav, lei.jiao, ole.granmo, morten.goodwi}@uia.no

Abstract — **Word Sense Disambiguation (WSD) is a longstanding unresolved task in Natural Language Processing. The challenge lies in the fact that words with the same spelling can have completely different senses, sometimes depending on subtle characteristics of the context. A weakness of the state-of-the-art supervised models, however, is that it can be difficult to interpret them, making it harder to check if they capture senses accurately or not. In this paper, we introduce a novel Tsetlin Machine (TM) based supervised model that distinguishes word senses by means of conjunctive clauses. The clauses are formulated based on contextual cues, represented in propositional logic. Our experiments on CoarseWSD-balanced dataset indicate that the learned word senses can be relatively effortlessly interpreted by analyzing the converged model of the TM. Additionally, the classification accuracy is higher than that of FastText-Base and similar to that of FastText-CommonCrawl.**

A.1 Introduction

Word Sense Disambiguation (WSD) is one of the unsolved task in Natural Language Processing (NLP) [1] with rapidly increasing importance, particularly due to the advent of chatbots. WSD consists of distinguishing the meaning of homographs – identically spelled words whose sense or meaning depends on the surrounding context words in a sentence or a paragraph. WSD is one of the main NLP tasks that still revolves around the perfect solution of sense classification and indication [2], and it usually fails to be integrated into NLP applications [3]. Many supervised approaches attempt to solve the WSD problem by training a model on sense annotated data [4]. However, most of them fail to produce interpretable models. Because word senses can be radically different depending on the context, interpretation errors can have adverse consequences in real applications, such as chatbots. It is therefore crucial for a WSD model to be easily interpretable for human beings, by showing the significance of context words for WSD.

NLP is one of the discipline that are used as the application in a chatbot. With the recent proliferation of chatbots, the limitations of the state-of-the-art WSD has become increasingly apparent. In real-life operation, chatbots are notoriously poor in distinguishing the meaning of words with multiple senses, distinct for different contexts. For example, let us consider the word "book" in the sentence "I want to book a ticket for the upcoming movie". Although a traditional chatbot can classify "book" as "reservation" rather than "reading material", it does not give us an explanation of how it learns the meaning of the target word "book". An unexplained model raises several questions, like: "How can we trust the model?" or "How did the model make the decision?". Answering these questions would undoubtedly make a chatbot more trustworthy. In particular, deciding word senses for the wrong reasons may lead to undesirable consequences, e.g., leading the chatbot astray or falsely categorizing a CV. Introducing a high level of interpretability while maintaining classification accuracy is a challenge that the state-of-the-art NLP techniques so far have failed to solve satisfactorily.

Although some of the rule-based methods, like decision trees, are somewhat easy to interpret, other methods are out of reach for comprehensive interpretation [5], such as Deep Neural Networks (DNNs). Despite the excellent accuracy achieved by DNNs, the "black box" nature impedes their impact [6]. It is difficult for human beings to interpret the decision-making process of artificial neurons. Weights and bias of deep neural networks are in the form of fine-tuned continuous values that make it intricate to distinguish the context words that drive the decision for classification. Some straightforward techniques such as Naive Bayes classifier, logistic regression, decision trees, random forest, and support vector machine are therefore still widely used because of their simplicity and interpretability. However, they provide reasonable accuracy only when the data is limited.

In this paper, we aim to obtain human-interpretable classification of the CoarseWSD-balanced dataset, using the recently introduced Tsetlin Machine (TM). Our goal is to achieve a viable balance between accuracy and interpretability by introducing a novel model for linguistic patterns. TM is a human interpretable pattern recognition method that composes patterns in propositional logic. Recently, it has provided comparable accuracy as compared to DNN with arguably less computational complexity, while maintaining high interpretability. We demonstrate how our model learns pertinent patterns based on the context words, and explore which context words drive the classification decisions of each particular word sense. The rest of the paper is arranged

as follows: The related work on WSD and TM are explained in Section A.2. Our TM-based WSD-architecture, the learning process, and our approach to interpretability are covered in Section A.3. Section A.4 presents the experiment results for interpretability and accuracy. We conclude the paper in Section A.5.

A.2 Related Work

The research area of WSD is attracting increasing attention in the NLP community [1] and has lately experienced rapid progress [7, 8, 9]. In all brevity, WSD methods can be categorized into two groups: knowledge-based and supervised WSD. Knowledge-based methods involve selecting the sense of an ambiguous word from the semantic structure of lexical knowledge bases [10]. For instance, the semantic structure of BabelNet has been used to measure word similarity [11]. The benefit of using such models is that they do not require annotated or unannotated data but rely heavily on the synset relations. Regarding supervised WSD, traditional approaches generally depend on extracting features from the context words that are present around the target word [12].

The success of deep learning has significantly fueled WSD research. For example, Le et al. have reproduced the state-of-the-art performance of an LSTM-based approach to WSD on several openly available datasets : GigaWord, SemCor [13], and OMSTI [14]. Apart from traditional supervised WSD, embedding is becoming increasingly popular to capture the senses of words [15]. Further, Majid et al. improve the state-of-the-art supervised WSD by assigning vector coefficients to obtain more precise context representations, and then applying PCA dimensionality reduction to find a better transformation of the features [16]. Salomonsson presents a supervised classifier based on bidirectional LSTM for the lexical sample task of the Senseval dataset [17].

Contextually-aware word embedding has been extensively addressed with other machine learning approaches across many disciplines. Perhaps the most relevant one is the work on neural network embedding [18, 19, 20]. There is a fundamental difference between our work and previous ones in terms of interpretability. Existing methods yield complex vectorized embedding, which can hardly be claimed to be human interpretable. Furthermore, natural language processing has, in recent years, been dominated by neural network-based attention mechanisms [21, 22]. Even though attentions and the attention-based transformers [23] implementation provide the state-of-the-art results, the methods are overly complicated and far from interpretable. The recently introduced work [24] shows how contextual information influences the sense of a word via the analysis of WSD on BERT.

All these contributions clearly show that supervised neural models can achieve the state-of-the-art performance in terms of accuracy without considering external language-specific features. However, such neural network models are criticized for being difficult to interpret due to their black-box nature [25]. To introduce interpretability, we employ the newly developed TM for WSD in this study. The TM paradigm is inherently interpretable by producing rules in propositional logic [26]. TMs have demonstrated promising results in various classification tasks involving image data [27], NLP tasks [28, 29, 30, 31] and board games [26]. Although the TM operates on binary data, recent work suggests that a threshold-based representation of continuous input allows the TM to perform successfully beyond binary data, e.g., applied to

diseases outbreak forecasting [32]. Additionally, the convergence of TM has been analysed in [33].

A.3 System Architecture for Word Sense Disambiguation

A.3.1 Basic Concept of Tsetlin Machine for Classifying Word Senses

At the core of the TM one finds a novel game-theoretic scheme that organizes a decentralized team of Tsetlin Automata (TAs). The scheme guides the TAs to learn arbitrarily complex propositional formula, based on disjunctive normal form (DNF). Despite its capacity to learn complex nonlinear patterns, a TM is still interpretable in the sense that it decomposes problems into self-contained sub-patterns that can be interpreted in isolation. Each sub-pattern is represented as a conjunctive clause, which is a conjunction of literals with each literal representing either an input bit or its negation. Accordingly, both the representation and evaluation of sub-patterns are Boolean. This makes the TM computationally efficient and hardware friendly compared with other methods. In the following paragraphs, we present how the TM architecture can be used for WSD.

The first step in our architecture for WSD, shown in Fig. A.2, is to remove the stop-words from the text corpus, and then stem the remaining words[1]. Thereafter, each word is assigned a propositional variable $x_k \in \{0, 1\}$, $k \in \{1, 2, \ldots, n\}$, determining the presence or absence of that word in the context, with n being the size of the vocabulary. Let $\boldsymbol{X} = [x_1, x_2, ..., x_n]$ be the feature vector (input) for the TM, which is thus a simple bag of words constructed from the text corpus, as shown in Fig. A.2.

The above feature vector is then fed to a TM classifier, whose overall architecture is shown in Fig. A.1. Multiclass Tsetlin Machine consists of multiple TM and each TM has several TA teams which is expanded in Fig. A.1(b). We first cover how the TM performs classification before we show how the classification rules are formed to perform WSD. As shown in Fig. A.1(b), $\boldsymbol{X}$ is the input to the TM. For our purpose, each sense is seen as a class, and the context of the word to be disambiguated is the feature vector (the bag of words). If there are q classes and m sub-patterns per class, the classification problem can be solved using $q \times m$ conjunctive clauses, $C_i^j, 1 \leq j \leq q, 1 \leq i \leq m$:

$$C_i^j = \left(\bigwedge_{k \in I_i^j} x_k \right) \wedge \left(\bigwedge_{k \in \bar{I}_i^j} \neg x_k \right), \tag{A.1}$$

where I_i^j and $\bar{I}_i^j$ are non-overlapping subsets of the input variable indexes. A particular subset is responsible for deciding which of the propositional variables take part in the clause and also if they are negated or not. In more details, the indices of input variables in I_j^i represent the literals that are included as is, while the indices of input variables in $\bar{I}_j^i$ correspond to the negated ones. The propositional variables or their negations are related with the conjunction operator to form a clause $C_i^j(X)$ which is shown as example in Eq. (A.2)

$$C_i^j(X) = x_1 \wedge \neg x_3 \wedge \ldots \wedge x_{k-1} \wedge \neg x_k. \tag{A.2}$$

[1] In this work, we used the PortStemmer package.

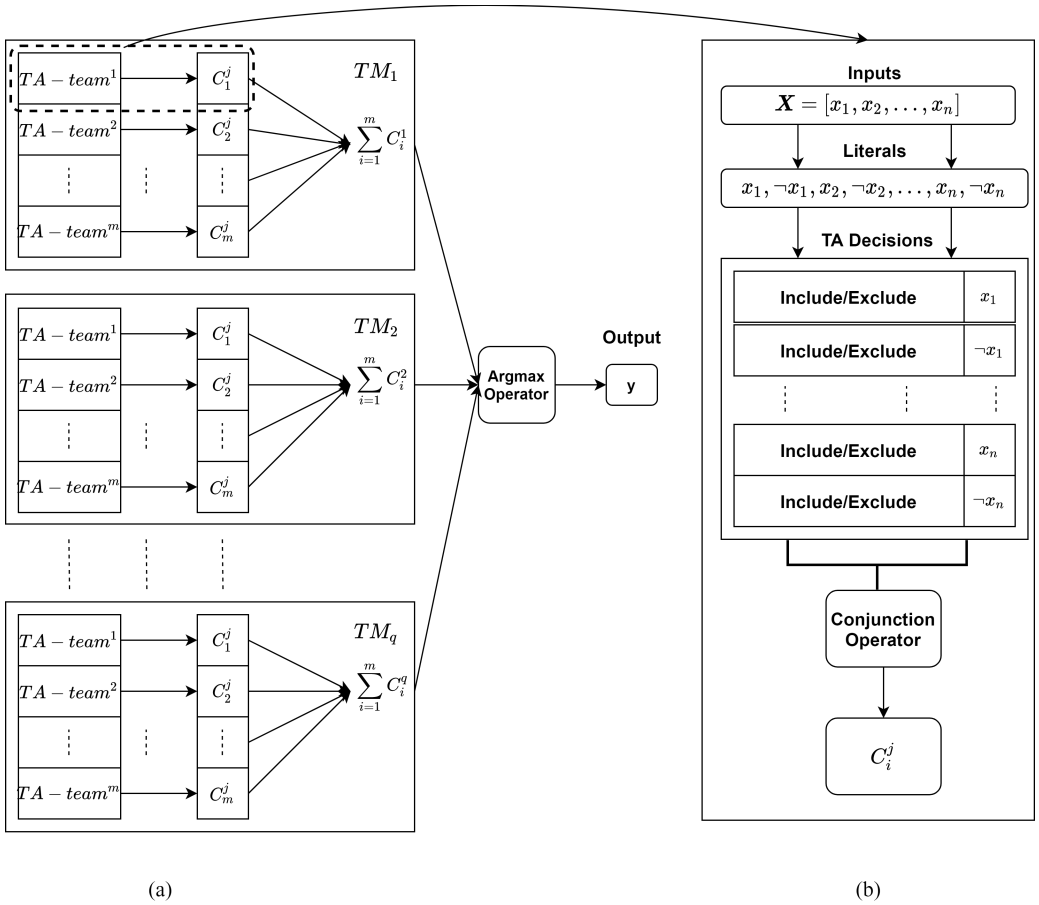


Figure A.1: The architecture of (a) multiclass Tsetlin Machine, (b) a TA-team forms the clause $C_i^j, 1 \leq j \leq q, 1 \leq i \leq m$.

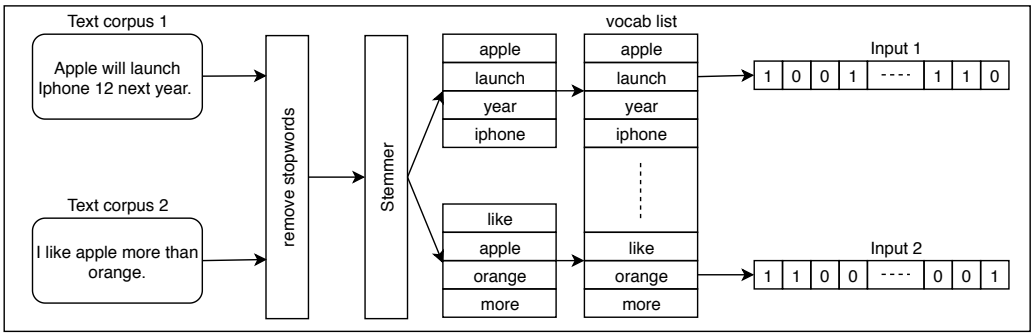


Figure A.2: Preprocessing of text corpus for input to TM.

To distinguish the class pattern from other patterns (1-vs-all), clauses with odd indexes are assigned positive polarity $(+)$ and the even indexed ones are assigned negative $(-)$. Clauses with positive polarity vote for the target class, while clauses with negative index vote against it. Finally, a summation operator aggregates the votes by subtracting the number of negative votes from the positive votes, per Eq. (A.3).

$$f^j(X) = \Sigma_{i=1}^{m}(-1)^{m-1} C_i^j(X). \tag{A.3}$$

In a multi-class TM, the final decision is made by an argmax operator to classify the input

based on the highest sum of votes, as shown in Eq. (A.4):

$$y = \text{argmax}_j \left(f^j(X) \right). \tag{A.4}$$

A.3.2 Training of the Proposed Scheme

The training of the TM is explained in detail in [26]. Our focus here is how the word senses are captured from data. Let us consider one training example $(\boldsymbol{X}, \hat{y})$. The input vector $\boldsymbol{X}$ – a bag of words – represents the input to the TM. The target $\hat{y}$ is the sense of the target word.

Multiple teams of TAs are responsible for TM learning. As shown in Fig. A.1(b), a clause is assigned one TA per literal. A TA is a deterministic automaton that learns the optimal action among the set of actions provided by the environment. The environment, in this particular application, is the training samples together with the updating rule of the TA, which is detailed in [26]. Each TA in the TM has $2N$ states and decides among two actions: Action 1 and Action 2, as shown in Fig. A.3. The present state of the TA decides its action. Action 1 is performed from state 1 to N whereas Action 2 is performed for states $N+1$ to $2N$. The selected action is rewarded or penalized by the environment. When a TA receives a reward, it emphasizes the action performed by moving away from the center (towards left or right end). However, if penalty happens, the TA moves towards the center to weaken the performed action, eventually switching to the other action.

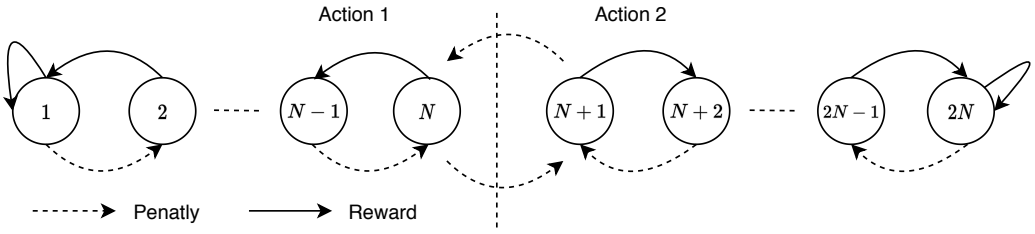


Figure A.3: Representation of two actions of TA.

In TM, each TA chooses either to exclude (Action 1) or include (Action 2) its assigned literal. Based on the decisions of the TA team, the structure of the clause is determined and the clause can therefore generate an output for the given input $\boldsymbol{X}$. Thereafter, the state of each TA is updated based on its current state, the output of the clause C_i^j for the training input $\boldsymbol{X}$, and the target $\hat{y}$.

We illustrate here the training process by way of example, showing how a clause is built by excluding and including words. We consider the bag of words for "Text Corpus 2": (apple, like, orange, and more) in Fig. A.2, converted into binary form "Input 2". As per Fig. A.4, there are eight TAs with $N = 100$ states per action that co-produce a single clause. The four TAs (TA to the left in Fig. A.4) vote for the intended sense with "more", "like", "orange", and "apple", whereas the four TAs (TA' to the right in Fig. A.4) vote against it. The terms that are moving away from the central states are receiving rewards, while those moving towards the centre states are receiving penalties. In Fig. A.4, from the TAs to the left, we obtain a clause[2]

[2]As the clause describes a sub-pattern within the same class, we ignore the superscript for different classes in notation C_i^j.

C_1 = “apple” ∧ “like”. The status of “orange” is excluded for now. However, after observing more evidences from the “Input 2”, the TA of “orange” is penalized for its current action, making it change its action from exclude to include eventually. In this way, after more updates, the word “orange” is to be included in the clause, thereby making C_1 = “apple” ∧ “like” ∧ “orange”, increasing the precision of the sub-pattern and thereby the classification.

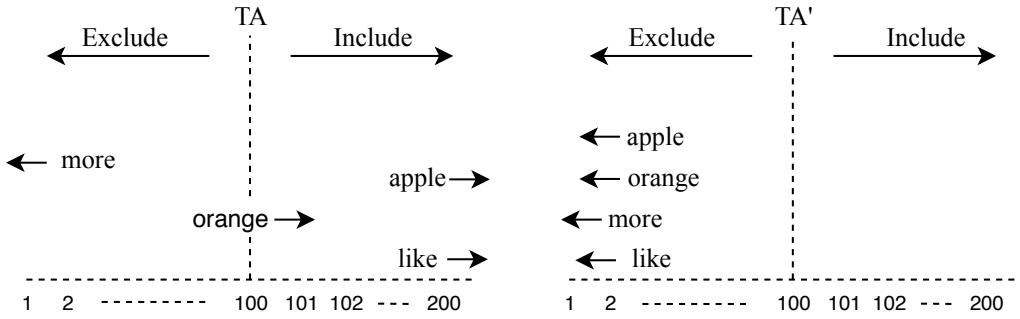


Figure A.4: Eight TA with 100 states per action that learn whether to exclude or include a specific word (or its negation) in a clause.

A.3.3 Interpretable Classification Process

We now detail the interpretability once the TM has been trained. In brief, the interpretability is based on the analysis of clauses. Let us consider the noun “apple” as the target word. For simplicity, we consider two senses of “apple”, i.e., Company as sense s_1 and Fruit as sense s_2. The text corpus for s_1 is related to the apple being a company, whereas for s_2 it is related to the apple being a fruit.

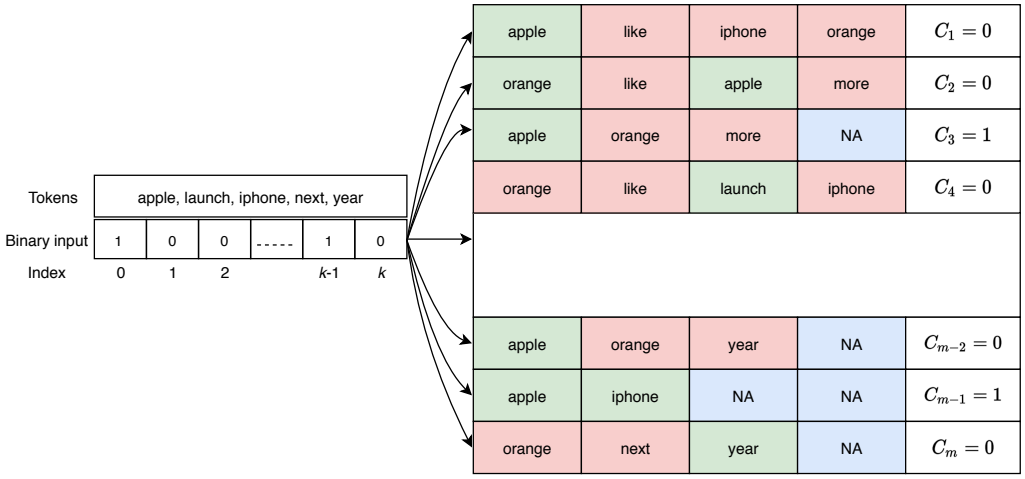


Figure A.5: Structure of clauses formed by the combination of sub-patterns. Green color indicates the literals that are included as original, red color indicates the literals that are included as the negated form and the blue color boxes indicates that there are no literals because not all the clauses has same number of literals.

Let us consider a test sample $I_{test} = [\text{apple}, \text{launch}, \text{iphone}, \text{next}, \text{year}]$ and how its sense is classified based on the context words. This set of words is first converted to binary form based

on a bag of words as described earlier in Fig. A.2.

To extract the clauses that vote for sense s_1, the test sample I_{test} is passed to the model and the clauses that vote for the presence of sense s_1 are observed as shown in Fig. A.5. The literals formed by TM are expressed in indices of the tokens. For ease of understanding, it has been replaced by the corresponding word tokens. The green box shows that the literal is non-negated whereas the red box denotes the negated form of the literal as shown in Fig. A.5. For example, the sub-patterns created by clause $C_3 = \text{apple} \wedge \neg\text{orange} \wedge \neg\text{more}$. These clauses consist of included literals in conjunctive normal form (CNF). Since the clauses in the TM are trained sample-wise, there exist several randomly placed literals in each clause. These random literals just occur because of randomly picked words that do not effect the classification. These literals are assigned to be non-important literals and their frequency of occurrence is low. On the other hand, the literals that has higher frequency among the clauses are considered to be important literals and hence makes significant impact on classification. Here, we emphasize on separating important and non-important literals for easy human interpretation. The general concept for finding the important words for a certain sense is to observe the frequency of appearances for a certain word in the trained clauses. To do that, in the above example, once the TM is trained, the literals in clauses that output 1 or votes for the presence of the class s_1 for I_{test} are collected first, as shown in Eq. (A.5):

$$L_t = \bigcup_{\substack{k,j,\\ \forall C_j=1}} \{x_k^j, \neg x_k^j\}, \tag{A.5}$$

where x_k^j is the k^{th} literal, i.e., x_k, that appears in clause j and $\neg x_k^j$ is the negation of the literal. Note that a certain literal x_k may appear many times in L_t due to the multiple clauses that output 1. Clearly, L_t is a set of literals (words) that appears in all clauses that contribute to the classification of class s_1. The next step is to find frequently appearing literals (words) in L_t, which correspond to the important words. We define a function, $\beta(h, H)$, which returns the number of the elements h in the set H. We can then formulate a set of the numbers for all literals x_k and their negations $\neg x_k$ in L_t, $k \in \{1, 2, \ldots, n\}$, as shown in Eq. (A.6):

$$S_t = \Big\{ \bigcup_{k=1:n} \beta(x_k, L_t), \bigcup_{k=1:n} \beta(\neg x_k, L_t) \Big\}. \tag{A.6}$$

We rank the number of elements in set S_t in descending order and consider the first η percent in the rank as the important literals. Similarly, we define the last η percent in the rank as non-important literals. To distinguish the important literals more precisely, several independent experiments can be carried out for a certain sense. Following the same concept, the literals in M different experiments can be collected to one set $L_t(total)$ as shown in Eq. (A.7):

$$L_t(total) = \bigcup_{e=1}^{M} (L_t)_e, \tag{A.7}$$

where $(L_t)_e$ is the set of literals for the e^{th} experiment. Similarly, the counts of all literals in these experiments, stored in set $S_t(total)$ shown in Eq. (A.8), are again ranked and the top η percent is deemed as important literals and the last η percent is the non-important literals. The

parameter η is to be tuned according to the level of human interpretation required for a certain task.

$$S_t(total) = \left\{ \bigcup_{k=1:n} \beta(x_k, L_t(total)), \bigcup_{k=1:n} \beta(\neg x_k, L_t(total)) \right\}. \tag{A.8}$$

A.4 Evaluations

We present here the classification and interpretation results on CoarseWSD-balanced dataset. There are 20 words having more than two senses to be classified. We select four of them to evaluate our model. The reason for selecting only four words than using all 20 words is that we want to show that TM preserves interpretability with maintaining state-of-the-art accuracy. So using only four words are enough to represent the trade of between interpretability and accuracy. The details of four datasets are shown in Table A.1. To train the TM for this task, we use the same configuration of hyperparameters for all the target words. More specifically, we use the number of clauses, specificity s and target T as 500, 5 and 80 for Apple and JAVA whereas 250, 3 and 30 for Spring and Crane. After the model is trained for each target, we validate our results using test data.

Table A.1: Senses associated with each word that is to be classified.

Dataset	**Sense1**	**Sense2**	**Sense3**
Apple	fruit	company	NA
JAVA	computer	location	NA
Spring	hydrology	season	device
Crane	machine	bird	NA

To illustrate the interpretability, let us take a sample as an example to extract the literals that are responsible for the classification of an input sentence: "**former apple ceo, steve jobs, holding a white iphone 4**". Once this input is passed through the model, TM predicts its sense as a company and we examine the clauses that output 1. We append all the literals that are presented in each clause and calculate the number of appearances for each literal. The number of appearances of a certain literal for the selected sample after one experiment is shown in Figs. A.6 and A.7 by a blue line. After five experiments, the number for a certain literal is shown by a red line in Figs. A.6 and A.7. Clearly, it makes sense that the negated form of the mostly-appearing literals in '. A.6, i.e., "not tree", "not fruit", " not cherries" etc. indicate that the word "apple" does not mean a fruit but a company. Nevertheless, as stated in the previous section, there are also some literals which are randomly placed in the clause and are non repetitive because the counts refuse to climb up for the same input, marking them not important literals, shown in Fig. A.7.

In addition to the interpretability of TM based approach, the accuracy is also an important parameter for performance evaluation. Even though the selected datasets have binary sense classification, we will use Micro-F1 and Macro-F1 as the evaluation metrics as shown in [24].

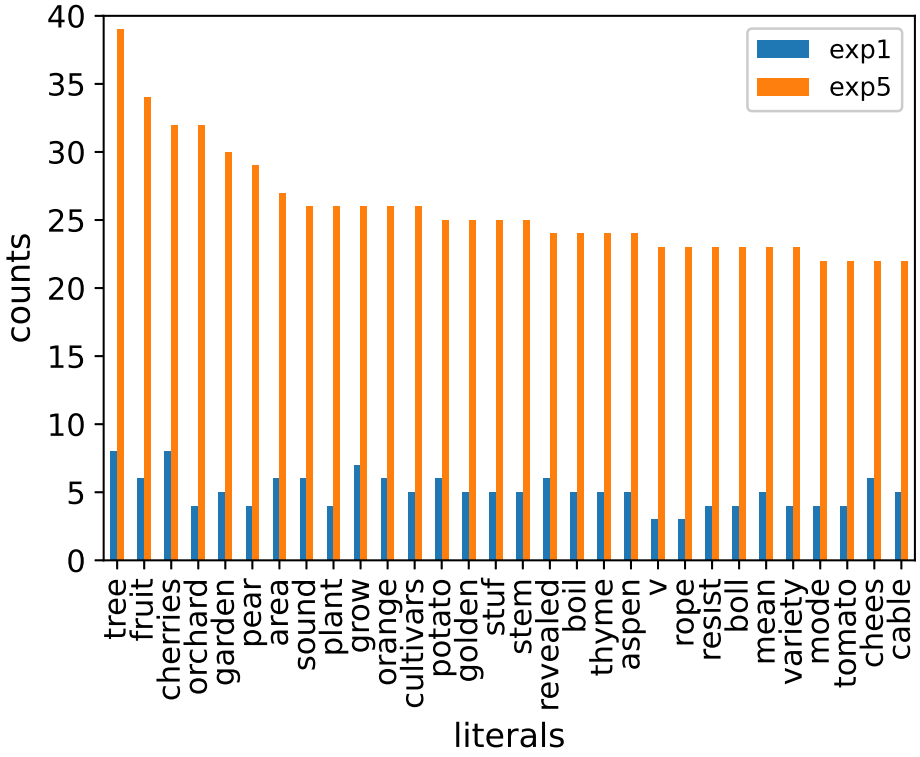


Figure A.6: Count of first 30 literals that are in negated form for classifying the sense of apple as company. (considered as important literals)

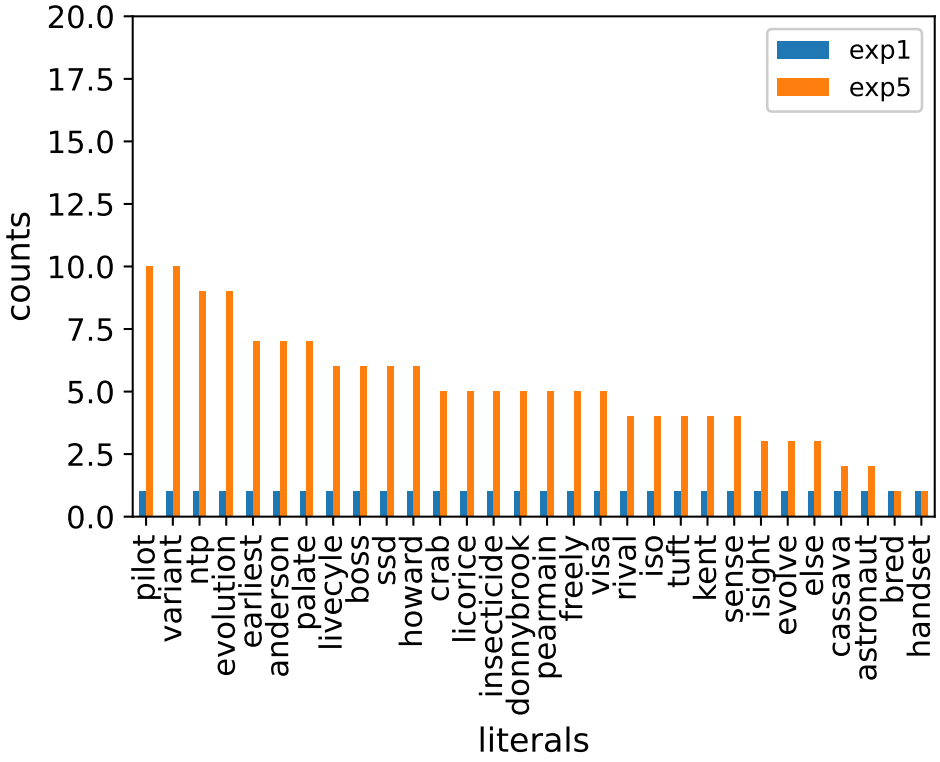


Figure A.7: Count of last 30 literals that are in negated form for classifying the sense of apple as company. (considered as non-important literals)

Since, interpretation of WSD is the main concern of the paper, we will compare our work with the latest benchmark [24]. Table A.2 show the comparison of Macro and Micro F1 score on CoarseWSD dataset for 4 different methods: FastText-Base (FTX-B), FastText-CommonCrawl (FTX-C), 1 neural network (NN) BERT base, and our proposed TM. FTX-B is a fast text linear classifier without pre-trained embeddings and FTX-C is a fast text linear classifier with pre-trained embedding from Common Crawl. These are considered as the standard baseline for this dataset [24]. Our proposed TM based WSD easily outperforms FTX-B baseline and is close to FTX-C without even considering the pretrained embedding. However, TM falls short of BERT's performance given that it is a huge language model that achieves the state-of-the-art performance on most of the task. This shows that TM not only possesses the interpretation of the WSD but also has performance close to the state of the art.

Datasets	Micro-F1				Macro-F1			
	FTX-B	FTX-C	BRT-B	TM	FTX-B	FTX-C	BRT-B	TM
Apple	96.3	97.8	99.0	97.58	96.6	97.7	99.0	97.45
JAVA	98.7	99.5	99.6	99.38	61.1	84.1	99.8	99.35
Spring	86.9	92.5	97.4	90.78	78.8	96.4	97.2	90.76
Crane	87.9	94.9	94.2	93.63	88.0	94.8	94.1	93.62

Table A.2: Results on the full CoarseWSD balanced dataset for 4 different models: FastText-Base (FTX-B), FastText-CommonCrawl (FTX-C), 1 Neural Network BERT-Base (BRT-B) and Tsetlin Machine (TM). Table cells are highlighted (dark blue to light blue) for better visualization of accuracy.

A.5 Conclusions

This paper proposed a sense categorization approach based on recently introduced TM. Although there are various methods for sense classification on CoarseWSD-balanced dataset with good accuracy, many machine learning algorithms fail to provide human interpretation that is used for explaining the procedure of particular classification. To overcome this issue, we present a TM-based sense classifier that learns the formulae form text corpus utilizing conjunctive clauses to demonstrate a particular feature of each category. Numerical results indicate that the TM based approach is human-interpretable and it achieves a competitive accuracy, which shows its potential for further WSD studies. In conclusion, we believe that the novel TM-based approach can have a significant impact on sense identification that is a very important factor in a chatbot or other WSD tasks.

A

Bibliography

[1] E. Agirre and P. Edmonds, "Word sense disambiguation: Algorithms and applications," in *Springer, Dordrecht*, 2007.

[2] R. Navigli, J. Camacho-Collados, and A. Raganato, "Word sense disambiguation: A unified evaluation framework and empirical comparison," in *EACL*, 2017.

[3] O. L. de Lacalle and E. Agirre, "A methodology for word sense disambiguation at 90% based on large-scale crowdsourcing," in *SEM@NAACL-HLT*, 2015.

[4] K. Liao, D. Ye, and Y. Xi, "Research on enterprise text knowledge classification based on knowledge schema," in *2010 2nd IEEE International Conference on Information Management and Engineering*, pp. 452–456, April 2010.

[5] Y. Wang, L. Wang, M. Rastegar-Mojarad, S. Moon, F. Shen, N. Afzal, S. Liu, Y. Zeng, S. Mehrabi, S. Sohn, and H. Liu, "Clinical information extraction applications: A literature review," *Journal of Biomedical Informatics*, vol. 77, pp. 34 – 49, 2018.

[6] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," 2018.

[7] D. Yuan, J. Richardson, R. Doherty, C. Evans, and E. Altendorf, "Semi-supervised word sense disambiguation with neural models," in *COLING*, 2016.

[8] R. Tripodi and M. Pelillo, "A game-theoretic approach to word sense disambiguation," *Computational Linguistics*, vol. 43, p. 31–70, Apr 2017.

[9] C. Hadiwinoto, H. T. Ng, and W. C. Gan, "Improved word sense disambiguation using pre-trained contextualized word representations," 2019.

[10] R. Navigli and P. Velardi, "Structural semantic interconnection: A knowledge-based approach to word sense disambiguation," in *SENSEVAL@ACL*, 2004.

[11] O. Dongsuk, S. Kwon, K. Kim, and Y. Ko, "Word sense disambiguation based on word similarity calculation using word vector representation from a knowledge-based graph," in *COLING*, 2018.

[12] Z. Zhong and H. T. Ng, "It makes sense: A wide-coverage word sense disambiguation system for free text," in *ACL*, 2010.

[13] G. A. Miller, M. Chodorow, S. Landes, C. Leacock, and R. G. Thomas, "Using a semantic concordance for sense identification," in *HLT*, 1994.

[14] K. Taghipour and H. T. Ng, "One million sense-tagged instances for word sense disambiguation and induction," in *CoNLL*, 2015.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *ArXiv*, vol. abs/1310.4546, 2013.

[16] M. F. Sadi, E. Ansari, and M. Afsharchi, "Supervised word sense disambiguation using new features based on word embeddings," *J. Intell. Fuzzy Syst.*, vol. 37, pp. 1467–1476, 2019.

[17] M. Kågebäck and H. Salomonsson, "Word sense disambiguation using a bidirectional LSTM," in *CogALex@COLING*, 2016.

[18] S. M. Rezaeinia, R. Rahmani, A. Ghodsi, and H. Veisi, "Sentiment analysis based on improved pre-trained word embeddings," *Expert Systems with Applications*, vol. 117, pp. 139–147, 2019.

[19] F. K. Khattak, S. Jeblee, C. Pou-Prom, M. Abdalla, C. Meaney, and F. Rudzicz, "A survey of word embeddings for clinical text," *Journal of Biomedical Informatics: X*, vol. 4, p. 100057, 2019.

[20] M. B. Lazreg, M. Goodwin, and O.-C. Granmo, "Combining a context aware neural network with a denoising autoencoder for measuring string similarities," *Computer Speech & Language*, vol. 60, p. 101028, 2020.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[22] S. Sonkar, A. E. Waters, and R. G. Baraniuk, "Attention word embedding," *arXiv preprint arXiv:2006.00988*, 2020.

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

[24] D. Loureiro, K. Rezaee, M. T. Pilehvar, and J. Camacho-Collados, "Language models and word sense disambiguation: An overview and analysis," 2020.

[25] V. Buhrmester, D. Münch, and M. Arens, "Analysis of explainers of black box deep neural networks for computer vision: A survey," 2019.

[26] O.-C. Granmo, "The Tsetlin machine - a game theoretic bandit driven approach to optimal pattern recognition with propositional logic," 2018.

[27] O.-C. Granmo, S. Glimsdal, L. Jiao, M. Goodwin, C. W. Omlin, and G. T. Berge, "The convolutional Tsetlin machine," 2019.

[28] R. K. Yadav, L. Jiao, O.-C. Granmo, and M. Goodwin, "Human-level interpretable learning for aspect-based sentiment analysis," in *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*, AAAI, 2021.

[29] B. Bhattarai, O.-C. Granmo, and L. Jiao, "Measuring the novelty of natural language text using the conjunctive clauses of a Tsetlin machine text classifier," *ArXiv*, vol. abs/2011.08755, 2020.

[30] G. T. Berge, O. Granmo, T. O. Tveit, M. Goodwin, L. Jiao, and B. V. Matheussen, "Using the Tsetlin machine to learn human-interpretable rules for high-accuracy text categorization with medical applications," *IEEE Access*, vol. 7, pp. 115134–115146, 2019.

[31] R. Saha, O.-C. Granmo, and M. Goodwin, "Mining interpretable rules for sentiment and semantic relation analysis using Tsetlin machines," in *Artificial Intelligence XXXVII*, pp. 67–78, Springer International Publishing, 2020.

[32] K. D. Abeyrathna, O.-C. Granmo, X. Zhang, L. Jiao, and M. Goodwin, "The regression Tsetlin machine: A novel approach to interpretable nonlinear regression," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 378, 2019.

[33] X. Zhang, L. Jiao, O.-C. Granmo, and M. Goodwin, "On the convergence of Tsetlin machines for the identity-and not operators," *arXiv preprint arXiv:2007.14268*, 2020.

Appendix B

Paper B

Title:	Human-Level Interpretable Learning for Aspect-Based Sentiment Analysis
Authors:	Rohan Kumar Yadav, Lei Jiao, Ole-Christoffer Granmo, and Morten Goodwin
Affiliation:	University of Agder, Faculty of Engineering and Science, 4879, Grimstad, Norway
Conference:	*35th AAAI Conference on Artificial Intelligence (AAAI)*, Online, Feb. 2021.
DOI:	.

B

Human-Level Interpretable Learning for Aspect-Based Sentiment Analysis

Rohan Kumar Yadav, Lei Jiao, Ole-Christoffer Granmo, and Morten Goodwin
Department of Information and Communication Technology
Faculty of Engineering and Science, University of Agder
4879, Grimstad, Norway
E-mails: {rohan.k.yadav, lei.jiao, ole.granmo, morten.goodwi}@uia.no

Abstract — **This paper proposes a human-interpretable learning approach for aspect-based sentiment analysis (ABSA), employing the recently introduced Tsetlin Machines (TMs). We attain interpretability by converting the intricate position-dependent textual semantics into binary form, mapping all the features into bag-of-words (BOWs). The binary-form BOWs are encoded so that the information on the aspect and context words are retained for sentiment classification. We further adopt the BOWs as input to the TM, enabling learning of aspect-based sentiment patterns in propositional logic. To evaluate interpretability and accuracy, we conducted experiments on two widely used ABSA datasets from SemEval 2014: Restaurant 14 and Laptop 14. The experiments show how each relevant feature takes part in conjunctive clauses that contain the context information for the corresponding aspect word, demonstrating human-level interpretability. At the same time, the obtained accuracy is on par with existing neural network models, reaching 78.02% on Restaurant 14 and 73.51% on Laptop 14.**

B.1 Introduction

Sentiment analysis, which identifies people's opinion on specific topics, is a classic problem in natural language processing (NLP). Under the umbrella of sentiment analysis, aspect-based sentiment analysis (ABSA), which is a fine-grained evaluation framework for sentiment classification [1], has become a hot research topic [2]. Among various tasks in ABSA, this paper focuses on the sentiment polarity (positive, neutral, negative) of a target word in given comments or reviews. For example, let us consider a review: *"Certainly not the best **sushi** in New York, however, it is always fresh and the **place** is very clean, sterile"*. The target word *"sushi"* is closely associated with its context words *"not best"*, assorting it as a negative polarity. The target word, *"place"*, is associated with its context words *"clean"* and *"sterile"*, classifying it as a positive sentiment. Such a complex form of sentiment classification is highly dependent on where the word appears in the sentence. To address this challenge, several recent approaches to ABSA have been based on attention mechanisms [3]. Although the accuracy of attention-based ABSA approaches are progressively improved, the interpretability of these models is still questionable, making them less trust-worthy. Not surprisingly, little research has been done on ABSA learning techniques that are interpretable at a human level [4].

Recently, interpretable AI has taken a big leap in industrial application [5]. Indeed, the scientific community has performed extensive research on ways to interpret neural networks. In a modern neural network, one can use the fact that the variants of attention [6] assign soft weights to the input representations, and then extract highly weighted tokens as rationales. However, these attention weights do not provide faithful explanations for classification [7, 8, 9, 10]. On the other hand, certain classic models, like Decision Trees, are particularly easy to understand, yet still compromise on accuracy compared with neural networks. Hence, an effective trade-off between accuracy and interpretability has still not been achieved.

In this article, we propose a Tsetlin Machine (TM) [11] based ABSA that employs a binary representation of the input features. The resulting architecture is *interpretable* and achieves competitive accuracy compared with state-of-the-art techniques. The ABSA task has two important inputs: a context word and an aspect word. Such aspect-based classification usually relies heavily on the position of the aspect word in the context. Such position information can be easily embedded in the neural network models. However, in TM, as all patterns and outputs are expressed in bits, learning and classification depend on bit manipulation, making it a challenging task to embed all the information into binary form. We therefore also aim to propose an extensive pre-processing approach for the ABSA inputs so that the binary form retains as much useful information as possible for the classification.

Our main contributions can be summarized as follows:

- We propose a novel pre-processing scheme to convert the ABSA inputs into binary form with limited information loss.
- We design an interpretable learning architecture using TM. The architecture offers human-level interpretable results with comparable classification accuracy.
- We employ additional knowledge from SentiWordnet [12] to enhance the accuracy of the architecture. It provides additional knowledge to the model and has significant impact on accuracy as explained later.

The remainder of the paper is organized as follows: We summarize related work in Section 2. The proposed pre-processing and TM architecture along with its learning process are described in Section 3. In Section 4, we report the experiment results and the comparisons with state-of-the-art. The interpretability of trained models is demonstrated in Section 5 before we conclude this work in Section 6.

B.2 Related Work

Sentiment analysis operates at three levels: document level, sentence level and aspect level. This work focuses on aspect level. Most of traditional supervised approaches depend heavily on handcrafted features to identify the sentiment of a word based on its context [13, 14]. However, these models fail to capture the semantic relatedness between the aspect word and its context. This problem gives rise to the attention-based models that are able to capture such a relationship [15, 6, 16, 17]. Furthermore, it is shown in [18] how an attention layer captures the weightage of the context words for predicting the sentiment of an aspect word. However, existing models cannot leverage the syntactic structure of the sentence, thereby making it difficult to distinguish various sentiments for multiple aspects of the sentence. To address this challenge, the RepWalk neural network model was recentely proposed [19]. It performs a replicated random walk on a syntax graph, effectively focusing on the descriptive contextual words.

Despite the fact that neural network-based models with attention, including BERT and contextualized embedding [6, 20, 21], capture the semantic relatedness among words in the context, they still lack interpretability. This arguably makes them black box models [22]. Many applications of attention mechanisms show, however, that a model can interpreted based on the weight assigned by the attention vector to each input, but they do not provide a faithful explanation of classification [7, 8]. Many researchers have attempted to replicate human learning behavior in neural networks [23], but have failed to answer the question of making the learning interpretable. In order to overcome the issue of interpretability in NLP, we explore the recently introduced Tsetlin Machine (TM), which recognizes patterns in the form of propositional logic [11, 24]. TMs have demonstrated promising results in various classification tasks involving numerical data, image data, text data, and board games [25, 26].

In this paper, we aim to reduce the gap between interpretability and accuracy with a significant margin on the ABSA task. To the best of our knowledge, this is the first study using TM to explore how each word in the context includes or excludes themselves to form conjunctive clauses for sentiment classification. Once the model is trained, clauses in the TM hold the information about which individual features in the context take part in the sentiment classification of the aspect word.

B.3 Methodology

B.3.1 Input Binarization

For both datasets, the ABSA tasks have a context word and an aspect word whose polarity is to be classified. Usually, the sentiment of the aspect word is reflected by its surrounding words in a

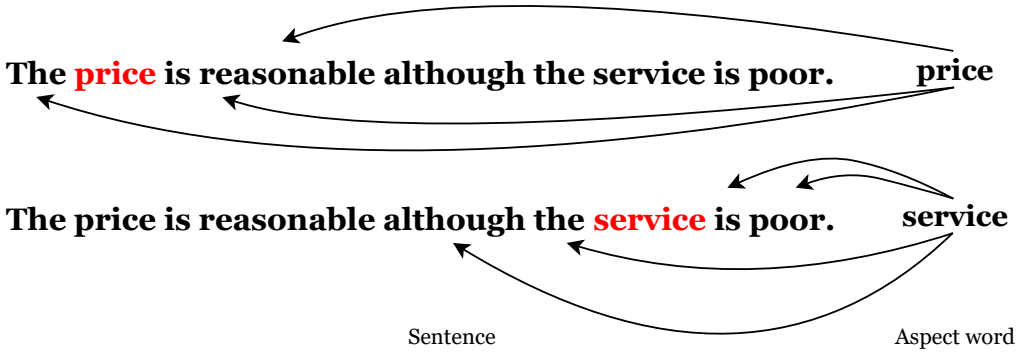


Figure B.1: Representation of an aspect word and its surrounding words.

sentence, as shown in Figure B.1. In this example, the aspect word "price" has positive sentiment due to the word "reasonable" in the context. Similarly, for "service", the context word "poor" describes its negative sentiment. This reveals that the sentiment of aspect words heavily relies on its position in a sentence and thus position embedding [27] is necessary. Such embedding creates a probability distribution of the sentence based on the aspect word. Recently, position-aware modelling has shown promising results on ABSA tasks [28].

Since TM requires binary inputs, to utilize TM for interpretability, the inputs must be binarized. It is challenging to incorporate the required position-based word relations in binary form, to allow for ABSA. In particular, since a TM does not employ any world knowledge like Word2vec [29], Elmo [21] or BERT [20], so as to retain the interpretability of the model, we reduce the size of vocabulary by replacing the sentiment carrying words with a common token. Understandably, without pre-trained embeddings, a model cannot find the similarity between two semantically related words such as "excellent" and "good". Hence, we adopt Opinion Lexicon [30], which is a list of English positive and negative sentiment words. In more details, we replace every possible word in the dataset by the common token "positive" or "negative", as shown in Figure B.2. Such external knowledge also helps to reduce the vocabulary size thereby decreasing the sparsity of BOW representations.

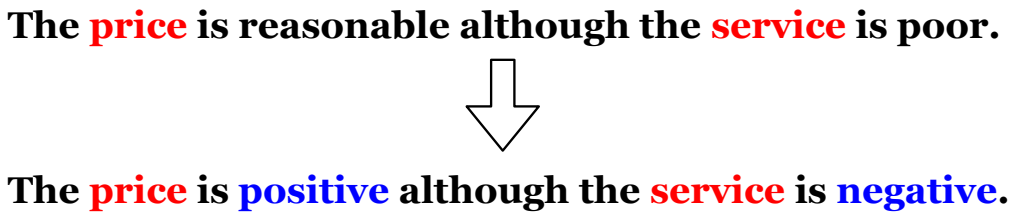


Figure B.2: Replacement of sentiment-carrying words with a common sentiment token using Opinion Lexicon.

Once the vocabulary size is determined, the context word and the aspect word can be converted into binary form, named as $BOW_{context}$ and BOW_{aspect} respectively. Since BOW in binary form does not consider the frequency of the replaced common tag (i.e., "positive" and "negative"), it becomes a rough representation of those tokens. In order to determine the location of these sentiment-carrying tokens, the sentence is split into two parts, divided by the aspect word. More specifically, we create additional binary vectors LOC^1_{vec} and LOC^2_{vec}, representing

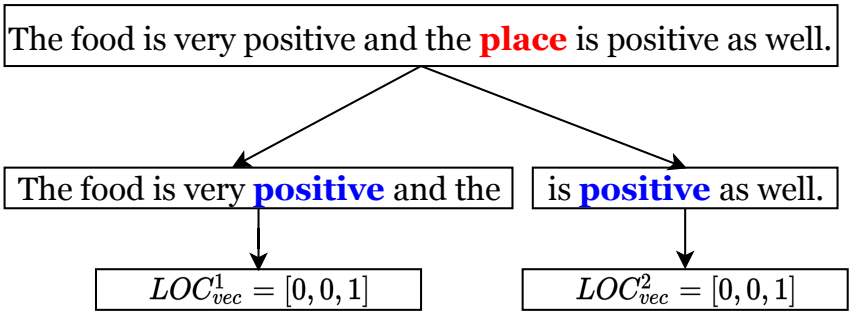


Figure B.3: 3-bit input feature representing the location of common sentiment-carrying tokens: negative, no sentiment, and positive.

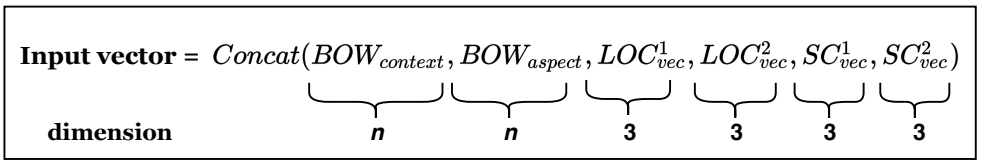


Figure B.4: Construction of binary input by concatenating all the pre-processed features.

the location of the common tokens. The dimension of LOC^1_{vec} and LOC^2_{vec} is three (the 1^{st} bit: negative, the 2^{nd} bit: no sentiment, the 3^{rd} bit: positive) as shown in Figure B.3. LOC^1_{vec} represents the presence of the common tokens "positive" or "negative" in the first part. If there are no sentiment tags, this is represented by "no sentiment". Similarly, LOC^2_{vec} represents the presence of the common tokens in the second part.

After the pre-processing of inputs, we use SentiWordNet to obtain the sentiment score (SC) of the 1^{st} part and the 2^{nd} part of the split sentence. This involvement of such additional knowledge enrich the input information. We adopt the sentiment score in a 3-D binary form for each part of the sentence. The SC vector SC^1_{vec} for the 1^{st} part of the context is given by Eq. (B.1). Similarly, vector SC^2_{vec} is utilized for the second part of the context.

$$SC^1_{vec} = \begin{cases} [0, 0, 1](positive), & \text{if } SC > 0, \\ [1, 0, 0](negative), & \text{if } SC < 0, \\ [0, 1, 0](no\ sentiment), & \text{if } SC = 0. \end{cases} \tag{B.1}$$

After processing all these binary representations, we concatenate them all to make a final input vector of size $(2n + 12)$ as shown in Figure B.4.

B.3.2 The Tsetlin Machine Based ABSA

TM is a recent classification method that manipulates expressions in propositional logic based on a team of Tsetlin Automata (TA) [11]. TA is a fixed structure deterministic automaton that learns the optimal action from a set of actions suggested by the environment. In TM, each input bit corresponds to two TAs, i.e., *TA* and *TA'*. *TA* controls the original bit of the input sample whereas *TA'* controls its negation. Here we use TA to represent a general Tsetlin automata that can be a *TA* or a *TA'*. Each TA corresponds to one literal. A literal here indicates an input bit or its negation. For example, if the bit represents the word "food", *TA* controls "food" itself and

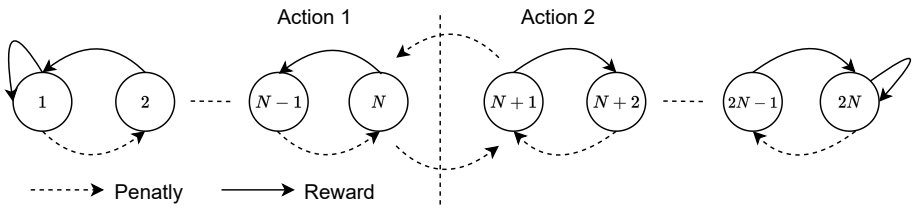


Figure B.5: The two-action TA and its transition in TM.

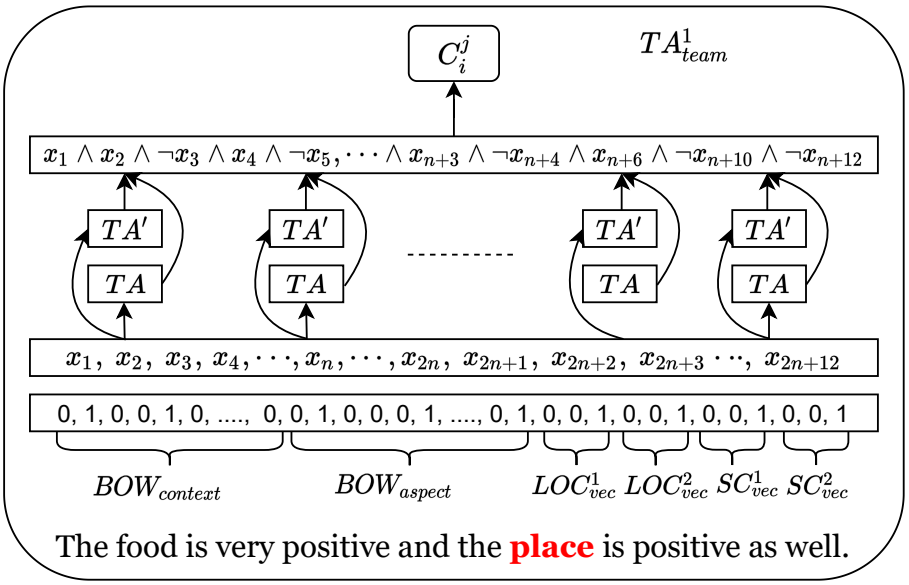


Figure B.6: TA team forms a Clause C_i^j by either including or excluding the input features.

then *TA'* handles "not food". Any TA employed by a TM has two actions with $2N$ states in total, as shown in Figure B.5. When it operates in states from 1 to N, action "exclude" is selected while action "include" is adopted for states from $N+1$ to $2N$. For each iteration, a TA performs "include" or "exclude" based on the current state. This in turn triggers a reward or penalty. If a reward is received, the TA moves to the deeper side of the action whereas if it obtains a penalty, it moves towards the center and eventually jumps to the other side of the action. Clearly, a TA, through its actions, decides whether to include or exclude its corresponding literal.

TM has a novel game theoretic strategy that regulates a decentralized team of TAs. This strategy guides the TAs to learn an arbitrarily complex propositional formula by including or excluding certain literals. More specifically, the included literals, by the operation of conjunction, formulate clauses. Each clause, after training, is expected to capture a sub-pattern. The overall pattern is decided by summing up the output of all clauses for any unknown input. The architecture for ABSA using TM is shown in Figs. B.6 and B.7.

Let us consider the input feature as a vector with a vocabulary size of n words, which is represented in BOW as $X_s = [x_1, x_2, x_3, \cdots, x_n, \cdots, x_{2n}, x_{2n+1}, x_{2n+2}, \cdots, x_{2n+12}]$ with $x_k \in \{0,1\}$ and $k \in \{1, \ldots, 2n+12\}$. Here, $[x_{2n+1}, x_{2n+2}, x_{2n+3}]$ and $[x_{2n+4}, x_{2n+5}, x_{2n+6}]$ represent LOC^1_{vec} and LOC^2_{vec} respectively. Similarly, $[x_{2n+7}, x_{2n+8}, x_{2n+9}]$ and $[x_{2n+10}, x_{2n+11}, x_{2n+12}]$ represent SC^1_{vec} and SC^2_{vec} respectively. Let q be the number of classes ($q = 3$ in ABSA task: positive, neutral and negative). If a pattern has m sub-patterns, the pattern can be captured using

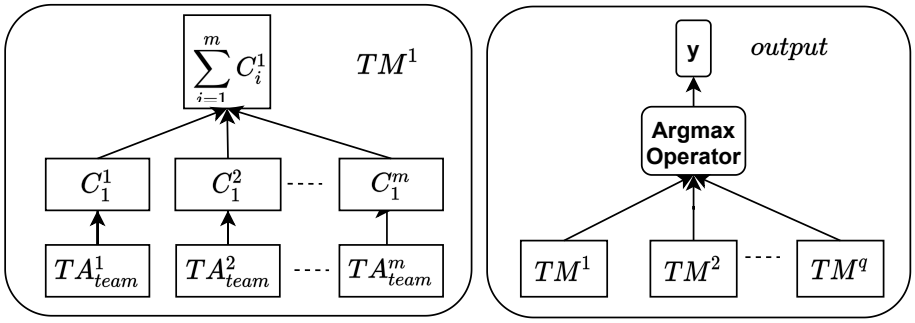


Figure B.7: (a). The sum of the votes for the clauses offers a score for a particular class. (b). Argmax operator decides the output class based on the score of the clauses in each class.

$q \times m$ conjunctive clauses C_i^j, $1 \leq j \leq q$, $1 \leq i \leq m$:

$$C_i^j = \left(\bigwedge_{k \in I_i^j} x_k \right) \wedge \left(\bigwedge_{k \in \bar{I}_i^j} \neg x_k \right), \tag{B.2}$$

where I_i^j and $\bar{I}_i^j$ are non-overlapping subsets of the input variable indices, $I_j^i, \bar{I}_j^i \subseteq \{1, \cdots, 2n+12\}, I_j^i \cap \bar{I}_j^i = \emptyset$. The subsets decide which of the input variables take part in the clause, and whether they are negated or not. The indices of input variables in I_j^i represent the literals that are included as is, while the indices of input variables in $\bar{I}_j^i$ correspond to the negated ones. Among m clauses in each class, clauses with odd indexes are assigned to positive polarity (+) whereas those with even indices are assigned to negative polarity (-). The clauses with positive polarity vote for the target class and those with the negative vote against it.

$$f^j(X_s) = \Sigma_{i=1,3,\ldots}^{m-1} C_i^j(X_s) - \Sigma_{i=2,4,\ldots}^{m} C_i^j(X_s). \tag{B.3}$$

For q number of classes, the final output y is given by the argmax operator to classify the input based on the highest sum of votes, as shown in Eq. (B.4).

$$y = \text{argmax}_j \left(f^j(X_s) \right). \tag{B.4}$$

B.3.3 The Learning Process of TM Based ABSA

In this section, we will detail the learning process of TM for the ABSA task. We explain the learning process with a walk-through of a specific sample context: "The food is very good and the place is clean as well", using the aspect word "place" whose sentiment is to be predicted. The context is first changed to "The food is very **positive** and the place is **positive** as well." For ease of explanation, we use the text word as a feature instead of the index in its binary form. For additional features, we will use the index of the binary input so as to differentiate the features that take part in classification. The indexes for additional features are LOC_{vec}^1 = $[2n+1, 2n+2, 2n+3]$, LOC_{vec}^2 = $[2n+4, 2n+5, 2n+6]$. Since the sentiment scores for both the first part of the context ("The food is very good and the") and that for the second part ("is clean as well") are greater than zero, we have SC_{vec}^1 =$[2n+7, 2n+8, 2n+9]$= [0,0,1], and SC_{vec}^2 = $[2n+10, 2n+11, 2n+12]$= [0,0,1], according to Eq. (B.1).

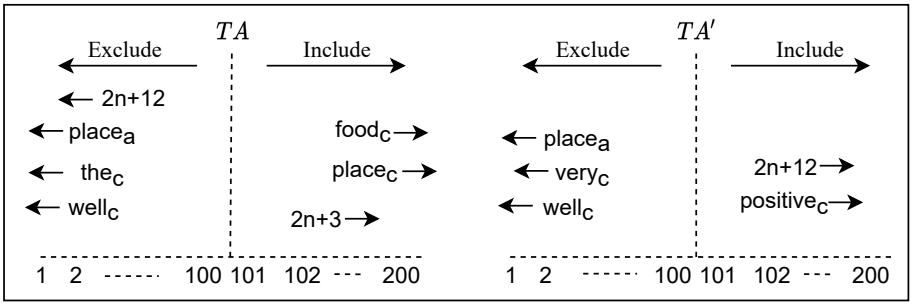


Figure B.8: TAs with 100 states per action that learn whether to exclude or include a specific word (or its negation), location of common token (or its negation) and the sentiment score information (or its negation) in a clause at time step 1.

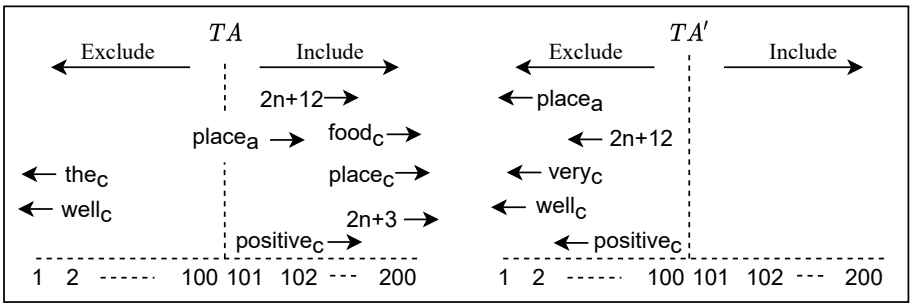


Figure B.9: TAs with 100 states per action that learn whether to exclude or include a specific word (or its negation), location of common token (or its negation) and the sentiment score information (or its negation) in a clause at time step t.

Figures B.8, B.9, and B.10 show the learning process of the ABSA task with the TM model. The subscripts c and a in the figures represent the word from context and aspect respectively. The *TA* or *TA'* that received reward will move away from the center while those that received penalty will move towards the center. In this way, the *TA* (or *TA'*) can be trained to either "include" or "exclude" a word (or its negation), helping the clauses, which are composed by the literals, learn different subpatterns. Consequently, the TM, composed by clauses, will gradually converge to the intended pattern. The feedback (reward or penalty) given to the TM follows two types: Type I and Type II feedback. Based on these feedback types, rewards or penalties are fed to the TA for the training samples. Type I Feedback is activated when a given input feature is either correctly assigned to the target sentiment (true positive) or mistakenly ignored (false negative). This feedback provides two countering effects: (1) involving more literals from the sample to refine the clauses; (2) trimming of the clauses by a factor specificity s that makes all clauses eventually evaluate to 1. The s-parameter is also responsible for avoiding overfitting. Type II Feedback is activated when an input feature is wrongly assigned to the target sentiment (false positive). It is responsible for introducing literals that make the clause evaluate to false, every time a false positive occurs. Type I Feedback and Type II Feedback are summarized in Tables B.1 and B.2 respectively.

Let us consider an example: a clause $C_1^1 = [food_c \wedge \neg positive_c \wedge place_c \wedge (2n{+}3) \wedge \neg(2n{+}12)]$ that is formed at time step $t = 1$, as shown in Figure B.8. Here, the time step indicates the instant the clause is updated during training iterations. The clause is composed by a combination

Input features	time	Clause formed and its learning in each step	Clause Output	Pred Class for true class = 1	Feedback type
the_c $food_c$ $very_c$ $positive_c$ $place_c$ $well_c$ $place_a$ $2n+3$ $2n+12$	1	$C_1^1 = food_c \wedge \neg positive_c \wedge place_c \wedge (2n+3) \wedge \neg(2n+12)$ $C_1^1 = 1 \wedge 0 \wedge 1 \wedge 1 \wedge 0$	$C_1^1 = 0$	$y = 0$	Feedback I
	2	$C_1^1 = food_c \wedge place_c \wedge (2n+3)$ $C_1^1 = 1 \wedge 1 \wedge 1$	$C_1^1 = 1$	$y = 0$	Feedback I
	3	$C_1^1 = food_c \wedge positive_c \wedge place_c \wedge (2n+3) \wedge (2n+12)$ $C_1^1 = 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1$	$C_1^1 = 1$	$y = 1$	Feedback I
	⋮	⋮	⋮	⋮	⋮
	t	$C_1^1 = food_c \wedge positive_c \wedge place_c \wedge place_a \wedge (2n+3) \wedge (2n+12)$ $C_1^1 = 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1$	$C_1^1 = 1$	$y = 1$	Feedback I

Figure B.10: The illustration of the clause update until reaching to an intended pattern at time step t.

Input	Clause	1		0	
	Literal	1	0	1	0
Include Literal	P(Reward)	$\frac{s-1}{s}$	NA	0	0
	P(Inaction)	$\frac{1}{s}$	NA	$\frac{s-1}{s}$	$\frac{s-1}{s}$
	P(Penalty)	0	NA	$\frac{1}{s}$	$\frac{1}{s}$
Exclude Literal	P(Reward)	0	$\frac{1}{s}$	$\frac{1}{s}$	$\frac{1}{s}$
	P(Inaction)	$\frac{1}{s}$	$\frac{s-1}{s}$	$\frac{s-1}{s}$	$\frac{s-1}{s}$
	P(Penalty)	$\frac{s-1}{s}$	0	0	0

Table B.1: The Type I Feedback.

of literals that are "included" by its associated TAs. At the current step, the excluded literals in this case (i.e, $2n+12, place_a, the_c, well_c$) are controlled by *TA*, and the negated literals (i.e, $place_a, very_c, well_c$) are governed by *TA'*. Clearly, this clause evaluates to 0, thereby contributing to predict class 0 despite the true class being 1, as shown in Figure B.10. This indeed triggers the Type I feedback. With Type I feedback, the reward or penalty for each literals is decided by Table B.1. Since the literal $\neg positive_c$ is included, its feature is 0 ($\neg 1$) and the clause output is 0. Therefore, it receives penalty for being included with the probability of $\frac{1}{s}$, making it slowly move towards the center and eventually jumping to the side with action "exclude". Similarly, the literal $\neg(2n+12)$ also receives the penalty with probability $\frac{1}{s}$, making it slowly moving towards the center, as well, eventually jumping to exclude action. Once this happens, the clause C_1^1 becomes $[food_c \wedge place_c \wedge (2n+3)]$ as shown in time step $t = 2$ that outputs 1, making a prediction of class 0 as depicted in Figure B.10. Table B.1 shows if the clause output is 1, the literals are of value 1, and the actions of the literals are "excluded", such literals obtain inaction or penalty with probability $\frac{1}{s}$ or $\frac{s-1}{s}$ respectively, making them slowly move towards

Input	Clause	1		0	
	Literal	1	0	1	0
Include Literal	P(Reward)	0	NA	0	0
	P(Inaction)	1.0	NA	1.0	1.0
	P(Penalty)	0	NA	0	0
Exclude Literal	P(Reward)	0	0	0	0
	P(Inaction)	1.0	0	1.0	1.0
	P(Penalty)	0	1.0	0	0

Table B.2: The Type II Feedback.

the center and eventually jump to "include" action. Once it happens, the clause becomes C_1^1 = $[food_c \wedge positive_c \wedge place_c \wedge (2n+3) \wedge (2n+12)]$ as shown in time step $t = 3$. Based on reward and penalty, TM reaches to the intended pattern at time step t by the arrangement of literals controlled by their respective TAs, as shown in Figure B.9. The final clause is given by C_1^1 = $[food_c \wedge positive_c \wedge place_c \wedge (2n+3) \wedge place_a \wedge (2n+12)]$. The clause will still obtain Type I feedback when more training samples are given and they reinforce the true positive occurrences until the sum of the votes by these clauses reaches a threshold parameter T.

The overall training and testing processes of TM-based ABSA are summarized in Algorithm 1 and Algorithm 2 respectively. For conciseness, we present, in Algorithm 1, the training procedure for the clauses with positive polarity, i.e., the clauses with odd index number. Clearly, the feedback types for the negative ones are just opposite. The complete training approach of a TM can be found in [11].

Once the class is predicted, we can explore its clauses for interpretability. The clauses that are triggered (i.e., $C_i^j(X_{s,te})$ = 1) are explored and their literals are converted into the original words for interpretation with the help of the additional information like LOC_{vec}^1, LOC_{vec}^2, SC_{vec}^1, or SC_{vec}^2.

B.4 Experiment Results

B.4.1 Datasets

The datasets are obtained from SemEval-2014 Task 4. The task has two domain-specific datasets, namely, Restaurant 14 (res14) and Laptop 14 (lap14). These datasets are provided with training and testing data. The statistics of the two datasets is shown in Table B.3. The code and the datasets are available online[1].

[1] https://github.com/rohanky/tm_absa

Algorithm 1 Training Process of TM based ABSA

Require: Given Input = [Context sentence, Aspect Word, Sentiment Score]

Pre-processed Input = Concat($BOW_{context}$, BOW_{aspect}, LOC^1_{vec}, LOC^2_{vec}, SC^1_{vec}, SC^2_{vec})
Final Input: $X_{s,tr}$ = [x_1, $\cdots$, x_n, $\cdots$, x_{2n}, $\cdots$, x_{2n+12}] and y ▷ y is the label of the input sample
Output: trained TM.
for Each training sample **do**
    $\hat{y}$ = TM($X_{s,tr}$, T, s) ▷ Current sentiment estimate for the input sample
    if $y = 1$ **then**:
        for each clause C^j_i with odd index **do**
            Use Type I Feedback($X_{s,tr}$, $\hat{y}$, T, C^j_i, s) to update all Tsetlin automata in C^j_i.
        end for
    else ▷ if $y = 0$
        for each clause C^j_i with odd index **do**
            Use Type II Feedback($X_{s,tr}$, $\hat{y}$, T, C^j_i, s) to update all Tsetlin automata C^j_i.
        end for
    end if
end for
return Trained TM.

Algorithm 2 Testing Process of TM based ABSA

Require: Given Input = $X_{s,te}$

Output: predicted class
$f^j(X_{s,te})$= 0, for all j
for all j **do** ▷ For all classes
    for all i in class j **do** ▷ For all clauses in this class
        $f^j(X_{s,te})$ = $f^j(X_{s,te})$+$(-1)^{i+1}C^j_i(X_{s,te})$
    end for
end for
return $\text{argmax}_j f^j(X_{s,te})$

B.4.2 Baselines

In our experiment, we evaluate the proposed method and compare it with related approaches for ABSA as baselines.

- **ContextAvg** averages the word embedding to form a context embedding [16].
- **LSTM** uses the last hidden vector of the LSTM for classification [31].
- **TD-LSTM** utilizes two LSTMs to learn the language model from the left and the right contexts of the aspect [16].
- **ATAE-BiLSTM** is an attention-based LSTM with Aspect Embedding model [32].

Dataset	Positive	Negative	Neutral	Total
res14 (train)	2164	807	637	3608
res14 (test)	728	196	196	1120
lap14 (train)	994	870	464	2238
lap14 (test)	341	128	169	638

Table B.3: The statistics of SemEval-2014 dataset.

- **MemNet** integrates the content and the position of the aspect word into a deep neural network [16].
- **RAM** is a multi-layer architecture where each layer consists of attention-based aggregation of word features and a GRU cell [33].
- **IAN** is an Interactive Attention Network model that calculates the attention weights of the word in its sentiment and aspect interactively [3].
- **PRET+MULT** uses two approaches of transfer knowledge from document level using pretraining and multitask training [34].
- **HCSN** proposes a Human-like Semantic Cognition network for the ABSA task, motivated by the human beings' reading cognitive process [23]. We show that performance of our proposed scheme is quite similar to this technique with high interpretability.
- **TNet** employs a CNN layer instead of attention layer to extract features from the transformed word representations originated from a bi-directional RNN layer [35].
- **AGDT** is an Aspect-Guided Deep Transition model that uses the given aspect to direct the sentence encoding from scratch with specially designed deep transition architecture. This model generates the aspect-based sentence representation and hence predicts sentiment more accurately [36].

B.4.3 Results

In our experiment, the main selling-point of the architecture is transparent learning and interpretability rather than accuracy. Better accuracy may be achieved when grid search is adopted. As we have used the integer weighted TM [37], the parameters available are the number of clauses, the threshold T, and the specificity s, which are configured as 700, 90×100, and 15 respectively for both datasets. For pre-processing of text, we substitute the short form to its full form, such as "isn't" to "is not". Additionally, we stem the words to reduce the vocabulary size created due to spelling mistakes and variants of words[2]. The remaining pre-processing procedure has already been explained before. We train the TM model on both the datasets for 100 epochs each.

Since the output sentiment label has imbalanced training samples, we use two evaluation metrics: Accuracy and Macro-F1 [38]. Following most of the related studied within the ABSA

[2]In this work, we adopt the Porter Stemmer.

Methods	Restaurant 14		Laptop 14	
	Accuracy	Macro-F1	Accuracy	Macro-F1
ContextAvg	71.5	58.0	61.5	53.9
LSTM	74.3	63.0	66.5	60.1
TD-LSTM	75.6	64.5	68.1	63.9
ATAE-BiLSTM	77.6	65.3	68.7	64.2
MemNet	76.9	66.4	68.9	62.8
RAM	78.5	68.5	72.1	68.4
IAN	78.6	NA	72.1	NA
PRET+MULT	79.1	69.7	71.2	67.5
HCSN	77.8	70.2	76.1	72.5
TNet	80.79	70.84	76.01	71.47
AGDT	78.85	NA	71.50	NA
TM based ABSA	78.02 (76.40 $\pm$ 1.0)	67.85 (64.01 $\pm$ 0.8)	73.51 (71.47 $\pm$ 0.9)	70.82 (67.48 $\pm$ 1.5)

Table B.4: Experiment results of various approaches for SemEval-2014 dataset. The upper results show the best reproducible accuracy and lower ones represent the mean and standard deviation of the last 50 epochs when running the model for five times.

task, we report the best reproducible results by running the ABSA TM for 100 epochs, as shown in Table B.4. We have reported the highest reproducible accuracy along with its mean and standard deviation obtained during 5 experiments. As we can see, Context2vec and LSTM perform quite poorly as they do not consider the aspect information when deciding the sentiment polarity. However, due to the consideration of left and right context information, TD-LSTM performs slightly better than LSTM. The variants of attention perform consistently better than LSTM and TD-LSTM. This is due to the fact that attention captures important information with regard to the aspect word. Other methods like RAM and MemNet perform slightly better because of the integrated memory in sentiment modeling. Another kind of the neural network-based model is HCSN. HCSN utilizes a human-being-like cognitive network for ABSA, which is motivated by the principles of human beings' reading cognitive processes. Its pre-reading, active reading, and post-reading technique mimics the human behavior, which is then fed to the GRU network. As interesting as it seems, the involvement of the neural network still brings this below human-level interpretation on what drives the model to make the decision. Our model, which offers a transparent view of the learning process, obtains quite similar or higher accuracy compared to HCSN and PRET+MULT techniques. However, the TNet architecture with a CNN layer, which extracts salient features from transformed word representation, achieves higher accuracy compared to TM. AGDT is a model that uses Aspect guided GRU along with Max pooling to obtain Aspect Concatenated Embedding. It obtains quite similar accuracy compared to TM on Restaurant 14, whereas accuracy is lower on Laptop 14. Note that we do not use any pre-trained word2vec or glove embedding for TM and our model still performs better than LSTM, TD-LSTM as well as attention based BilSTM for both datasets. The Macro-F1 score shows that TM does not only greedily learn a particular class but also creates a set of features for each and every class. Even though the performance of our proposed model does not outperform

the state-of-the-arts models, it reaches to comparable accuracy and Macro-F1 with transparent learning and interpretable prediction.

In addition to the above comparisons, we demonstrate here the necessity of including both $LOCs$ and SCs vectors. First we only used the $LOCs$ in the model and observed that the accuracy of the model reaches 76.51%. Secondly, we replaced $LOCs$ with SCs and the performance of the model decreased to 75%. This shows that both vectors add useful information when employed together thereby reaching the stated accuracy of 78.02%.

To compare the performance of TM with classical interpretable models such as Logistic Regression (LR), we use our preprocessed BOW as input to LR. We observed that the TM performs better than LR in terms of accuracy. LR obtains the accuracy of 75.38% as compared with TM's 78.02% on the Restaurant 14 dataset. Indeed, those two approaches operate based on different concepts. LR is trained by adjusting weights and bias. TMs, on the other hand, relates words using propositional logic to represent a class. Employing propositional logic for knowledge representation provides rules rather than a mathematical computation. This crucial difference between a rule-based approach and regression methods is explored in [39]. One can analyze why a LR model assigns a particular class to an input by inspecting the weights and bias. However, assigning them meanings requires understanding of the mathematical computation that LR carries out. Since TM creates a list of patterns for a particular class based on the interaction of aspect words and the sentiment words in the context, its conjunctive clauses hold information of words in a rule-based form. It is well-known that evaluating a conjunctive clause is particularly easy for humans, making them natively interpretable and easier to explain than LR.

B.5 Interpretability and Analysis

B.5.1 Characteristics of Clauses

In this section, we will explain one phenomenon of a TM after training with the datasets. When analyzing the clauses after training, we noticed that the TM employs more negated literals to form a clause. This is a bit counter intuitive as there should be, intuitively, more literals in their original forms than their negations in a clause. To explain this behavior, let us study two general sentences having positive sentiments for aspect word "laptop":

- This laptop is in excellent condition.
- Battery life of this laptop is better compared to other brands.

In this example, we assume a vocabulary containing both positive and negative sentiment words of size 6, V = [excellent, bad, condition, worst, costly, better, laptop]. When literals in their original forms are utilized to compose a clause, the two sentences require two clauses to follow the sentiment, i.e., $C_1 = [laptop \wedge excellent \wedge condition]$ and $C_2 = [battery \wedge better \wedge laptop \wedge others]$. On the contrary, when negated form of literals are employed, only one clause is sufficient to satisfy the sentiment in both sentences: $C_1 = [laptop \wedge \neg bad \wedge \neg worst \wedge \neg costly]$. As the negation is a more efficient way to represent a pattern in NLP, the trained TM employs naturally more negated literals to form a clause.

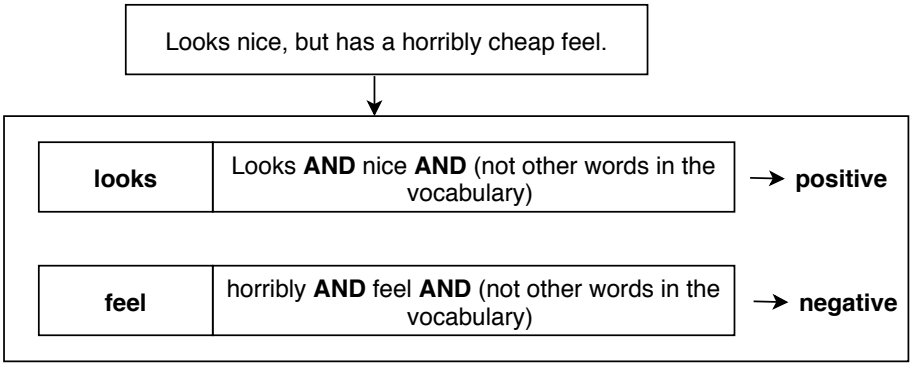


Figure B.11: Interpretation of a randomly selected sample from ABSA task.

B.5.2 A Case Study for Interpretability

In this case study, we demonstrate the interpretable result from trained model. We randomly select a sentence from the dataset as an example and demonstrate its literals that are responsible to form the clause. The selected sentence is "Looks nice, but has a horribly cheap feel." with a aspect word "looks" whose sentiment prediction of TM is positive. The sentence after pre-processing becomes "Looks positive, but has a negative negative feel." Among various clauses that are triggered by the given input, we randomly select a clause for interpretation. The clause is given by:

- $C_i^j = positive \wedge (2n+6) \wedge \neg(words\ not\ in\ the\ sentence\ and\ aspect)$.

The above clause can be interpreted as: the aspect word "looks" has positive sentiment because it has words "positive" and it lies in the second part of the sentence (indicated by $2n+6$, i.e., $LOC_{vec}^2 = [0, 0, 1]$) when split from aspect word "looks". Similarly, if the aspect word in the sentence is "feel" then its sentiment is predicted to be negative and a randomly selected clause is:

- $C_i^j = negative \wedge (2n+1) \wedge \neg(words\ not\ in\ the\ sentence\ and\ aspect)$.

This clause means that the sentiment is negative because it has words like "negative" and it lies in the first part of the sentence (indicated by $2n+1$, i.e $LOC_{vec}^1 = [1, 0, 0]$) when split from aspect word "feel". In both the cases, $\neg(words\ not\ in\ the\ sentence\ and\ aspect)$ represents the words in negated form that are presented in the input features. Finally, reversing back all the information and binarization to the original form of the words, we can obtain interpretation that shows the influence of words in the classification as in Figure B.11.

B.6 Conclusions

In this paper, we aim to reduce the gap between the interpretability and the accuracy of aspect based sentiment analysis (ABSA) by employing the recently introduced Tsetlin Machine (TM). Our proposed model embeds the aspect-based inputs into binary form for classifying the sentiment of a particular word in a sentence. Such binary representations are then fed to a TM architecture where the learning process is transparent, which gives a clear picture of what actually drives the TM to learn the particular sentiment for a given input. Additionally, we show the

involvement of words carrying the sentiment for the aspect words in the case study. In short, the proposed model successfully provides an human-interpretable learning approach on ABSA task with comparable accuracy.

B

Bibliography

[1] L. Zhang and B. Liu, *Sentiment Analysis and Opinion Mining*, pp. 1152–1161. Boston, MA: Springer US, 2017.

[2] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar, "SemEval-2014 task 4: Aspect based sentiment analysis," in *International Workshop on Semantic Evaluation (SemEval 2014)*, (Dublin, Ireland), pp. 27–35, ACL, 2014.

[3] D. Ma, S. Li, X. Zhang, and H. Wang, "Interactive attention networks for aspect-level sentiment classification," in *IJCAI*, (Melbourne, Australia), pp. 4068–4074, 2017.

[4] H. H. Do, P. Prasad, A. Maag, and A. Alsadoon, "Deep learning for aspect-based sentiment analysis: A comparative review," *Expert Systems with Applications*, vol. 118, pp. 272–299, 2019.

[5] W. Samek, G. Montavon, A. Vedaldi, L. Hansen, and K. Müller, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019.

[6] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, (California, USA), 2015.

[7] S. Serrano and N. A. Smith, "Is attention interpretable?," in *ACL*, (Florence, Italy), pp. 2931–2951, ACL, 2019.

[8] S. Wiegreffe and Y. Pinter, "Attention is not not explanation," in *EMNLP-IJCNLP*, (Hong Kong, China), pp. 11–20, ACL, 2019.

[9] G. Brunner, Y. Liu, D. Pascual, O. Richter, M. Ciaramita, and R. Wattenhofer, "On identifiability in transformers," in *ICLR*, (Addis Ababa, Ethiopia), 2020.

[10] S. Vashishth, S. Upadhyay, G. S. Tomar, and M. Faruqui, "Attention interpretability across NLP tasks," *arXiv*, vol. 1909.11218, 2019.

[11] O.-C. Granmo, "The tsetlin machine - a game theoretic bandit driven approach to optimal pattern recognition with propositional logic," *ArXiv*, vol. abs/1804.01508, 2018.

[12] A. Esuli and F. Sebastiani, "Sentiwordnet: A publicly available lexical resource for opinion mining," in *LREC*, (Genoa - Italy), 2006.

[13] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao, "Target-dependent Twitter sentiment classification," in *ACL*, (Portland, Oregon, USA), pp. 151–160, ACL, 2011.

[14] S. Kiritchenko, X. Zhu, C. Cherry, and S. Mohammad, "NRC-Canada-2014: Detecting aspects and sentiment in customer reviews," in *International Workshop on Semantic Evaluation (SemEval 2014)*, (Dublin, Ireland), pp. 437–442, ACL, 2014.

[15] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, "Disan: Directional self-attention network for rnn/cnn-free language understanding," in *AAAI*, (New Orleans, USA), 2018.

[16] D. Tang, B. Qin, and T. Liu, "Aspect level sentiment classification with deep memory network," in *EMNLP*, (Austin, Texas), pp. 214–224, ACL, 2016.

[17] J. Liu and Y. Zhang, "Attention modeling for targeted sentiment," in *EACL*, (Valencia, Spain), pp. 572–577, ACL, 2017.

[18] D. Tang, B. Qin, X. Feng, and T. Liu, "Effective LSTMs for target-dependent sentiment classification," in *COLING*, (Osaka, Japan), pp. 3298–3307, ACL, 2016.

[19] Y. Zheng, R. Zhang, S. Mensah, and Y. yi Mao, "Replicate, walk, and stop on syntax: An effective neural network model for aspect-level sentiment classification," in *AAAI*, (New York, USA), 2020.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, (Minneapolis, Minnesota), pp. 4171–4186, ACL, 2019.

[21] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *NAACL*, (New Orleans, Louisiana), pp. 2227–2237, ACL, 2018.

[22] C. Rudin, ""'stop explaining black box machine learning models for high stakes decisions and use interpretable models instead'"," *Nature Machine Intelligence*, vol. 1, pp. 206–215, 2018.

[23] Z. Lei, Y. Yang, M. Yang, W. Zhao, J. Guo, and Y. Liu, "A human-like semantic cognition network for aspect-level sentiment classification," in *AAAI*, (Hawaii, USA), 2019.

[24] X. Zhang, L. Jiao, O.-C. Granmo, and M. Goodwin, "On the convergence of tsetlin machines for the identity- and not operators," *arXiv*, vol. 2007.14268, 2020.

[25] O.-C. Granmo, S. Glimsdal, L. Jiao, M. Goodwin, C. W. Omlin, and G. T. Berge, "The convolutional tsetlin machine," *arXiv*, vol. 1905.09688, 2019.

[26] G. T. Berge, O.-C. Granmo, T. O. Tveit, M. Goodwin, L. Jiao, and B. V. Matheussen, "Using the tsetlin machine to learn human-interpretable rules for high-accuracy text categorization with medical applications," *IEEE Access*, vol. 7, pp. 115134–115146, 2019.

[27] S. Gu, L. Zhang, Y. Hou, and Y. Song, "A position-aware bidirectional attention network for aspect-level sentiment analysis," in *COLING*, (Santa Fe, New Mexico, USA), pp. 774–784, ACL, 2018.

[28] J. Zhou, Q. Chen, X. Huang, Q. Hu, and L. He, "Position-aware hierarchical transfer model for aspect-level sentiment classification," *Information Sciences*, vol. 513, pp. 1–16, 2020.

[29] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, (Stateline, United States), Curran Associates, Inc., 2013.

[30] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *ACM SIGKDD*, (New York, NY, USA), p. 168–177, ACM, 2004.

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[32] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based LSTM for aspect-level sentiment classification," in *EMNLP*, (Austin, Texas), pp. 606–615, ACL, 2016.

[33] P. Chen, Z. Sun, L. Bing, and W. Yang, "Recurrent attention network on memory for aspect sentiment analysis," in *EMNLP*, (Copenhagen, Denmark), pp. 452–461, ACL, 2017.

[34] R. He, W. S. Lee, H. T. Ng, and D. Dahlmeier, "Exploiting document knowledge for aspect-level sentiment classification," in *ACL*, (Melbourne, Australia), pp. 579–585, ACL, 2018.

[35] X. Li, L. Bing, W. Lam, and B. Shi, "Transformation networks for target-oriented sentiment classification," in *ACL*, (Melbourne, Australia), pp. 946–956, ACL, 2018.

[36] Y. Liang, F. Meng, J. Zhang, J. Xu, Y. Chen, and J. Zhou, "A novel aspect-guided deep transition model for aspect based sentiment analysis," in *EMNLP-IJCNLP*, (Hong Kong, China), pp. 5569–5580, ACL, 2019.

[37] K. D. Abeyrathna, O.-C. Granmo, and M. Goodwin, "Extending the tsetlin machine with integer-weighted clauses for increased interpretability," *arXiv*, vol. 2005.05131, 2020.

[38] C. D. Manning and H. Schütze, "Foundations of statistical natural language processing," in *SGMD*, 2002.

[39] M. Haghighi, S. Johnson, X. Qian, K. Lynch, K. Vehik, and a. T. S. G. S. Huang, "A comparison of rule-based analysis with regression methods in understanding the risk factors for study withdrawal in a pediatric study," *Scientific Reports*, vol. 6, 2016.

Appendix C

Paper C

Title: Positionless Aspect based Sentiment Analysis Using Attention Mechanism
Authors: Rohan Kumar Yadav, Lei Jiao, Ole-Christoffer Granmo, and Morten Goodwin
Affiliation: University of Agder, Faculty of Engineering and Science, 4879, Grimstad, Norway
Journal: *Knowledge-based System, Elsevier*, May, 2021.
DOI: 10.1016/j.knosys.2021.107136.

C

Positionless Aspect based Sentiment Analysis using Attention Mechanism

Rohan Kumar Yadav, Lei Jiao, Ole-Christoffer Granmo, and Morten Goodwin
Department of Information and Communication Technology
Faculty of Engineering and Science, University of Agder
4879, Grimstad, Norway
E-mails: {rohan.k.yadav, lei.jiao, ole.granmo, morten.goodwi}@uia.no

Abstract — **Aspect-based sentiment analysis (ABSA) aims at identifying fine-grained polarity of opinion associated with a given aspect word. Several existing articles demonstrated promising ABSA accuracy using positional embedding to show the relationship between an aspect word and its context. In most cases, the positional embedding depends on the distance between the aspect word and the remaining words in the context, known as the position index sequence. However, these techniques usually employ both complex preprocessing approaches with additional trainable positional embedding and complex architectures to obtain the state-of-the-art performance. In this paper, we simplify preprocessing by including polarity lexicon replacement and masking techniques that carry the information of the aspect word's position and eliminate the positional embedding. We then adopt a novel and concise architecture using two Bidirectional GRU along with an attention layer to classify the aspect based on its context words. Experiment results show that the simplified preprocessing and the concise architecture significantly improve the accuracy of the publicly available ABSA datasets, obtaining 81.37%, 75.39%, 80.88%, and 89.30% in restaurant 14, laptop 14, restaurant 15, and restaurant 16 respectively.**

C.1 Introduction

Aspect-based sentiment analysis (ABSA) is one of the sentiment analysis that aims to identify the polarity of aspect word associated with its context. It has been categorized as a standard evaluation framework for fine-grained sentiment analysis [1]. Among various aspect-based sentiment classification problems, we focus, in this study, on the task that is to map the polarity of the opinion on a aspect word into one of the following potential sentiments, namely, positive, neutral, or negative. For instance, the sentence "great food but the service was dreadful." has an aspect word "food" having a positive polarity and another aspect word "service" having a negative polarity in this context. ABSA includes various tasks including identification, classification, and aggregation. Most of the existing studies formulate ABSA as a classification problem where the information of aspect word is integrated [2]. Following the same stream of research, we also focus on the sentiment classification [3] in this article.

ABSA can be a quite challenging classification problem because of the ambiguity of sentiment in the sentence. The context-based feature usually plays an important role in the classification of sentiment, which introduces the hypothesis that the understanding of a word is mostly dependent on the context words and their locations. Hence both context words, as well as the position of the aspect word, become important features for sentiment classification [4, 5]. Understandably, even human beings spontaneously search for context words to evaluate the sentiment of a word when we read an article. This naturally makes the context and the position information vital features to be embedded into the deep learning model for better performance.

Various neural network architectures, from simple to complex ones, have been developed for position-aware sentiment classifications with a focus on the aspect word [6, 7]. A position encoding vector developed in [8] has been a popular choice for embedding positional information in Long Short term Memory (LSTM) based models. There the position index of the surrounding words is represented by the relative distance to the aspect word. Such position embedding creates a probability distribution among the context that is then embedded along with the word embedding of each word for classification of sentiments. However, a sophisticated neural network architecture is required for good performance because of the lack of sentiment lexicon knowledge with the integration of positional embedding [9]. Even a slight increment of accuracy in ABSA task usually requires a more complex architecture [10].

In this paper, we propose a very simple preprocessing of ABSA task by using sentiment lexicons and a masking technique that removes complex positional embedding thereby requiring a very straightforward architecture to obtain the state-of-the-art performance. As we know, human being usually makes sentiment classification of a particular word based on the surrounding words. Besides, human being, most probably, understands the meaning of each word and the sentiment associated with it as a priori. On the contrary, a neural network does not have this inbuilt knowledge. Even though various pre-trained word embedding captures the semantic relationship among the words, they are usually complicated. Therefore, it is important to find an efficient way to offer the model necessary knowledge, as priori, as much as possible. To give the model extra knowledge about sentiment in a simple way, we employ Opinion Lexicon [11] that has a list of positive and negative sentiment words. We use these lexicons and replace all the possible positive words with the "positive" tokens and negative words with "negative" tokens. The words that are not in the Opinion Lexicon will be left as they are. Additionally, to

avoid complex positional embedding, the aspect word is masked with a common token, making it a Masked Aspect Embedding and the original sentence as Sentence Embedding. Then, we adopt the Attention-based Bidirectional Gated Recurrent Unit (BiGRU) to train both the input to classify the sentiment of the masked aspect word. To evaluate the performance of the proposed methodology, we experiment with all available restaurant and laptop datasets of the ABSA task [12, 13, 14]. The numerical results show that the proposed scheme obtains either similar or higher accuracy compared with the state-of-the-art solutions that use positional embedding and a complex architecture.

The main contributions of the paper are summarized as follows:

1. Mask aspect words with common token and use it as aspect embedding along with original sentence embedding thereby removing the complex positional embedding.
2. Propose a very straightforward Attention based BiGRU architecture that performs either similar or better than the comparable state-of-the-art solutions.

The rest of the paper is organized as follows. We review related studies in Section 2. The proposed preprocessing and deep learning architecture are described in detail in Section 3. In Section 4, we show the experiment results and reveal the benefits of proposed schemes before concluding the paper in Section 5.

C.2 Related Work

This section consists of three parts. The first part includes the related studies on sentiment analysis in general. The second part surveys, in brief, ABSA tasks based on LSTM as encoder [15]. The last part reviews the attention based ABSA models that highly depend on the positional embedding [16].

C.2.1 Sentiment Analysis

Sentiment Analysis is a task involving polarity detection, subjectivity/objectivity identification as well as multi-modal fusion [17]. Sentiment analysis can be carried out in different levels, such as in document, sentence, or aspect level [18]. For document-level sentiment analysis, the goal is to detect the polarity of the whole document irrespective of any mentioned aspects. Tripathy et al. explored various machine learning algorithms on IMDB and polarity dataset demonstrating document level sentiment classification [19]. Other several large dataset has been explored to show that the character-level convolution networks could achieve state-of-the-art result [20]. A Linguistically Regularized LSTM is another variant of deep neural networks that can achieve competitive performance [21]. On the other hand, for sentence-level sentiment analysis, it has been developed in [22] a Bidirectional Emotional Recurrent Unit (BiERU) for Conversational Sentiment Analysis using generalized neural tensor block followed by a two-channel classifier. Various sentiment analysis tasks usually focus on analyzing data at the aggregate level, merely providing a binary classification (positive vs. negative), which does not account for finer characterization of emotion involved. On the contrary, a Multi-Level Fine-Scaled Sentiment Sensing with Ambivalence Handling is proposed for analyzing fine scale

of both positive or negative sentiments [23]. Such fine-grained sentiment classification mostly relies of the the weightage of word in the context.

Recently, the attention mechanism has shown promising performance in natural language processing (NLP) tasks, which improves deep neural network by letting them learn about where to focus. Recent studies on attention-based sentiment analysis are exampled by [24, 25, 26]. One of the applications of attention mechanism is the Attention-based Bidirectional CNN-RNN Deep Model (ABCDM) that extracts both past and future contexts by considering temporal information flow [27]. The attention mechanism in ABCDM is applied to the outputs of the bidirectional layers to shift the emphasis more or less on various words. On the other hand, some sentiment analysis studies focus not only on language modeling but also on common sense knowledge. For example, in [28], SenticNet 6 is proposed, which integrates top-down and bottom-up learning via an ensemble of symbolic and subsymbolic AI tools. However, these sentence-level sentiment analyses cannot be directly applied to the ABSA task where the sentiment of the sentence holds different opinions for distinct aspect words.

C.2.2 ABSA based on LSTM

ABSA tends to infer the polarity of a sentence's sentiment towards a particular aspect word. The sentiment may change throughout the sentence based on the context words. Hence, the main task is to model the relationship between the aspect word and the context words in an efficient manner. It is explained in [29] that around 40% classification error in this task is due to the ignorance of the aspect word. This significantly increased interests in the studies including early work on machine learning algorithms [30] that extracts a set of features to demonstrate the relationship between them.

Neural networks, such as the LSTM network, can encode sentences without feature engineering, and have been implemented in many (NLP) tasks [31, 32, 33]. In [34], TD-LSTM is proposed, which consists of two dependent LSTMs to model the left and the right contexts divided by the aspect word, where the aspect word is also input into the model as word embedding. Similarly, Gated Neural Networks is designed to control the importance of left and right context [35]. However, these methods do not capture the relationship between the context and the aspect word because the divided sentence most probably contains only one aspect word. Since the introduction of attention network on translation task [36, 37], many NLP tasks are interested to employ attention mechanism to model the relationship between the words in the sentence that seems very relevant to ABSA tasks. AE-LSTM adopts the attention mechanism to shift the focus of the aspect word towards relative context words [38]. Another work, similar to AE-LSTM, was proposed in [2] and it learns to interact between context words and aspect word based on associative relationships. In this way, the model can resiliently focus on the correct context words given the aspect word.

Although the attention mechanism has enhanced the efficiency of ABSA tasks, it simply processes the aspect word using the average pooling method while computing the attention score for the context. A typical drawback is that the performance suffers if the aspect consists of multiple words. To solve this problem, it is designed in [39] an Interactive Attention Network (IAN) to learn the attention representation for the context and the aspect word based on two different attention models in parallel and combining them eventually for sentiment classification. While

IAN is an important work that considered context and aspect words's interactive learning, it still utilizes average vectors to calculate the attention score for both aspect word and context words. In [40], it is presented a hierarchical model of attention for the task of aspect-based sentiment analysis that included both attention at the aspect level and attention at the sentence level, where the attention used in the aspect-level is a self-attention that takes the output of hidden layers as the input. Moreover, several other studies use knowledge-based approaches to tackle the ABSA task. It is proposed in [41] methods of rule-based ontology that constructed ontologies to help improve the outcome of ABSA by using common domain information. Additionally, to guide the model to learn relevant rules so that the model can capture more useful features, it is pertinent to incorporate external information. Such rule based models are highly interpretable compared to the neural network [42]. Those knowledge-based approaches, however, are very dependent on the knowledge that they possess, which may be difficult to construct, and the knowledge rule may also be intricate to design effectively through neural networks.

C.2.3 Positional embedding based ABSA

To enhance the classification accuracy in the ABSA task, the position information of the given aspect word is integrated into the models [4, 5, 43]. These methods utilize position between the aspect word and the context words either by counting the number of words between them or using tree structure dependency as relevant information. With the concept that the context word closer to the aspect word would be more important, attention mechanisms are preferred in the memory-based models [44]. Similarly, it is employed in [4] the word distance between the aspect word and the context word to mitigate the disadvantage of memory network [45]. The performance is further improved by scaling the input representation of the convolutional layer with the positional relevance between the contexts and the aspect word, which helps CNN feature extractor easily locate the sentiment indicators more accurately [5]. Another position based ABSA task is represented in [6] where position-aware sentence representation is applied by concatenating position embedding and word embedding. Similarly, it is proposed in [46] a position-dependent method using position-aware attention and a deep bidirectional LSTM (DBi-LSTM).

Despite the promising performance enhancement using positional embedding in ABSA tasks, we observed that the model is usually very complex to obtain the best range of classification accuracy. Additionally, since the RNN models are good enough to capture the time series information, encoding extra dimension as a position embedding seems a complicated preprocessing scheme. Therefore, we propose a masking technique that replaces the aspect word with a common token in the aspect embedding so that it creates different order information for distinct aspect words. We incorporate this masked aspect embedding along with the original sentence embedding into attention based Bi-GRU to capture the context-dependent sentiment, which is to be detailed in the next section.

C.3 Proposed Method

In this section, we describe in detail the proposed prepossessing and architecture for the ABSA task.

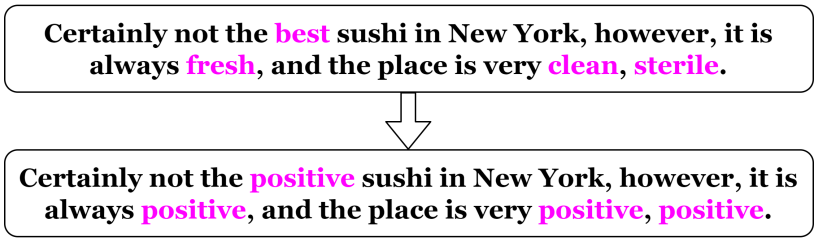


Figure C.1: Replacement of sentiment carrying word with a common tag using Opinion Lexicon.

C.3.1 Preprocessing

As mentioned earlier, the sentiment of aspect word highly depends on the context words surrounding it. Human beings can understand the meaning and the sentiment of context words that describe the aspect word. That is why human being can easily extract the sentiment of any particular word. On the contrary, a neural network does not have the knowledge that shows the semantic and syntactic relationship between words. Word2vec and Glove embedding [47, 48] capture the semantic relationship between the words but they are still far from human efficiency. Hence, we try to reduce the gap between the human knowledge and word embedding by making the semantically related word as the same token. To simplify the problem so that the neural networks can solve it better, we replace the sentiment-carrying words with the tag "positive" or "negative" based on Opinion Lexicon [11] as shown in Fig. C.1. Opinion Lexicon is a list of English words with positive or negative sentiment. Such use of external resource in preprocessing not only integrates sentiment knowledge but also reduces the vocabulary size that is a substantial concern by itself in NLP [49]. Altogether this process replaces around 550 words with the token "positive" and "negative".

Another important aspect of the preprocessing is to embed the position information of the aspect word. Traditional positional embedding considers the relative distance between aspect words and the context words in a sentence. Such embedding creates a probability distribution over the sentence with respect to the aspect word. However, such position embedding integrated with the input sentence is often initialized with trainable weights that increase the complexity of the model [50]. To mitigate this problem, we propose a simple masking technique that is based on the pattern learning behavior of the neural network. Usually, an ABSA task has two inputs: Sentence Embedding carrying the original sentence where position information is integrated and Aspect Embedding carrying aspect or aspect word. Here, we modify Aspect Embedding as Masked Aspect Embedding that carries the sentence with the aspect word masked by a common tag (here we call the common tag as "MASK"). We propose this preprocessing to remove the positional embedding required by Sentence Embedding. The modification between existing positional embedding and proposed masking technique is shown in Fig. C.2, where $pos(w^1)$ is the relative positional encoding of the first word with respect to the aspect word and the total number of words in the sentence is n. We hypothesize that the masked token is a common token present in every sample at a different location, creates a positional pattern. Since any machine learning model tries to capture the repetitive patterns, we hypothesize that the model will pick the masked token and its necessary context words around it to classify the sentiment. In all brevity, we propose a model that learns sentiment patterns for the position of the masked token. The overall preprocessed input is shown in Fig. C.3.

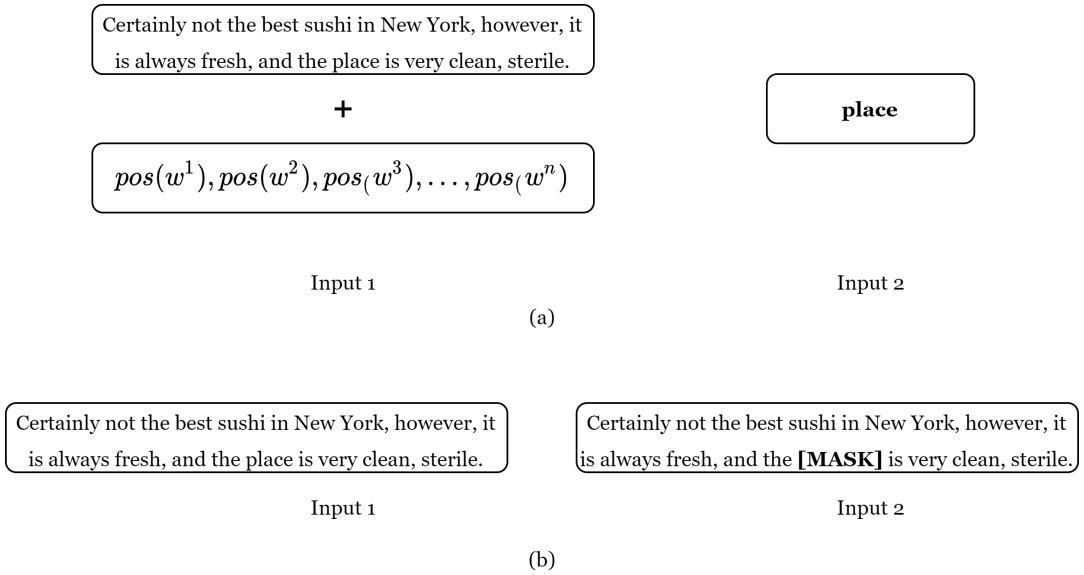


Figure C.2: (a). Existing approach of position embedding. (b). Proposed masking technique to learn pattern for the position.

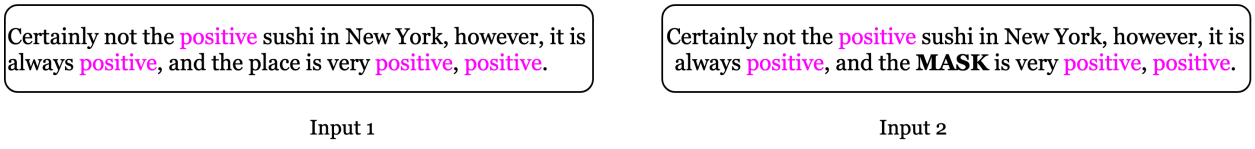


Figure C.3: Proposed preprocessed input.

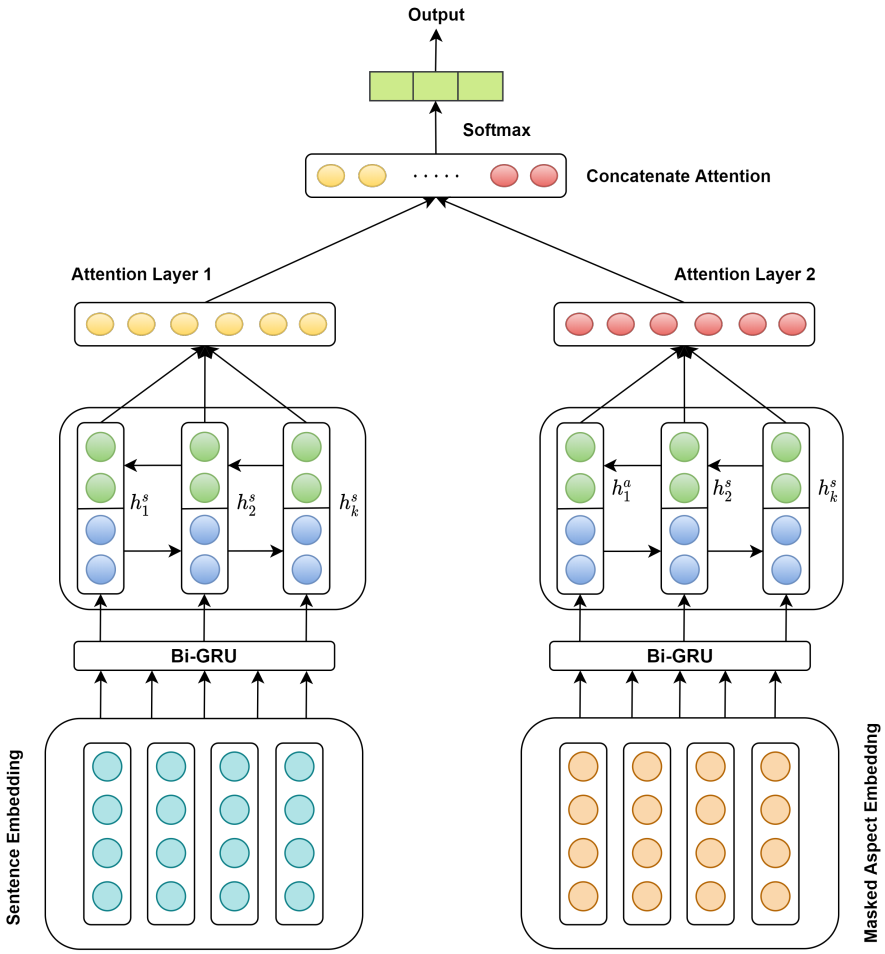


Figure C.4: Proposed Attention based Bi-GRU architecture.

C.3.2 Architecture description

The overall architecture of proposed model is shown in Fig. C.4, which consists 3 sections: Input Embedding, Bi-GRU, and Attention Layer. As the input embedding has been explained in the preprosessing part, we will focus on the latter two in the following paragraphs.

C.3.2.1 Bidirectional Gated Recurrent Unit (Bi-GRU)

Recurrent neural network (RNNs) [51] have been the baseline for NLP recently, where the internal states are utilized to process data sequentially. However, RNNs have certain limitations that lead to the development of their variants, such as LSTM and GRU. Here, we have explored both LSTM and GRU for sequencing modeling. Since we aim at developing a very concise and efficient model, we opt for GRU in our final architecture. The GRU controls the flow of information like the LSTM unit without employing a memory unit, which makes it more efficient with uncompromised performance compared to LSTM [52]. In addition, GRU mitigates the problem of vanishing gradients and gradient explosions in vanilla RNN.

Our proposed model consists of two Attention-based Bi-GRUs: GRU_1 for Sentence Embedding and GRU_2 for Masked Aspect Embedding. Both of them are identical in architecture that has similar learning pattern with the same hyperparameters. The only difference is how

the preprocessed input data is passed to these two separate Bi-GRUs. We assumed that GRU_2 captures the position of the masked token. Additionally, attention layer 2 gives the highest weightage to the masked token wherever it presents in the sentence. Similarly, GRU_1 is supposed to capture the context features from Sentence Embedding with attention layer 1, assigning higher weightage to the necessary context words. This hypothesis seems quite similar to how human being operates to understand the aspect-based sentiment.

Define $X = [x_1, x_2, x_3, \cdots, x_k]$ the Sentence Embedding (or Input 1), where k is the padded length of the sentence embedding to the forward layer of the GRU. There are two kinds of gates in GRU: the update gate and the reset gate. The update gate decides the amount of past information that needs to be brought into the current state and how much the new information is added. On the other hand, the reset gate takes care of how much information about the previous steps is written into the current candidate state h_t. Here, h_t is the output of the GRU at time step t and z_t represents the update gate. At a particular time step t, the new state h_t is given by:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t, \tag{C.1}$$

where $\odot$ is the element-wise multiplication and $\hat{h}_t$ is candidate activation. To update z_t, we have

$$z_t = \sigma \left(W_{z_t} x_t + U_{z_t} h_{t-1} + b_{z_t} \right). \tag{C.2}$$

Here, x_t is the word of the sentence at time step t that is plugged into the network unit and it is multiplied with its own weight W_{z_t}. Similarly, h_{t-1} holds the information of the previous unit and is multiplied with its own weight U_{z_t} and b_{z_t} is the bias associated with update state. The current state h_t can be updated using reset gate r_t by

$$\hat{h_t} = \tanh \left(W_h x_t + r_t \odot (U_h h_t) + b_h \right). \tag{C.3}$$

where W_h and U_h are weights associated with the candidate activation along with bais b_h.

At r_t, the candidate state of step t can get the information of input x_t and the status of h_{t-1} of step $t-1$. The update function of r_t is given by

$$r_t = \sigma \left(W_{r_t} x_t + U_{r_t} h_{t-1} + b_{r_t} \right), \tag{C.4}$$

where W_{r_t} and U_{r_t} are the weights associated with the reset state and b_{r_t} is the bias.

The Bi-GRU contains the forward GRU layer $(\overrightarrow{h_t})$ that reads the input sentence from step 0 to t and the backward GRU $(\overleftarrow{h_t})$.

$$\overrightarrow{h_t} = \overrightarrow{GRU}(x_t),\ t \in [1, T], \tag{C.5}$$

$$\overleftarrow{h_t} = \overleftarrow{GRU}(x_t),\ t \in [T, 1], \tag{C.6}$$

$$h_t = \left[\overrightarrow{h_t}, \overleftarrow{h_t} \right]. \tag{C.7}$$

C.3.2.2 Attention Layer

As we know that not all the words in the context have equal contribution for sentiment classification, an attention layer is assigned to prioritize important words in the context. Attention layer 1 is wrapped on top of GRU_1 to learn a weight α_t^1 for each hidden state h_t obtained at time step t. Since there are k inputs in the padded sequences, time step t will be from 1 to k. The weighting vector for attention layer 1, $\alpha_t^1 = [\alpha_1^1, \alpha_2^1, \alpha_3^1, \cdots, \alpha_k^1]$ is calculated based on the output sequence $H = [h_1, h_2, h_3, \cdots, h_k]$. The attention vector s_1 for attention layer 1 is calculated based on the weighted sum of these hidden states, as:

$$s_1 = \sum_{t=1}^{k} \left(\alpha_t^1 h_t\right), \tag{C.8}$$

where the weighted parameter α_t^1 is calculated by:

$$\alpha_t^1 = \frac{\exp\left(u_t^T u_w\right)}{\sum_t \exp\left(u_t^T u_w\right)}, \tag{C.9}$$

and $u_t = \tanh\left(W_w h_t + b_w\right)$. Here W_w and h_t are the weight matrices and b_w represents the bias. The parameter u_w represents context vector that is different at each step, which is randomly initialized and learned jointly during the training process.

Similarly, the attention layer 2 is wrapped on top of GRU_2 for assigning weightage to the masked token based on its position. The attention vector s_2 for attention layer 2 is given by:

$$s_2 = \sum_{t=1}^{k} \left(\alpha_t^2 h_t\right). \tag{C.10}$$

Finally, both of the attention layers are concatenated

$$s = Concatenate\left(s_1, s_2\right). \tag{C.11}$$

The concatenated layer is then sent to a fully connected layer and the softmax function generates a probability over c class labels.

C.4 Experiment Results and Evaluations

In this section, we will present the experiment results of the proposed scheme in detail. We conduct experiments on SemEval 2014 "restaurant" and "laptop", SemEval 2015 "restaurant" and SemEval 2016 "restaurant" dataset to verify the proposed hypothesis of lexicon addition and position-less masking. Additionally, we will also show the analysis of how the lexicon information and masking technique enhance the performance individually. We employ Keras [53] to implement our model. Adam [54] is adopted as the models' optimization method with the learning rate of $1 \times e^{-3}$. We also utilize Dropout [55] as the regularization strategy and the probability of Dropout is kept 0.6. Words are initialized with Glove [48] of 300-dimension word embedding. The batch size is 128 and is run for 100 epochs in the test datasets for obtaining the best results.

Dataset	Train			Test		
	Pos	Neu	Neg	Pos	Neu	Neg
Rest 14	2164	637	807	728	196	196
Lap 14	994	464	870	341	169	128
Rest 15	948	34	269	432	38	257
Rest 16	1289	63	457	474	29	123

Table C.1: Details of ABSA datasets.

C.4.1 Datasets

Our experiments are conducted on four publicly available ABSA datasets. Each sample of every dataset is a single sentence of a product review with aspect word and the corresponding sentiment label associated. While the datasets are given in the laptop domain by SemEval 2015 and SemEval 2016, they only contain the aspect category without the aspect word. The ”null” aspect terms are excluded from the datasets, and the ”dispute” or more than one sentiment labels are also excluded from the aspect terms in the analysis. The remaining sentences contain at least one aspect of the word with a {positive, neutral, negative} sentiment tag. The numerical details of the datasets are shown in Table C.1.

C.4.2 Compared Methods

To evaluate the performance of the proposed model on ABSA datasets, we consider the following approaches as comparative models. These models are proper baselines for ABSA and they are as close as possible to this work. Additionally, we have used the models that have exactly been evaluated in these specific four datasets of ABSA.

- **Feature+SVM** extracts n-gram as a feature, parse feature, and lexicon features to train the classifier [30].
- **ContextAvg** averages the word embedding to form a context embedding and then it is feed to the softmax function along with aspect vector [45].
- **LSTM** uses the last hidden vector information of the LSTM as a sentence representation for classifying aspect level sentiment [15].
- **TD_LSTM** utilizes two LSTMs to learn the language model from left and right contexts of the aspect respectively [45].
- **ATAE_BiLSTM** This model is similar to our approach, which is an Attention-based LSTM architecture with Aspect Embedding. It computes the aspect-specific weighted score of each word according to the representation of the aspect. The sums of the LSTM hidden outputs based on the attention weights are utilized to generate the sentence representation for ABSA classification [38].
- **IAN** is an Interactive Attention Network model that calculates the attention weights of the word in sentiment and aspect interactively to generate aspect and sentence representations [39].

Dataset	Restaurant 14		Laptop 14		Restaurant 15		Restaurant 16	
	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1
Majority	65.00	26.26	53.45	23.22	54.74	23.58	72.36	29.99
Feature+SVM	80.16	-	70.49	-	-	-	-	-
ContextAvg	71.53	58.02	61.59	53.92	73.79	47.43	79.87	55.68
LSTM	74.49	59.32	66.51	59.44	75.40	53.30	80.67	54.53
TD_LSTM	78.00	68.43	71.83	68.43	76.39	58.70	82.16	54.21
ATAE_BiLSTM	77.63	64.97	69.61	63.04	77.40	54.29	86.01	60.32
IAN	77.35	64.77	69.58	61.08	78.07	51.89	85.44	56.51
MemNet	78.16	65.83	70.33	64.09	77.89	59.52	83.04	57.91
RAM	78.48	68.54	72.08	68.43	79.98	60.57	83.88	62.14
Ont+LCR-Rot-hop	-	-	-	-	80.60	-	88.00	-
PBAN	81.16	-	74.12	-	-	-	-	-
PAHT	79.29	68.49	75.71	69.55	80.86	60.76	85.81	67.11
MTKFN	79.47	68.08	73.43	69.12	80.67	58.38	88.28	66.15
BERTADA-base	84.92	76.93	77.69	72.60	-	-	-	-
XLNetADA-base	85.84	78.35	79.89	77.78	-	-	-	-
Proposed Model	81.37	72.06	75.39	70.50	80.88	62.48	89.30	66.93

Table C.2: The state-of-the-art performance of ABSA on four datasets.

- **MemNet** integrates the content and the position of the aspect word into deep neural network [45].
- **RAM** is a multi-layer architecture where each layer consists of attention based aggregation of word features. A GRU cell is used to learn the sentence representation [4].
- **Ont+LCR-Rot-hop** uses a lexicon domain ontology and a rotatory attention mechanism to predict the sentiment of the aspect word [56].
- **PBAN** is a position-aware bidirectional attention network on bidirectional GRU. It also uses the mean pool and dot product to embed the position information of aspect word into sentence representation. It performs on par with the state of the art [6].
- **PAHT** is a position-aware hierarchical transfer model that models the position information from multiple levels to enhance the ABSA performance by transferring hierarchical knowledge from the resource-rich sentence-level sentiment classification (SSC) dataset [46].
- **MTKFN** is a Multi-source Textual Knowledge Fusing Network that incorporates knowledge from multiple sources to enhance the performance of ABSA. It uses pre-trained layers to extract contextual features and predicts the sentiment polarities. Additionally, it uses the information of conjunctions that captures the relationship between clauses and provides additional sentiment features [57].
- **BERTADA-base** is further trained on a domain-specific dataset and evaluated on the test set from the same domain [58].
- **XLNetADA-base** model is like BERTADA-base except for adopting XLNet [58].

Dataset	Restaurant 14		Laptop 14		Restaurant 15		Restaurant 16	
	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1
Lexicon replacement	79.55	67.31	71.32	64.80	78.82	57.86	87.38	63.66
Masked aspect embedding	80.62	70.72	73.51	68.01	80.33	61.26	88.65	65.00

Table C.3: Effect of the proposed preprocessing on all four datasets.

C.4.3 Hardware configuration

The experiments are conducted on a Linux platform. The OS and GPU configuration of our server is NVIDIA DGX Server Version 4.6.0 (GNU/Linux 4.15.0-121-generic x86_64). We have used NVIDIA Tesla V100 SXM3 32 GB to train our model.

C.4.4 Performance Comparison and Analysis

The comparison of our proposed model with recent similar studies is shown in Table C.2. These state-of-the-art studies are selected for comparison because they mostly depend on positional embedding as well as complex architecture, which are more relevant to our proposed model.

Among the baseline models that depend on language modeling, LSTM performs poorly on all four datasets. Above this lies ATAE_BiLSTM that has poor performance considering the fact that it utilizes the attention mechanism to model the aspect word. However, TD_LSTM performs better than the two models mentioned above because it considers both the left and right context as an aspect rather than the entire sentence. Similarly, MemNet performs better than IAN but it is not as good as RAM, because it does not use multiple attention mechanisms. Moreover, PAHT and MTKFN that utilize the hierarchical transfer model and external knowledge fusion respectively exhibit quite similar results as both of them are position-aware models. However, our proposed model surpasses both of these recent models with a significant margin using a much simpler architecture.

Among PBAN, PAHT and MTKFN, PBAN has higher accuracy than PAHT and MTKFN that is quite similar model to our proposed architecture. Hence, we focus on PBAN more than other listed models for comparison. PBAN that adopts two BiGRUs still falls behind in performance compared with our model. As shown in Table C.2, PBAN achieves 81.16% accuracy on restaurant 14 and 74.12% on laptop 14 dataset. However, it uses traditional embedding with a more complex model than ours. On the other hand, by employing the Opinion lexicon and masked aspect embedding, our model gives 81.37% and 75.39% accuracy on restaurant 14 and laptop 14 datasets respectively. Additionally, the macro-F1 score also outperforms the above-mentioned models with a significant margin except for the restaurant 16 dataset where the macro-F1 score is slightly below PAHT.

Even though the main aim of this paper is to design a simple yet effective model, to make a comprehensive comparison, we would present the performance of some new transformer-based models, such as BERTADA-base and XLNetADA-base approaches. It is quite obvious that this sophisticated pre-trained contextualized embedding achieves the upper state-of-the-arts results by using a softmax classifier. Since our model does not apply position embedding thereby reducing the trainable parameters, our comparison is mostly focused on the position-dependent model as explained before.

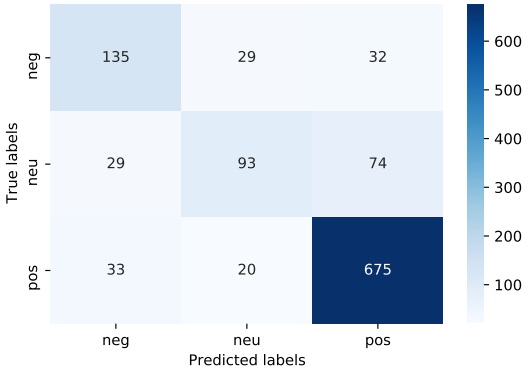


Figure C.5: Confusion Matrix of restaurant 14 dataset.

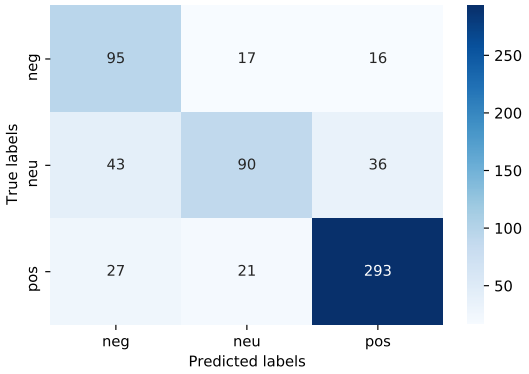


Figure C.6: Confusion Matrix of laptop 14 dataset.

C.4.5 Error and Sensitivity Analysis

Error analysis studies the impact of inaccuracy in the model for meaningful insight. In this study, we demonstrate error analysis using the confusion matrix. The confusion matrices show the relationship between the true and the predicted class as shown in Figs. C.5, C.6, C.7, and C.8. It can be seen from the confusion matrix that our model slightly suffers to identify the neutral sentiment. We believe that this may be because our model gives more focus to sentiment carrying lexicon tokens such as “positive” and “negative”.

In addition to the accuracy of the model, we also explore the sensitivity of the model with respect to input representations, detailed in the subsections below.

C.4.5.1 Effect of input representations

We here demonstrate the sensitivity of the model by changing the dimension of Glove input representation as shown in Table C.4. One can observe that the dimensions of the glove vectors have a significant impact on the performance of the model. However, the dimension of 200d and 300d does not have much difference except for restaurant 16 dataset. Hence, higher dimension representation has more semantic impact that boosts the accuracy of the model.

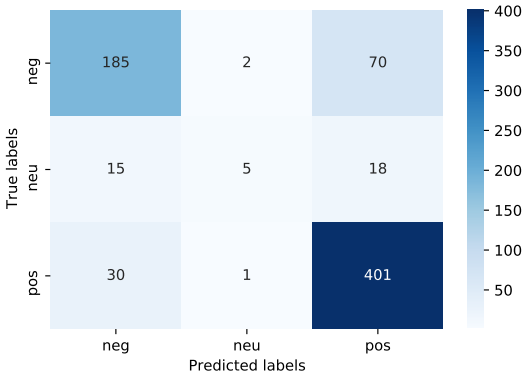


Figure C.7: Confusion Matrix of restaurant 15 dataset.

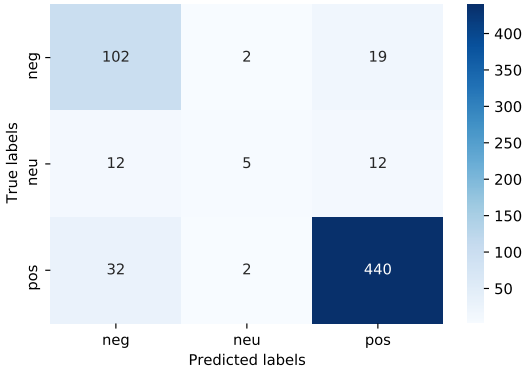


Figure C.8: Confusion Matrix of restaurant 16 dataset.

Glove	res14	lap14	res15	res16
glove.6B.50d	79.29	72.73	78.13	85.46
glove.6B.100d	80.18	72.29	78.82	86.42
glove.6B.200d	78.93	74.92	79.92	86.42
glove.6B.300d	79.73	73.67	80.33	86.26
glove.42B.300d	81.37	75.39	80.88	89.30
glove.840B.300d	80.00	74.14	79.09	87.54

Table C.4: Effect of various Glove vector for word representation on accuracy (%).

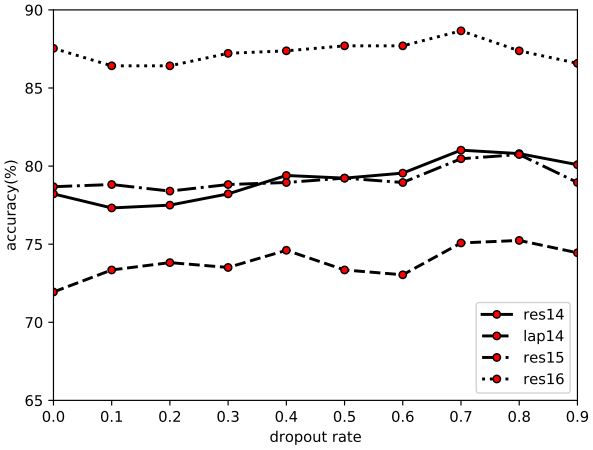


Figure C.9: Effect of dropout rate.

C.4.5.2 Effect of dropout rate

We here explore the effect of the dropout rate that is applied to the input of each layer of BiGRU. For simplicity, we employ the same dropout for both BiGRU varying from 0.0 to 0.9. The results of various dropout rates on the accuracy are shown in Fig. C.9. As we can see, the effect of dropout does not affect accuracy significantly. The change in accuracy is less than 3% in all 4 datasets. However, the best result always occurs at the dropout of range 0.6 to 0.8.

C.4.5.3 Effect of Opinion Lexicon and Masked Aspect Embedding

To verify the efficiency of proposed preprocessing, we further evaluate the performance of lexicon replacement and masking aspect words individually. The two main preprocessing used in our paper are :

- Lexicon replacement using Opinion Lexicon where Input 1 is Sentence embedding and Input 2 is aspect word embedding. (without masking aspect word)
- Masking aspect word where Input 1 is Sentence embedding and Input 2 is masked aspect embedding. (without lexicon replacement)

The performance of the scheme under these two conditions is evaluated for all four datasets and shown in Table C.3. As one can see, the effect of masking aspect word is significantly higher than the lexicon replacement. This is because the masking technique carries the position information into the model. On the contrary, the lexicon information helps to generalize the word and reduce the vocabulary size hence it has lower influence on the accuracy compared with the masking technique. Therefore, the lexicon replacement just boosts the final accuracy by a small margin.

C.4.6 Two-class sentiment classification

Some of the positional embedding based studies on ABSA also focus on the performance in the binary classification of ABSA task, in which it neglects the neutral class and makes it a

Model	Restaurant 14	Laptop 14
PBAN	91.67	87.81
Proposed Model	**92.51**	**90.15**

Table C.5: Comparison of binary classification on ABSA datasets.

binary classification as to predict positive or negative sentiment. Hence, we evaluate here the performance of our proposed model in binary classification and compare it with a position embedding based model [6]. The comparison is shown in Table C.5. We can see that our proposed method significantly outperforms the traditional positional embedding technique on both restaurant 14 and laptop 14 datasets.

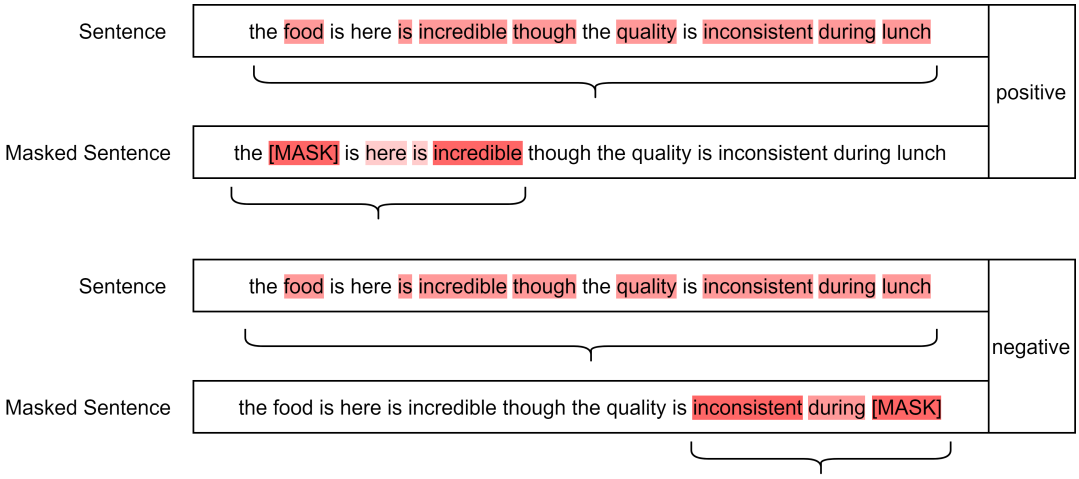


Figure C.10: Visualization of two typical examples. The red color represents the attentive weight of the word. A deeper color indicates a larger weight value.

C.4.7 Case studies

To have a detailed insight into why our proposed model performs better than the baselines with a straightforward architecture, we sample two examples from the restaurant 16 dataset and visualize attention heatmaps based on the trained model. Here we have two inputs for two separate BiGRUs: one being the sentence itself and the other being the masked sentence. As we have already discussed, the aspect words are masked with a common token, say "**MASK**". From Fig. C.10, we can see that the sentiment of the aspect word "food" is classified as positive sentiment from our model. The original sentence is fed to GRU_1 that has an attention layer 1 and the Masked sentence is fed to GRU_2 that has an attention layer 2. Here, attention layer 1 captures the context words throughout the sentence whereas attention layer 2 pays high attention to the masked token with weight narrowing down to important context words as shown in Fig. C.10. In the first example, attention layer 2 shifts the attention weightage towards the masked token "food" (the first half of the sentence) that highly depends on the context "incredible" for positive sentiment. For the second example, attention layer 2 shifts the attention weightage towards the masked token "lunch" (the second half of the sentence) whose sentiment depends on context

"inconsistent" for negative sentiment. This is a clear validation of our hypothesis that the masked token will hold the position information without using any additional trainable positional embedding. In all brevity, attention layer 1 assigns weightage to words in the sentence, and attention layer 2 narrows down the weightage from these selected words to important context words, carrying sentiment of the aspect word.

C.5 Conclusions

In this paper, we propose an efficient preprocessing scheme with an attention-based GRU model for aspect-based sentiment analysis. We first explore sentiment knowledge called Opinion Lexicon that is a list of positive, neutral, and negative sentiment words. In more detail, we replaced the words in ABSA dataset with a common tag, such as "positive" for positive sentiment words. This external input helps to bridge the gap from semantically related words to a certain extent and reduces the task's vocabulary. Since the ABSA is a position-dependent task, it requires the information of position along with the sentence embedding or aspect embedding. The extra trainable weights for position information increase the complexity of the model. To reduce the complexity, we proposed a masking technique that masks the aspect word in the sentence with a common token "MASK". This masked embedding is separately passed to the model along with the sentence embedding. Experimentally, we have shown that the proposed scheme performs at par with the state of the art and it outperforms several position-aware methods with very straightforward attention-based BiGRUs architecture.

Bibliography

[1] J. Zhao, K. Liu, and L. Xu, "Sentiment analysis: Mining opinions, sentiments, and emotions," *Computational Linguistics*, vol. 42, pp. 595–598, 2016.

[2] Y. Tay, L. A. Tuan, and S. C. Hui, "Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis," in *AAAI, New Orleans, Louisiana, USA*, 2018.

[3] K. Schouten and F. Frasincar, "Survey on aspect-level sentiment analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 813–830, 2016.

[4] P. Chen, Z. Sun, L. Bing, and W. Yang, "Recurrent attention network on memory for aspect sentiment analysis," in *EMNLP*, (Copenhagen, Denmark), pp. 452–461, ACL, Sept. 2017.

[5] X. Li, L. Bing, W. Lam, and B. Shi, "Transformation networks for target-oriented sentiment classification," in *ACL*, (Melbourne, Australia), pp. 946–956, ACL, July 2018.

[6] S. Gu, L. Zhang, Y. Hou, and Y. Song, "A position-aware bidirectional attention network for aspect-level sentiment analysis," in *COLING*, (Santa Fe, New Mexico, USA), pp. 774–784, 2018.

[7] B. Xu, X. Wang, B. Yang, and Z. Kang, "Target embedding and position attention with LSTM for aspect based sentiment analysis," in *International Conference on Mathematics and Artificial Intelligence*, ICMAI, (New York, NY, USA), p. 93–97, ACM, 2020.

[8] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao, "Relation classification via convolutional deep neural network," in *COLING, Dublin, Ireland*, p. 2335–2344, 2014.

[9] Y. Song, J. Wang, T. Jiang, Z. Liu, and Y. Rao, "Attentional encoder network for targeted sentiment classification," *ArXiv*, vol. abs/1902.09314, 2019.

[10] K. Xu, H. Zhao, and T. Liu, "Aspect-specific heterogeneous graph convolutional network for aspect-based sentiment classification," *IEEE Access*, vol. 8, pp. 139346–139355, 2020.

[11] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *ACM SIGKDD, New York, NY, United States*, p. 168–177, 2004.

[12] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar, "SemEval-2014 task 4: Aspect based sentiment analysis," in *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval Dublin, Ireland*, pp. 27–35, Aug. 2014.

[13] M. Pontiki, D. Galanis, H. Papageorgiou, S. Manandhar, and I. Androutsopoulos, "SemEval-2015 task 12: Aspect based sentiment analysis," in *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval, Denver, Colorado, USA*, pp. 486–495, 2015.

[14] M. Pontiki, D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, M. AL-Smadi, M. Al-Ayyoub, Y. Zhao, B. Qin, O. De Clercq, V. Hoste, M. Apidianaki, X. Tannier,

N. Loukachevitch, E. Kotelnikov, N. Bel, S. M. Jiménez-Zafra, and G. Eryiğit, "SemEval-2016 task 5: Aspect based sentiment analysis," in *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval, San Diego, California, USA*, pp. 19–30, 2016.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[16] J. Zhou, J. Huang, Q. Hu, and L. He, "Is position important? deep multi-task learning for aspect-based sentiment analysis," *Applied Intelligence*, vol. 50, pp. 3367–3378, 2020.

[17] E. Cambria, "Affective computing and sentiment analysis," *IEEE Intelligent Systems*, vol. 31, no. 2, pp. 102–107, 2016.

[18] K. Ravi and V. Ravi, "A survey on opinion mining and sentiment analysis: Tasks, approaches and applications," *Knowledge-Based Systems*, vol. 89, pp. 14–46, 2015.

[19] A. Tripathy, A. Anand, and S. K. Rath, "Document-level sentiment classification using hybrid machine learning approach," *Knowledge and Information Systems*, vol. 53, pp. 805–831, 2017.

[20] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *NIPS*, vol. 28, (Montréal CANADA), pp. 649–657, Curran Associates, Inc., 2015.

[21] Q. Qian, M. Huang, J. Lei, and X. Zhu, "Linguistically regularized LSTM for sentiment classification," in *ACL*, (Vancouver, Canada), pp. 1679–1689, ACL, July 2017.

[22] W. Li, W. Shao, S. Ji, and E. Cambria, "BIERU: Bidirectional emotional recurrent unit for conversational sentiment analysis," 2020.

[23] Z. Wang, S.-B. Ho, and E. Cambria, "Multi-level fine-scaled sentiment sensing with ambivalence handling," *Int. J. Uncertain. Fuzziness Knowl. Based Syst.*, vol. 28, pp. 683–697, 2020.

[24] Y. Lou, Y. Zhang, F. Li, T. Qian, and D. Ji, "Emoji-based sentiment analysis using attention networks," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 19, June 2020.

[25] M. Usama, B. Ahmad, E. Song, M. Hossain, M. Alrashoud, and M. Ghulam, "Attention-based sentiment analysis using convolutional and recurrent neural network," *Future Generation Computer Systems*, vol. 113, pp. 571–578, 2020.

[26] C. Xi, G. Lu, and J. Yan, "Multimodal sentiment analysis based on multi-head attention mechanism," in *International Conference on Machine Learning and Soft Computing, New York, NY, United States*, p. 34–39, 2020.

[27] M. E. Basiri, S. Nemati, M. Abdar, E. Cambria, and U. R. Acharya, "ABCDM: An attention-based bidirectional CNN-RNN deep model for sentiment analysis," *Future Generation Computer Systems*, vol. 115, pp. 279–294, feb 2021.

[28] E. Cambria, Y. Li, F. Z. Xing, S. Poria, and K. Kwok, *SenticNet 6: Ensemble Application of Symbolic and Subsymbolic AI for Sentiment Analysis*, p. 105–114. New York, NY, USA: ACM, 2020.

[29] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao, "Target-dependent Twitter sentiment classification," in *ACL*, (Portland, Oregon, USA), pp. 151–160, ACL, 2011.

[30] S. Kiritchenko, X. Zhu, C. Cherry, and S. Mohammad, "NRC-Canada-2014: Detecting aspects and sentiment in customer reviews," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, (Dublin, Ireland), pp. 437–442, ACL, 2014.

[31] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *INTERSPEECH, Portland, OR, USA*, pp. 194–197, 2012.

[32] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, (Cambridge, MA, USA), p. 3104–3112, MIT Press, 2014.

[33] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *EMNLP, Lisbon, Portugal*, pp. 1422–1432, 2015.

[34] D. Tang, W. Qin, X. Feng, and T. Liu, "Effective LSTMs for target-dependent sentiment classification," in *COLING, Osaka, Japan*, pp. 3298–3307, 2016.

[35] M. Zhang, Y. Zhang, and D.-T. Vo, "Gated neural networks for targeted sentiment analysis," in *AAAI, Phoenix, Arizona USA*, 2016.

[36] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2016.

[37] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *ArXiv*, vol. abs/1508.04025, 2015.

[38] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based LSTM for aspect-level sentiment classification," in *EMNLP, Austin, Texas, USA*, pp. 606–615, 2016.

[39] D. Ma, S. Li, X. Zhang, and H. Wang, "Interactive attention networks for aspect-level sentiment classification," in *IJCAI, Melbourne, Australia*, pp. 4068–4074, 2017.

[40] Y. Ma, H. Peng, and E. Cambria, "Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM," in *AAAI, New Orleans, Louisiana, USA*, pp. 5876–5883, 2018.

[41] K. Schouten, F. Frasincar, and F. de Jong, "Ontology-enhanced aspect-based sentiment analysis," in *Web Engineering* (J. Cabot, R. De Virgilio, and R. Torlone, eds.), pp. 302–320, Springer International Publishing, 2017.

[42] R. K. Yadav, L. Jiao, O.-C. Granmo, and M. Goodwin, "Human-level interpretable learning for aspect-based sentiment analysis," in *AAAI, Vancouver, Canada*, 2021.

[43] X. Li and W. Lam, "Deep multi-task learning for aspect term extraction with memory interaction," in *EMNLP*, (Copenhagen, Denmark), pp. 2886–2892, AACL, Sept. 2017.

[44] S. Sukhbaatar, a. szlam, J. Weston, and R. Fergus, "End-to-end memory networks," in *NIPS*, vol. 28, (Montréal CANADA), pp. 2440–2448, Curran Associates, Inc., 2015.

[45] D. Tang, B. Qin, and T. Liu, "Aspect level sentiment classification with deep memory network," in *EMNLP, Austin, Texas, USA*, pp. 214–224, 2016.

[46] J. Zhou, Q. Chen, X. Huang, Q. Hu, and L. He, "Position-aware hierarchical transfer model for aspect-level sentiment classification," *Information Sciences*, vol. 513, pp. 1–16, 2020.

[47] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS, Nevada, USA*, vol. 26, pp. 3111–3119, Curran Associates, Inc., 2013.

[48] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP, Doha, Qatar*, p. 1532–1543, 2014.

[49] W. Chen, Y. Su, Y. Shen, Z. Chen, X. Yan, and W. Y. Wang, "How large a vocabulary does text classification need? a variational approach to vocabulary selection," in *NAACL*, (Minneapolis, MN, USA), pp. 3487–3497, ACL, June 2019.

[50] C. W. Wu, "Prodsumnet: reducing model parameters in deep neural networks via product-of-sums matrix decompositions," *arXiv*, vol. abs/1809.02209, 2019.

[51] T. Mikolov, M. Karafi, and S. Khudanpur, "Recurrent neural network based language model," in *INTERSPEECH, Makuhari, Chiba, Japan*, 2010.

[52] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Workshop on Deep Learning@NIPS*, (Montréal, Canada), 2014.

[53] F. Chollet *et al.*, "Keras," 2015.

[54] D. P. Kingma and J. Ba, "ADAM: A method for stochastic optimization," in *ICLR 2015, San Diego, CA, USA, Conference Track Proceedings*, 2015.

[55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[56] O. Wallaart and F. Frasincar, "A hybrid approach for aspect-based sentiment analysis using a lexicalized domain ontology and attentional neural models," in *ESWC, Portoroz, Slovenia*, 2019.

[57] S. Wu, Y. Xu, F. Wu, Z. Yuan, Y. Huang, and X. Li, "Aspect-based sentiment analysis via fusing multiple sources of textual knowledge," *Knowledge-Based Systems*, vol. 183, p. 104868, 2019.

[58] A. Rietzler, S. Stabinger, P. Opitz, and S. Engl, "Adapt or get left behind: Domain adaptation through bert language model finetuning for aspect-target sentiment classification," *ArXiv*, vol. abs/1908.11860, 2020.

Appendix D

Paper D

Title:	Enhancing Interpretable Clauses Semantically using Pretrained Word Representation
Authors:	Rohan Kumar Yadav, Lei Jiao, Ole-Christoffer Granmo, and Morten Goodwin
Affiliation:	University of Agder, Faculty of Engineering and Science, 4879, Grimstad, Norway
Conference:	*4th Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, Punta Cana, Domincan Republic, Nov. 2021.
DOI:	10.18653/v1/2021.blackboxnlp-1.19.

Enhancing Interpretable Clauses Semantically using Pretrained Word Representation

Rohan Kumar Yadav, Lei Jiao, Ole-Christoffer Granmo, and Morten Goodwin
Department of Information and Communication Technology
Faculty of Engineering and Science, University of Agder
4879, Grimstad, Norway
E-mails: {rohan.k.yadav, lei.jiao, ole.granmo, morten.goodwi}@uia.no

Abstract — **Tsetlin Machine (TM) is an interpretable pattern recognition algorithm based on propositional logic, which has demonstrated competitive performance in many Natural Language Processing (NLP) tasks, including sentiment analysis, text classification, and Word Sense Disambiguation. To obtain human-level interpretability, legacy TM employs Boolean input features such as bag-of-words (BOW). However, the BOW representation makes it difficult to use any pre-trained information, for instance, word2vec and GloVe word representations. This restriction has constrained the performance of TM compared to deep neural networks (DNNs) in NLP. To reduce the performance gap, in this paper, we propose a novel way of using pre-trained word representations for TM. The approach significantly enhances the performance and interpretability of TM. We achieve this by extracting semantically related words from pre-trained word representations as input features to the TM. Our experiments show that the accuracy of the proposed approach is significantly higher than the previous BOW-based TM, reaching the level of DNN-based models.**

D.1 Introduction

Tsetlin Machine (TM) is an explainable pattern recognition approach that solves complex classification problems using propositional formulas [1]. Text- [2], numerical data- [3], and image classification [4] are recent areas of application. In Natural Language Processing (NLP), TM has provided encouraging trade-offs between accuracy and interpretability for various tasks. These include Sentiment Analysis (SA) [5, 6], Word Sense Disambiguation (WSD) [7], and novelty detection [8]. Because TM NLP models employ bag-of-words (BOW) that treat each word as independent features, it is easy for humans to interpret them. The models can be interpreted simply by inspecting the words that take part in the conjunctive clauses. However, using a simple BOW makes it challenging to attain the same accuracy level as deep neural network (DNN) based models.

A key advantage of DNN models is distributed representation of words in a vector space. By using a single-layer neural network, Mikolov et al. introduced such a representation, allowing for relating words based on the inner product between word vectors [9]. One of the popular methods is skip-gram, an approach that learns word representations by predicting the context surrounding a word within a given window length. However, skip-gram has the disadvantage of not considering the co-occurrence statistics of the corpus. Later, Pennington et al. developed *GloVe* – a model that combines the advantages of local window-based methods and global matrix factorization [10]. The foundation for the above vector representation of words is the distributional hypothesis that states that "the word that occurs in the same contexts tend to have similar meanings" [11]. This means that in addition to forming a rich high-dimensional representation of words, words that are closer to each other in vector space tend to represent similar meaning. As such, vector representations have been used to enhance for instance information retrieval [12], name entity recognition [13], and parsing [14].

The state of the art in DNN-based NLP has been advanced by incorporating various pre-trained word representations such as GloVe [10], word2vec [9], and fasttext [15]. Indeed, building semantic representations of the words has been demonstrated to be a vital factor for improved performance. Most DNN-based models utilize the pre-trained word representations to initialize their word embeddings. This provides them with additional semantic information that goes beyond a traditional BOW.

However, in the case of TM, such word representations cannot be directly employed because they consist of floating-point numbers. First, these numbers must be converted into Boolean form for TM to use, which may result in information loss. Secondly, replacing the straightforward BOW of a TM with a large number of floating-point numbers in fine-grained Boolean form would impede interpretability. In this paper, we propose a novel pre-processing technique that evades the above challenges entirely by extracting additional features for the BOW. The additional features are found using the pre-trained distributed word representations to identify words that enrich the BOW, based on cosine similarity. In this way, TM can use the information from word representations for increasing performance, and at the same time retaining the interpretability of the model.

The rest of the paper is organised as follows. We summarize related work in Section D.2. The proposed semantic feature extraction for TM is then explained in Section D.3. In Section D.4, we present the TM architecture employing the proposed feature extension. We provide extensive

experiment results in Section D.5, demonstrating the benefits of our approach, before concluding the paper in Section A.5.

D.2 Related Work

Conventional text classification usually focuses on feature engineering and classification algorithms. One of the most popular feature engineering approaches is the derivation of BOW features. Several complex variants of BOW have been designed such as n-grams [16] and entities in ontologies [17]. Apart from BOW approaches, Tang et al. demonstrated a new mechanism for feature engineering using a time series model for short text samples [18]. There are also several techniques to convert text into a graph and sub-graph [19, 20]. In general, none of the above methods adopt any pre-trained information, hence have inferior performance.

Deep learning-based text classification either depends on initializing models from pre-trained word representations, or on jointly learning both the word- and document level representations. Various studies report that incorporating such word representations, embedding the words, significantly enhances the accuracy of text classification [21, 22]. Another approach related to pre-trained word embedding is to aggregate unsupervised word embeddings into a document embedding, which is then fed to a classifier [23, 24].

Despite being empowered with world knowledge through pre-trained information, DNNs such as BERT [25] and XLNet [26] can be very hard to interpret. One interpretation approach is to use attention-based models, relying on the weights they assign to the inputs. However, more careful studies reveal that attention weights in general do not provide a useful explanation [27, 28]. Researchers are thus increasingly shifting focus to other kinds of machine learning, with the TM being a recent approach considered to provide human-level interpretability [2, 1, 5]. It offers a very simple model consisting of multiple Tsetlin Automata (TAs) that select which features take part in the classification. However, despite promising performance, there is still a performance gap to the DNN models that utilize pre-trained word embedding. Yet, several TM studies demonstrate high degree of interpretability through simple rules, with a marginal loss in accuracy [5, 7, 6].

A significant reason for the performance gap between TM-based and state-of-the-art DNN-based NLP models is that TM operates on Boolean inputs, lacking a method for incorporating pre-trained word embeddings. Without pre-trained information, TMs must rely on labelled data available for supervised learning. On the other hand, incorporating high-dimensional Booleanized word embedding vectors directly into the TM would significantly reduce interpretability. In this paper, we address this intertwined challenge. We propose a novel technique that boosts the TM BOW approach, enhancing the BOW with additional word features. The enhancement consists of using cosine similarity between GloVe word representations to obtain semantically related words. We thus distill information from the pre-trained word representations for utilization by the TM. To this end, we propose two methods of feature extension: (1) using the k nearest words in embedding space and (2) using words within a given similarity threshold, measured as cosine angle (θ). By adopting the two methods, we aim to reduce the current performance gap between interpretable TM and black-box DNN, by achieving either higher or similar accuracy.

D.3 Boosting TM BOW with Semantically Related Words

Here, we introduce our novel method for boosting the BOW of TM with semantically related words. The method is based on comparing pre-trained word representations using cosine similarity, leveraging distributed word representation. There are various distributional representations of words available. These are obtained from different corpora, using various techniques, such as word2vec, GloVe, and fastText. We here use GloVe because of its general applicability.

D.3.1 Input Feature Extraction from Distributed Word Representation

Distributed word representation does not necessarily derive word similarity based on synonyms but based on the words that appear in the same context. As such, the representation is essential for NLP because it captures the semantically interconnecting words. Our approach utilizes this property to expand the range of features that we can use in an interpretable manner in TM.

Consider a full vocabulary W of m words, $W = [w_1, w_2, w_3 \ldots, w_m]$. Further consider a particular sentence that is represented as a Boolean BOW $X = [x_1, x_2, x_3, \ldots, x_m]$. In a Boolean BOW, each element x_r, $r = 1, 2, 3, \ldots, m$, refers to a specific word w_r in the vocabulary W. The element x_r takes the value 1 if the corresponding word w_r is present in the sentence and the value 0 if the word is absent. Assume that n words are present in the sentence, i.e., n of the elements in X are 1-valued. Our strategy is to extract additional features from these by expanding them using cosine similarity. To this end, we use a GloVe embedding of each present word $w_r, r \in \{z|x_z = 1, z = 1, 2, 3 \ldots, m\}$. The embedding for word w_r is represented by vector $w_r^e \in \Re^d$, where d is the dimensionality of the embedding (typically varying from 25 to 300).

We next introduce two selection techniques to expand upon each word:

- Select the top k most similar words,
- Select words up to a fixed similarity angle $\cos(\theta) = \phi$.

For example, let us consider two contexts: “very good movie” and “excellent film, enjoyable”. Figs. D.1 and D.2 list similar words showing the difference between top k words and words within angle $\cos(\theta)$, i.e., ϕ. In what follows, we will explain how these words are found.

D.3.2 Similar Words based on Top *k* Nearest Words

We first boost the Boolean BOW of the considered sentence by expanding X with $(k-1) \times n$ semantically related words. That is, we add $k-1$ new words for each of the n present words. We do this by identifying neighbouring words in the GloVe embedding space, using cosine similarity between the embedding vectors.

Consider the GloVe embedding vectors $W_G^e = [w_1^e, w_2^e, \ldots, w_m^e]$ of the full vocabulary W. For each word w_r from the sentence considered, the cosine similarity to each word w_t, $t = 1, 2, \ldots, m$, of the full vocabulary is given by Eq. (A.1),

$$\phi_r^t = \cos(w_r^e, w_t^e) = \frac{w_r^e \cdot w_t^e}{||w_r^e|| \cdot ||w_t^e||}. \tag{A.1}$$

Clearly, ϕ_r^t is the cosine similarity between w_r^e and w_t^e. By calculating the cosine similarity of w_r to the words in the vocabulary, we obtain m values: ϕ_r^t, $t = 1, 2, \ldots, m$. We arrange these values in a vector Φ_r:

$$\Phi_r = [\phi_r^1, \phi_r^2, \ldots, \phi_r^m]. \tag{A.2}$$

The k elements from Φ_r of largest value are then identified and their indices are stored in a new set A_r.

Finally, a boosted BOW, referred to as X_{mod}, can be formed by assigning element x_t value 1 whenever one of the A_r contains t, and 0 otherwise:

$$X_{mod} = [x_1, x_2, x_3, \ldots, x_m], \tag{A.3}$$
$$x_t = \begin{cases} 1 & \exists r, t \in A_r \\ 0 & \nexists r, t \in A_r. \end{cases}$$

In addition, the vocabulary size for a particular task/dataset can be changed accordingly, which is usually less than m. Note that implementation-wise, the GloVe library provides the top k similar words of w_r without considering the word w_r itself, having similarity score 1. Hence, using the GloVe library, w_r must also be added to the boosted BOW.

D.3.3 Similar Words within Cosine Angle Threshold

Another approach to enrich the Boolean BOW of a sentence is thresholding the cosine angle. This is different from the first technique because the number of additional words extracted will vary rather than being fixed. Whereas the first approach always produces $k - 1$ new features for each given word, the cosine angle thresholding brings in all those words that are sufficiently similar. The cosine similarity threshold is given by $\phi = \cos(\theta)$, where θ is the threshold for vector angle, while ϕ is the corresponding similarity score.

As per Eq. (A.2), we obtain Φ_r, which consists of the similarity scores of the given word w_r in comparison to the m words in the vocabulary. Then, for each given word w_r, the indices of those scores ϕ_r^t that are greater than or equal to ϕ ($\phi_r^t \geq \phi$) are stored in the set A_r. Similar to the first technique, the words in W with the indices in A_r are utilized to create X_{mod} as given by Eq. (A.3).

D.4 Tsetlin Machine-based Classification

D.4.1 Tsetlin Machine Architecture

A TM is composed by TAs that operate with literals – Boolean inputs and their negations – to form conjunctions of literals (conjunctive clauses). A dedicated team of TAs builds each clause, with each input being associated with a pair of TAs. One TA controls the original Boolean input whereas the other TA controls its negation. The TA pair selects a combination of "Include" or "Exclude" actions, which decide the form of the literal to include or exclude in the clause.

Each TA decides upon an action according to its current state. There are N states per TA action, $2N$ states in total. When a TA finds itself in states 1 to N, it performs the "Exclude"

excellent film, enjoyable					
excellent	similarity	film	similarity	enjoyable	similarity
good	0.703	movie	0.858	entertaining	0.688
superb	0.680	films	0.839	pleasurable	0.660
terrific	0.656	movies	0.716	exhilarating	0.647
quality	0.603	directed	0.683	fun	0.625
wonderful	0.584	documentary	0.656	exciting	0.619
best	0.550	starring	0.651	amusing	0.605
exceptional	0.550	cinema	0.637	satisfying	0.585
perfect	0.535	screenplay	0.632	engrossing	0.578
impressive	0.534	drama	0.622	enlightening	0.573
decent	0.532	comedy	0.615	informative	0.571

top k similar words

words within ϕ

Figure D.1: Similar words for an example "excellent film, enjoyable" using 300d GloVe word representation.

very good movie					
very	similarity	good	similarity	movie	similarity
extremely	0.872	better	0.765	film	0.858
quite	0.858	really	0.736	movies	0.849
so	0.785	always	0.717	films	0.790
pretty	0.738	you	0.707	hollywood	0.679
too	0.731	well	0.704	starring	0.675
really	0.729	excellent	0.703	comedy	0.658
well	0.720	very	0.696	sequel	0.646
always	0.712	things	0.693	remake	0.624
especially	0.709	think	0.689	drama	0.608
but	0.707	way	0.683	actor	0.599

top k similar words

words within ϕ

Figure D.2: Similar words for an example "very good movie" using 300d GloVe word representation.

action. When in states $N+1$ to $2N$, it performs the "Include" action. How the TA updates its state is shown in Fig. D.3. If it receives Reward, the TA moves to a deeper state thereby increasing its confidence in the current action. However, if it receives Penalty, it moves towards the centre, weakening the action. It may eventually jump over the middle decision boundary, to the other action. It is through this game of TAs that the TM shapes the clauses into frequent and discriminative patterns.

With respect to NLP, TM heavily relies on the Boolean BOW introduced earlier in the paper. We now make use of our proposed modified BOW $X_{mod} = [x_1, x_2, x_3, \ldots, x_m]$. Let l be the number of clauses that represent each class of the TM, covering q classes altogether. Then, the overall pattern recognition problem is solved using $l \times q$ clauses. Each clause C_i^j, $1 \leq j \leq q$, $1 \leq i \leq l$ of the TM is given by $C_i^j = \left(\bigwedge_{k \in I_i^j} x_k \right) \wedge \left(\bigwedge_{k \in \bar{I}_i^j} \neg x_k \right)$, where I_i^j and $\bar{I}_i^j$ are non-overlapping subsets of the input variable indices, $I_j^i, \bar{I}_j^i \subseteq \{1, \ldots, m\}, I_j^i \cap \bar{I}_j^i = \emptyset$. The subsets decide which of the input variables take part in the clause, and whether they are negated

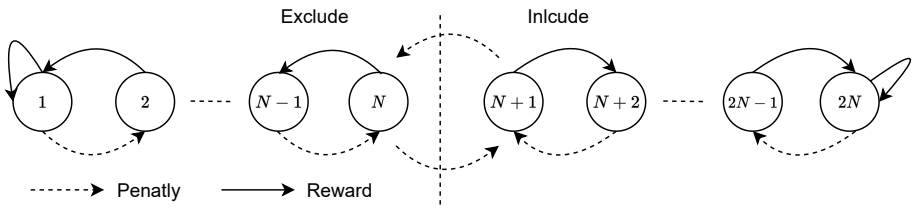


Figure D.3: A TA with two actions: “Include” and “Exclude”.

or not. The indices of input variables in I_j^i represent the literals that are included as is, while the indices of input variables in $\bar{I}_j^i$ correspond to the negated ones. Among the q clauses of each class, clauses with odd indexes are assigned positive polarity (+) whereas those with even indices are assigned negative polarity (-). The clauses with positive polarity vote for the target class and those with negative polarity vote against it. A summation operator aggregates the votes by subtracting the total number of negative votes from positive votes, as shown in Eq. (A.4).

$$f^j(X_{mod}) = \Sigma_{i=1,3,...}^{l-1} C_i^j(X_{mod}) - \Sigma_{i=2,4,...}^{l} C_i^j(X_{mod}). \tag{A.4}$$

For q number of classes, the final output y is given by the argmax operator to classify the input based on the highest sum of votes, $\hat{y} = \text{argmax}_j\left(f^j(X_{mod})\right)$.

D.4.2 Distributed Word Representation in TM

Consider two contexts for sentiment classification: “Very good movie” and “Excellent film, enjoyable”. Both contexts have different vocabularies but some of them are semantically related to each other. For example, “good” and “excellent” have similar semantics as well as “film” and “movie”. Such semantics are not captured in the BOW-based input. However, as shown in Fig. D.4, adding words to the BOWs that are semantically related, as proposed in the previous section, makes distributed word representation available to the TM.

The resulting BOW-boosted TM architecture is shown in Fig. D.5. Here each input feature is first expanded using the GloVe representation, adding semantically related words. Each feature is then transferred to its corresponding TAs, both in original and negated form. Each TA, in turn, decides whether to include or exclude its literal in the clause by taking part in a decentralized game. The actions of each TA is decided by its current state and updated by the the feedback it receives based on its action. As shown in the figure, the TA actions produce a collection of conjunctive clauses, joining the words into more complex linguistic patterns.

There are two types of feedback that guides the TA learning. They are Type I feedback and Type II feedback, detailed in [1]. Type I feedback is triggered when the ground truth label is 1, i.e., $y = 1$. The purpose of Type I feedback is to include more literals from the BOW to refine the clauses, or to trim them by removing literals. The balance between refinement and trimming is controlled by a parameter called specificity, s. Type I feedback guides the clauses to provide true positive output, while simultaneously controlling over-fitting by producing frequent patterns. Conversely, Type II feedback is triggered in case of false positive output. Its main aim is to introduce zero-valued literals into clauses when they give false positive output. The purpose is to change them so that they correctly output zero later in the learning process. Based on these

feedback types, each TA in a clause receives Reward, Penalty or Inaction. The overall learning process is explained in detail by Yadav et al. in [5].

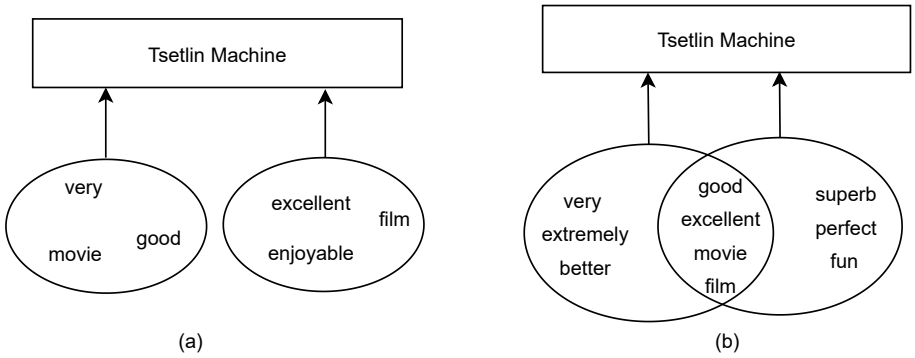


Figure D.4: (a) BOW input representation without distributed word representation. (b) BOW input using similar words based on distributed word representation.

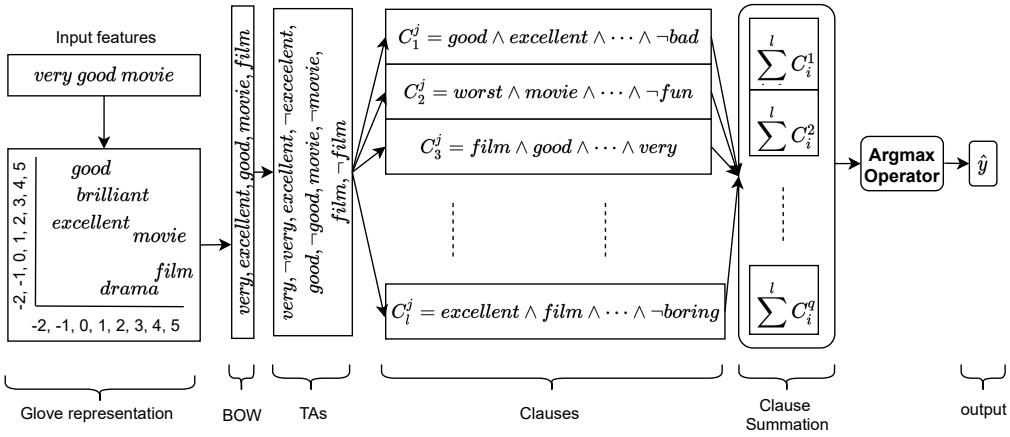


Figure D.5: Architecture of TM using modified BOW based on word similarity.

D.5 Experiments and Results

In this section, we evaluate our TM-based solution with the input features enhanced by distributed word representation. Here we use Glove pretrained word vector that is trained using CommonCrawl with the configuration of 42B tokens, 1.9M vocab, uncased, and 300d vectors.

D.5.1 Datasets

We have selected various types of datasets to investigate how broadly our method is applicable: R8 and R52 of Reuters, Movie Review (MR), and TREC-6. • **Reuters** 21578 dataset include two subsets: R52 and R8 (all-terms version). R8 is divided into 8 sections while there are 52 categories in R52. •**MR** is a movie analysis dataset for binary sentiment classification with just one sentence per review [29]. In this study, we used a training/test split from [24][1]. •**TREC-6**

[1] https://github.com/mnqu/PTE/tree/master/data/mr.

Parameters	R8	R52	MR	TREC
k=0	96.16	84.62	75.14	88.05
k=3	97.08	88.59	75.21	88.72
k=5	96.80	70.60	76.06	89.16
k=10	87.44	66.94	77.51	89.82

Table D.1: Comparison of feature extended TM with several parameters for k.

is a question classification dataset [30]. The task entails categorizing a query into six distinct categories (abbreviation, description, entity, human, location, numeric value).

D.5.2 TM Parameters

A TM has three parameters that must be initialized before training a model: number of clauses l, voting target T, and specificity s. We configure these parameters as follows. For R8, we use 2,500 clauses, a threshold of 80, and specificity 9. The vocabulary size is 5,000. For R52, we employ 1,500 clauses, the voting target is 80, and specificity is 9. Here, we use a vocabulary of size 6,000. For MR, the number of clauses is 3,000, the voting target is 80, and specificity is 9, with a vocabulary of size 5,000. Finally, for TREC, we use 2,000 clause, a voting target of 80, and specificity 9, with vocabulary size 6,000. These parameters are kept static as we explore various k and θ values for selecting similar words to facilitate comparison. The code and datasets are available online [2].

D.5.3 Performance When Using Top *k* Nearest Neighbors

Here, we demonstrate the performance on each of the datasets, exploring the effect of different k-values, i.e., 3, 5 and 10. The performance of the proposed technique for selected datasets with various values of k is compared in Table D.1. It can be seen that by using feature extension, performance is significantly enhanced. Both $k = 3$ and $k = 5$ outperform the simple BOW ($k = 0$). However, for this particular dataset, $k = 10$ performs poorly because extending each word to its 10 nearest neighbors includes many unnecessary contexts that have no significant impact on the classification. In terms of accuracy, $k = 5$ performs best for the R8 dataset. For the R52 dataset, the feature extension with $k = 5$ and $k = 10$ performs poorly compared to using $k = 0$ and $k = 3$. Here, $k = 3$ is the best-performing parameter. The improvement obtained by moving from a simple BOW to a BOW enhanced with semantically similar features is obvious in the case of the R52 dataset. Similarly, in the case of the TREC dataset, the performance of simple BOW ($k = 0$) is markedly outperformed by the feature extension techniques for all the tested k-values, with $k = 5$ and $k = 10$ being good candidates. The advantage of $k = 10$ over $k = 5$ is that $k = 10$ reaches its peak accuracy in an earlier epoch. Lastly, the performance of the MR is again clear that the feature extension technique outperforms the simple BOW ($k = 0$) with a high margin.

[2] `https://github.com/rohanky/Glove-TM`

Parameters	R8	R52	MR	TREC
$\phi = 0$	96.16	84.62	75.14	88.05
$\phi = 0.5$	88.08	89.14	73.24	90.04
$\phi = 0.6$	90.86	88.05	74.34	87.83
$\phi = 0.7$	96.53	88.51	76.55	89.38
$\phi = 0.8$	96.25	88.94	75.12	88.27
$\phi = 0.9$	96.39	87.50	74.59	87.39

Table D.2: Comparison of feature extended TM with several parameters for ϕ.

D.5.4 Performance When Using Neighbors Within a Similarity Threshold

This section demonstrates the performance of our BOW enhancement approach when using various similarity thresholds ϕ for feature extension. Here, ϕ refers to the cosine similarity between a word in the BOW and a target word from the overall vocabulary. Again, similarity is measured in the GloVe embedding space as the cosine of the angle θ between the embedding vectors compared, $\cos(\theta)$. For ϕ, we here explore the values 0.5, 0.6, 0.7, 0.8, and 0.9, whose corresponding angles are $60°$, $53.13°$, $45.57°$, $36.86°$, and $25.84°$, respectively. The performance of the various ϕ-values for the selected dataset is shown in Table D.2. For R8 dataset, feature extension using $\phi = 0.7$, $\phi = 0.8$, and $\phi = 0.9$ outperforms the simple BOW ($\phi = 0$) where $\phi = 0.7$ being the best. In case of the R52 dataset, all of the investigated ϕ-values outperform the simple BOW ($\phi = 0$) where $\phi = 0.5$ and $\phi = 0.8$ performs the best. Similar trend is observed in case of TREC and MR dataset where feature extension outperforms the simple BOW.

In most of the cases, however, a too strict similarity threshold ϕ tends to reduce performance because fewer features are added to the BOW. Even though using a looser similarity score thresholds also introduces unnecessary features, these do not seem to impact the formation of accurate clauses. Overall, our experiments show that using ϕ-values from 0.5 to 0.7 peaks performance.

D.5.5 Comparison with Baselines

We here compare our proposed model with selected text classification- and embedding methods. We have selected representative techniques from various main approaches, both those that leverage similar kinds of pre-trained word embedding and those that only use BOW. The selected baselines are: •**TF-IDF+LR**: This is a bag-of-words model employing Term Frequency-Inverse Document Frequency (TF-IDF) weighting. Logistic Regression is used as a softmax classifier. •**CNN**: The CNN-baselines cover both initialization with random word embedding (CNN-rand) as well as initialization with pretrained word embedding (CNN-non-static) [31]. • **LSTM**: The LSTM model that we employ here is from [32], representing the entire text using the last hidden state. We tested this model with and without pre-trained word embeddings. • **Bi-LSTM**: Bi-directional LSTMs are widely used for text classification. We compare our model with Bi-LSTM fed with pre-trained word embeddings. •**PV-DBOW**: PV-DBOW is a paragraph vector model where the word order is ignored. Logistic Regression is used as a softmax classifier [23]. • **PV-DM**: PV-DM is also a paragraph vector model, however with word ordering taken into account. Logistic Regression is used as a softmax classifier [23]. •**fastText**: This baseline is

a simple text classification technique that uses the average of the word embeddings provided by fastText as document embedding. The embedding is then fed to a linear classifier [21]. We evaluate both the use of uni-grams and bigrams. • **SWEM** : SWEM applies simple pooling techniques over the word embeddings to obtain a document embedding [33]. •**Graph-CNN-C**: A graph CNN model uses convolutions over a word embedding similarity graph [34], employing a Chebyshev filter. •S^2GC: This technique uses a modified Markov Diffusion Kernel to derive a variant of Graph Convolutional Network (GCN) [35]. •**LguidedLearn**: It is a label-guided learning framework for text classification. This technique is applied to BERT as well [36], which we use for comparison purposes here. •**Feature Projection (FP)**: It is a novel approach to improve representation learning through feature projection. Existing features are projected into an orthogonal space [37].

From Table D.3, we observe that the TM approaches that employ either of our feature extension techniques outperform several word embedding-based Logistic Regression approaches, such as PV-DBOW, PV-DM, and fastText. Similarly, the legacy TM outperforms sophisticated models like CNN and LSTM based on randomly initialized word embedding. Still, the legacy TM falls of other models when they are initialized by pre-trained word embeddings. By boosting the Boolean BOW with semantically similar features using our proposed technique, however, TM outperforms LSTM (pretrain) on the R8 dataset and performs similarly on R52 and MR. In addition to this, our proposed approach achieves quite similar performance compared to BERT, even though BERT has been pre-trained on a huge text corpus. However, it falls slightly short of sophisticated fine-tuned models like Lguided-BERT-1 and Lguided-BERT-3. Overall, our results show that our proposed feature extension technique for TMs significantly enhances accuracy, reaching state of the art accuracy. Importantly, this accuracy enhancement does not come at the cost of reduced interpretability, unlike DNNs, which we discuss below. The state of the art for the TREC dataset is different from the other three datasets, hence we report results separately in Table D.4. These results clearly show that although the basic TM model does not outperform the recent DNN- and transformer-based models, the feature-boosted TM outperforms all of those models except understandably BAE:BERT [38].

D.5.6 Interpretation

The proposed feature extension-based TM does not only impact accuracy. Perhaps surprisingly, our proposed technique also simplify the clauses that the TM produces, making them more meaningful in a semantic sense. To demonstrate this property, let us consider two samples from the MR dataset: S_1="the cast is uniformly excellent and relaxed" and S_2="the entire cast is extraordinarily good". Let the vocabulary, in this case, be [cast, excellent, relaxed, extraordinarily, good, bad, boring, worst] as shown in Fig. D.6. .

As we can see, that the TM initialized with normal BOW uses two separate clauses to represent two examples. However, augmenting feature on TM uses only one clause that learns the semantic for multiple examples.This indeed makes interpretation of TM more powerful and meaningful as compared to simple BOW based TM.

Model	R8	R52	MR
TF-IDF+LR	93.74	86.95	74.59
CNN-rand	94.02	85.37	74.98
CNN-non-static	95.71	87.59	77.75
LSTM	93.68	85.54	75.06
LSTM (pretrain)	96.09	90.48	77.33
Bi-LSTM	96.31	90.54	77.68
PV-DBOW	85.87	78.29	61.09
PV-DM	52.07	44.92	59.47
fastText	96.13	92.81	75.14
fastText (bigrams)	94.74	90.99	76.24
SWEM	95.32	92.94	76.65
LEAM	93.31	91.84	76.95
Graph-CNN-C	96.99	92.74	77.22
S^2GC	97.40	94.50	76.70
BERT	96.02	89.66	79.24
Lguided-BERT-1	97.49	94.26	81.03
Lguided-BERT-3	98.28	94.32	81.06
TM	96.16± 1.52	84.62± 1.8	75.14± 1.2
TM with k	97.50± 1.12	88.59± 1.2	77.51± 0.6
TM with ϕ	96.39± 1.0	89.14± 1.5	76.55± 0.9

Table D.3: Comparison of feature extended TM with the state of the art for R8, R52 and MR. Reported accuracy of TM is the mean of last 50 epochs of 5 independent experiments with their standard deviation.

D.6 Conclusions

In this paper, we aimed to enhance the performance of Tsetlin Machines (TMs) by introducing a novel way to exploit distributed feature representation for TMs. Given that a TM relies on Bag-of-words (BOW), it is not possible to introduce pre-trained word representation into a TM directly, without sacrificing the interpretability of the model. To address this intertwined challenge, we extended each word feature by using cosine similarity on the distributed word representation. We proposed two techniques for feature extension: (1) using the k nearest words in embedding space and (2) including words within a given cosine angle (θ). Through this enhancement, the TM BOW can be boosted with pre-trained world knowledge in a simple yet effective way. Our experiment results showed that the enhanced TM not only achieve competitive accuracy compared to state of the art, but also outperform some of the sophisticated deep neural network (DNN) models. In addition, our BOW boosting also improved the interpretability of the model by increasing the scope of each clause, semantically relating more samples. We thus believe that our proposed approach significantly enhance the TM in the accuracy/interpretability continuum, establishing a new standard in the field of explainable NLP.

Model	TREC
LSTM	87.19
FP+LSTM	88.83
Transformer	87.33
FP+Transformer	89.51
BAE: BERT	97.6
TM [39]	87.20
TM	88.05± 1.52
TM with k	89.82± 1.18
TM with ϕ	90.04± 0.94

Table D.4: Comparison of feature extended TM with the state of the art for TREC. Reported accuracy of TM is the mean of last 50 epochs of 5 independent experiments with their standard deviation.

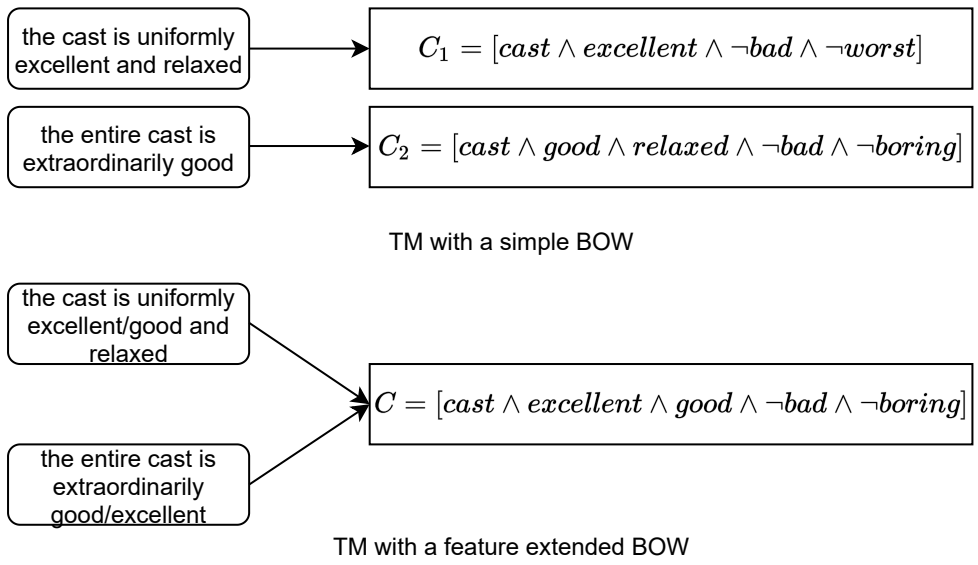


Figure D.6: Clause learning semantic for multiple examples compared to simple BOW based TM.

Bibliography

[1] O.-C. Granmo, "The tsetlin machine - a game theoretic bandit driven approach to optimal pattern recognition with propositional logic," *ArXiv*, vol. abs/1804.01508, 2018.

[2] G. T. Berge, O.-C. Granmo, T. O. Tveit, M. Goodwin, L. Jiao, and B. V. Matheussen, "Using the tsetlin machine to learn human-interpretable rules for high-accuracy text categorization with medical applications," *IEEE Access*, vol. 7, pp. 115134–115146, 2019.

[3] K. D. Abeyrathna, O.-C. Granmo, X. Zhang, and M. Goodwin, "A scheme for continuous input to the Tsetlin machine with applications to forecasting disease outbreaks," in *Advances and Trends in Artificial Intelligence. From Theory to Practice*, pp. 564–578, Springer International Publishing, 2019.

[4] O.-C. Granmo, S. Glimsdal, L. Jiao, M. Goodwin, C. W. Omlin, and G. T. Berge, "The convolutional tsetlin machine," *arXiv*, vol. 1905.09688, 2019.

[5] R. K. Yadav, L. Jiao, O.-C. Granmo, and M. Goodwin, "Human-level interpretable learning for aspect-based sentiment analysis," in *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*, AAAI, 2021.

[6] R. Saha, O.-C. Granmo, and M. Goodwin, "Mining interpretable rules for sentiment and semantic relation analysis using tsetlin machines," in *Artificial Intelligence XXXVII*, (Cham), pp. 67–78, Springer International Publishing, 2020.

[7] R. K. Yadav., L. Jiao., O. Granmo., and M. Goodwin., "Interpretability in word sense disambiguation using tsetlin machine," in *Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*, pp. 402–409, INSTICC, SciTePress, 2021.

[8] B. Bhattarai., O. Granmo., and L. Jiao., "Measuring the novelty of natural language text using the conjunctive clauses of a tsetlin machine text classifier," in *Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*, pp. 410–417, INSTICC, SciTePress, 2021.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS, Nevada, USA*, vol. 26, pp. 3111–3119, Curran Associates, Inc., 2013.

[10] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP, Doha, Qatar*, p. 1532–1543, 2014.

[11] Z. S. Harris, "Distributional structure," *WORD*, vol. 10, no. 2-3, pp. 146–162, 1954.

[12] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[13] J. Turian, L.-A. Ratinov, and Y. Bengio, "Word representations: A simple and general method for semi-supervised learning," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, (Uppsala, Sweden), pp. 384–394, Association for Computational Linguistics, 2010.

[14] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng, "Parsing with compositional vector grammars," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Sofia, Bulgaria), pp. 455–465, Association for Computational Linguistics, 2013.

[15] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[16] S. Wang and C. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *ACL (Volume 2: Short Papers)*, (Jeju Island, Korea), pp. 90–94, 2012.

[17] V. Chenthamarakshan, P. Melville, V. Sindhwani, and R. D. Lawrence, "Concept labeling: Building text classifiers with minimal supervision," in *IJCAI*, pp. 1225–1230, 2011.

[18] Y. Tang, K. Blincoe, and A. Kempa-Liehr, "Enriching feature engineering for short text samples by language time series analysis," *EPJ Data Science*, vol. 9, pp. 1–59, 2020.

[19] F. Rousseau, E. Kiagias, and M. Vazirgiannis, "Text categorization as a graph classification problem," in *ACL (Volume 1: Long Papers)*, (Beijing, China), pp. 1702–1712, ACL, 2015.

[20] Y. Luo, Ö. Uzuner, and P. Szolovits, "Bridging semantics and syntax with graph algorithms - state-of-the-art of extracting biomedical relations," *Briefings in bioinformatics*, vol. 18 1, pp. 160–178, 2017.

[21] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *EACL: Volume 2, Short Papers*, (Valencia, Spain), pp. 427–431, ACL, 2017.

[22] D. Shen, G. Wang, W. Wang, M. R. Min, Q. Su, Y. Zhang, C. Li, R. Henao, and L. Carin, "Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms," in *ACL Volume 1: Long Papers*, (Melbourne, Australia), pp. 440–450, ACL, 2018.

[23] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning*, vol. 32 of *Proceedings of Machine Learning Research*, (Bejing, China), pp. 1188–1196, PMLR, 22–24 Jun 2014.

[24] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, (Sydney, NSW, Australia), p. 1165–1174, Association for Computing Machinery, 2015.

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *ACL: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, ACL, 2019.

[26] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[27] B. Bai, J. Liang, G. Zhang, H. Li, K. Bai, and F. Wang, "Why is attention not so interpretable," *arXiv: Machine Learning*, 2020.

[28] S. Serrano and N. A. Smith, "Is attention interpretable?," in *ACL*, (Florence, Italy), pp. 2931–2951, ACL, 2019.

[29] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *ACL*, (Michigan, USA), p. 115–124, ACL, 2005.

[30] X. Li and D. Roth, "Learning question classifiers," in *COLING*, 2002.

[31] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1746–1751, ACL, 2014.

[32] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," in *IJCAI*, p. 2873–2879, 2016.

[33] D. Shen, G. Wang, W. Wang, M. R. Min, Q. Su, Y. Zhang, C. Li, R. Henao, and L. Carin, "Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms," in *ACL (Volume 1: Long Papers)*, (Melbourne, Australia), pp. 440–450, ACL, 2018.

[34] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[35] H. Zhu and P. Koniusz, "Simple spectral graph convolution," in *International Conference on Learning Representations*, 2021.

[36] X. Liu, S. Wang, X. Zhang, X. You, J. Wu, and D. Dou, "Label-guided learning for text classification," *ArXiv*, vol. abs/2002.10772, 2020.

[37] Q. Qin, W. Hu, and B. Liu, "Feature projection for improved text classification," in *ACL*, (Online), pp. 8161–8171, ACL, 2020.

[38] S. Garg and G. Ramakrishnan, "Bae: Bert-based adversarial examples for text classification," 2020.

[39] Dragoș, C. Nicolae, and dragosnicolae, "Question classification using interpretable tsetlin machine," in *International Workshop of Machine Reasoning*, ACM International Conference on Web Search and Data Mining, 2021.

Appendix E

Paper E

Title:	Robust Interpretable Text Classification against Spurious Correlations Using AND-rules with Negation
Authors:	Rohan Kumar Yadav, Lei Jiao, Ole-Christoffer Granmo, and Morten Goodwin
Affiliation:	University of Agder, Faculty of Engineering and Science, 4879, Grimstad, Norway
Conference:	*International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.
DOI:	.

Appendix F

Paper F

Title:	Enhancing Attention's Explanation Using Interpretable Tsetlin Machine
Authors:	Rohan Kumar Yadav [1,], Dragoş Constantin Nicolae [2,]
Affiliation:	[1,] University of Agder, Faculty of Engineering and Science, 4879, Grimstad, Norway [2,] Research Institute for Artificial Intelligence "Mihai Drăgănescu" 050711 Bucharest, Romania
Journal:	*Algorithms, MDPI*, 2022.
DOI:	.

Enhancing Attention's Explanation Using Interpretable Tsetlin Machine

Rohan Kumar Yadav [1], Dragoş Constantin Nicolae [2]
[1] Department of Information and Communication Technology
Faculty of Engineering and Science, University of Agder
4879, Grimstad, Norway
[2] Research Institute for Artificial Intelligence "Mihai Drăgănescu"
050711 Bucharest, Romania
E-mails: rohan.k.yadav@uia.no, dragosnicolae555@gmail.com

Abstract — **Explainability is one of the key factors in Natural Language Processing (NLP) specially for legal documents, medical diagnosis, and clinical text. Attention mechanism has been a popular choice for such explainability recently by estimating the relative importance of input units. Recent research has revealed, however, that such processes tend to misidentify irrelevant input units when explaining them. This is due to the fact that language representation layers are initialized by pre-trained word embedding that is not context-dependent. Such a lack of context-dependent knowledge in the initial layer makes it difficult for the model to concentrate on the important aspects of input. Usually, this does not impact the performance of the model, but the explainability differs from human understanding. Hence, in this paper, we propose an ensemble method to use logic-based information from the Tsetlin Machine to embed it into the initial representation layer in the neural network to enhance the model in terms of explainability. We obtain the global clause score for each word in the vocabulary and feed it into the neural network layer as context-dependent information. Our experiments show that the ensemble method enhances the explainability of the attention layer without sacrificing any performance of the model and even outperforming in some datasets.**

F.1 Introduction

In natural language processing, text categorization is a crucial task (NLP) [1, 2] and neural network models are the ones to dominate state-of-the-art approaches. However, these models are often assumed to be blackbox in nature. The models' opacity has become a serious impediment to their creation, implementation, and improvement, especially in crucial tasks like medical diagnosis [3] and legal document inspection [4]. As a result, explainable text classification has become a major topic, with the objective of providing end-users with human-readable descriptions of the classification logic [5, 6, 7, 1].

The attention mechanism is a prominent technique among current explainability approaches that identify essential sections of the input for the prediction job by offering a distribution across attended-to-input units [8]. Many NLP tasks, such as text categorization, question answering, and entity identification, have shown outstanding results using attention-based models [8, 9, 2]. In particular, in many NLP systems, the self-attention mechanism that underpins the Transformer design has played a key role [10, 11]. Despite this, recent research has revealed that learned attention weights are frequently unrelated to the relevance of input components as judged by various explainability approaches [12], and that alternative attention distributions can provide identical predictions [13, 14].

Various alternative approaches could replace attention-based neural networks for explainable NLP such as decision tree and logistic regression. However, they suffer from low performance compared to neural networks. In addition to this logistic regression does not provide a logical explanation but provides mathematical weights for selected inputs [15]. On the other hand, decision trees are only suited for a limited dataset size. It becomes extremely difficult to get the explainability once the trees get more complex. Due to these limitations, there has been a limited study in obtaining a logical explanation for NLP classification. A recent study has found that Tsetlin Machine (TM) has been a promising tool for rule-based explanation in image, text, and numerical data [16, 17, 18, 19]. TM is an interpretable rule-based model that uses conjunctive clauses to learn basic and complicated correlations. Unlike Deep Neural Netowrks (DNNs) and basic rule-based systems, TM learns rules in the same way that humans do, using logical reasoning, and it does so in a visible and interpretable manner [16, 20]. TM has shown that it obtains a good trade-off between accuracy and interpretability on many NLP tasks [21, 22]. However, there lie some limitations such as boolean bag-of-words input and incapable of using pre-trained information.

One efficient way to deal with the above-mentioned problem is to use prerequisite knowledge to enhance the input layer for better interpretation. Integrating human rationales as supplementary supervision information for attention learning is a promising way to enhance the explainability of attention-based models. Human rationales have previously been found to be useful input for increasing model performance and discovering explainable input in model prediction [23, 24]. However, obtaining such human rationales is an expensive and time-consuming process. Hence, to make it more easy and efficient, we use a logic-based model TM that mimics human-level understanding to generate prerequisite information to initialize the input layer of the neural network. Since TM can be explained by logic and rules, the information it provides can be easily explained to make the attention layer focus on important input tokens.

In this paper, we train TM on two movie review datasets and leverage the clause score of TM

for each word in the vocabulary. We then use this prerequisite information of each word as initial information for the input layer in the neural network. We use Bidirection Gated Recurrent Unit (Bi-GRU) [25] for language representation for neural networks and Glove [26] to initialize the word embedding. In addition to this, we multiply the input embedding layer with prerequisite information of each word from TM. This makes the attention layer on top of Bi-GRU focus on important words.

F.2 Related Work

Machine learning explainability has lately received a lot of attention, owing to the necessity for transparency [5, 27]. Existing explainability approaches may be divided into two types: post-hoc and intrinsic explainability. The goal of post-hoc explainability is to provide explanations for a model that already exists. In the feature space, a representative technique approximates decisions of the model with an explainable technique (e.g., a linear model) [5]. Generative Explanation Framework (GEF) [28] is a recent development in this field that aims to explain a generic encoder-predictor architecture by concurrently generating explanations and classification results. The goal of intrinsic explainability is to create self-explanatory models. This can be accomplished by enforcing feature sparsity [29], representation disentanglement [30], or sensitivity to input characteristics through explainability requirements in model learning. Attention mechanisms, that identify sections of the input that are considered by the model for specific output predictions, are a more prevalent technique to explain individual predictions [31, 8]. These attention processes have long been essential in NLP, not just because of their explainability, but also because of the improvements they provide to model performance [10, 11]. An empirical study recently questioned their effectiveness in explaining model performance, pointing out that attention distributions are contrary with the importance of input features measured by gradient-based methods, and those adversarial distributions can be found yielding similar model performance [13]. These discoveries have sparked heated debates, such as how attention mechanisms provide larger weights to key input features for a specific task even when the model for prediction changes [14].

The concept of adding human rationales for improvement of the model may be traced back to a situation in which a human teacher highlights sections of text in a document as a justification for label annotation [23]. By restricting the prediction labels, the logic is integrated into the loss function of SVM classifier. Similar concepts have been investigated for neural network models [24] and various methods of human reason integration, such as learning a mapping between human rationales and machine attention [32] or assuring variety among hidden representations learned at different time steps [33]. Even though recent studies in human computation [34] have shown that asking workers to provide human annotation rationales—by headlining the supporting text excerpts from the given context—requires no additional annotation effort, reassigning human rationales in the previous datasets requires additional time and cost. Hence, there is the need for a human explainable model that can substitute the human-in-loop system as prerequisite knowledge for the neural network model.

In this paper, we propose an alternative for human rationales by using Tsetlin Machine and its explainability. TM consists of several clauses in the form of propositional logic. Each feature in TM represents the collection of the clauses for a particular classification model. Such clause

score also represents the weightage of each feature in the model. Since TM is easily explainable, it makes sense to use this explanation as ensemble information for the neural network.

F.3 Proposed Architecture: TM Initialized Attention Model

Here, we discuss the architecture of the model that ensemble the information from TM into neural network. First we explain the architecture of TM and then process to obtain the clause score.

F.3.1 Clause Score from Tsetlin Machine Architecture

A revolutionary game-theoretic strategy that organizes a collection of decentralized team of Tsetlin Automata is at the heart of the TM (TAs). Based on disjunctive normal form, the strategy directs the TAs to learn arbitrarily complicated propositional formula in the form of conjunctive form [35, 36]. A TM is interpretable in a way that it decomposes issues into self-contained sub-patterns that may be interpreted separately, notwithstanding its ability to learn complicated nonlinear patterns. Each sub-pattern is represented by a conjunctive sentence, which is a series of literals, each of which represents an input bit or its negation. As a result, sub-pattern representation and evaluation are both Boolean. In comparison to other approaches, this makes the TM computationally efficient and hardware friendly [37, 38].

TM is a new classification approach based on a team of Tsetlin Automata that manipulates phrases in propositional logic. TA is a deterministic automaton with a fixed structure that learns the best action from a collection of actions provided by the environment. A two-action TA with $2N$ states is shown in Figure F.1. The states from 1 to N are referred to as Action 1, whereas the states from $(N+1)$ to $2N$ are referred to as Action 2. TA conducts the action depending on the current state and interacts with the environment during each iteration. This, in turn, causes the environment to issue a random reward or penalty based on an unknown probability distribution. If TA is rewarded, it advances deeper into the state; if it is penalized, it moves closer to the center of the state, weakening the preformed action, and finally jumping to the side of the other action. Each input bit in TM is represented by two TAs, *TA*and *TA'*. The original bit of the input sample is controlled by *TA*, while the negation is controlled by *TA'*. As a result, the TM, which is made up of clauses, will eventually converge to the desired pattern. There are two sorts of feedback (reward or penalty) supplied to the TM: Type I and Type II feedback. The TA for the training samples is given rewards or penalties based on these feedback types. The both feedbacks are shown in Tables F.1 and F.2 respectively.

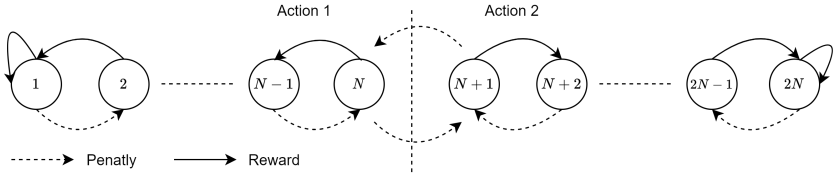


Figure F.1: The two-action TA and its transition in TM.

In regards to NLP, TM heavily relies on the Boolean Bag-of-words (BOW) given by $X =$

Input	Clause	1		0	
	Literal	1	0	1	0
Include Literal	P(Reward)	$\frac{s-1}{s}$	NA	0	0
	P(Inaction)	$\frac{1}{s}$	NA	$\frac{s-1}{s}$	$\frac{s-1}{s}$
	P(Penalty)	0	NA	$\frac{1}{s}$	$\frac{1}{s}$
Exclude Literal	P(Reward)	0	$\frac{1}{s}$	$\frac{1}{s}$	$\frac{1}{s}$
	P(Inaction)	$\frac{1}{s}$	$\frac{s-1}{s}$	$\frac{s-1}{s}$	$\frac{s-1}{s}$
	P(Penalty)	$\frac{s-1}{s}$	0	0	0

Table F.1: The Type I Feedback.

Input	Clause	1		0	
	Literal	1	0	1	0
Include Literal	P(Reward)	0	NA	0	0
	P(Inaction)	1.0	NA	1.0	1.0
	P(Penalty)	0	NA	0	0
Exclude Literal	P(Reward)	0	0	0	0
	P(Inaction)	1.0	0	1.0	1.0
	P(Penalty)	0	1.0	0	0

Table F.2: The Type II Feedback.

$[x_1, x_2, x_3, \ldots, x_n]$. Let l be the number of clauses that represent each class of the TM, covering q classes altogether. Then, the overall learning problem is solved using $l \times q$ clauses. Each clause C_i^j, $1 \leq j \leq q$, $1 \leq i \leq l$ of the TM is given by :

$$C_i^j = \left(\bigwedge_{k \in I_i^j} x_k \right) \wedge \left(\bigwedge_{k \in \bar{I}_i^j} \neg x_k \right), \tag{F.1}$$

where I_i^j and $\bar{I}_i^j$ are non-overlapping subgroup of the input variable indices, $I_j^i, \bar{I}_j^i \subseteq \{1, \ldots, m\}, I_j^i \cap \bar{I}_j^i = \emptyset$. The subgroup decide that which of the input variables to participate in the clause, and whether they are in the original form or the negated. The indices of input features in I_j^i represent the literals that are included as original form of the literals, while the indices of input features in $\bar{I}_j^i$ correspond to the negated ones. Among the q clauses of each class, clauses that are indexed with odd number are assigned positive polarity (+) whereas those with even indexed are assigned negative polarity ($-$). The clauses with positive polarity vote for the true target class and those with negative polarity vote against it. A summation operator aggregates the votes by subtracting the total number of negative votes from positive votes, as shown in Equation (F.2).

$$f^j(X) = \Sigma_{i=1,3,\ldots}^{l-1} C_i^j(X) - \Sigma_{i=2,4,\ldots}^{l} C_i^j(X). \tag{F.2}$$

For q number of classes, the predicted output y is given by the argmax operator which classifies the input features based on the highest sum of votes obtained, as shown in Equation (F.3).

$$\hat{y} = \text{argmax}_j \left(f^j(X) \right). \tag{F.3}$$

Once the model is trained with a particular dataset, we can explore the clauses that holds information of combination of literals in propositional form. Such information is humanly interpretable and can be used for downstream applications of NLP. Here, we explore the weightage of each word in the model. We pass each word in the vocabulary into the TM and obtain the clause score. The clause score is calculated by:

$$SC_{x_k} = |f^{\kappa=tp}(X_{x_k=1}) - \Sigma f^{\kappa=fp}(X_{x_k=1})|. \tag{F.4}$$

Here *tp* refers to true prediction, *fp* refers to false prediction, $|.|$ refers to the absolute value, and $k = 1, 2, \ldots, n$, where n is the number of vocabularies. We then create the input map for each input sentence with the score obtained for each word which will then fed to neural network initial embedding layer.

F.3.2 Attention Based Neural Network

Here we explain the attention-based neural network for text classification where we use conventional Bi-GRU as the language representation layer and attention on top of it.

Because of its linked hidden layers, where the internal states are used to process data in a sequential fashion, recurrent neural networks (RNNs) [39] have lately become the standard for NLP. RNNs, on the other hand, have several drawbacks that have led to the creation of versions like LSTM and GRU. The GRU, like the LSTM unit, regulates the flow of information without using a memory unit, making it more efficient with near-lossless performance [40]. GRU also overcomes the issue of vanishing gradients and gradient explosions in vanilla RNN. Our selected model consists of a Bi-GRU layer on top of embedding layer initialized with Glove embedding. This layer consists of a attention layer on top of Bi-GRU. The overall architecture of proposed model is shown in Figure F.2.

Consider a sentence "This is wonderful movie." which is fed to the embedding layer initialized by Glove embedding. On the other hand, we obtain the clause score for each word in the sentence and feed to the embedding layer to match the dimension of input sentence embedding. Then both the embedding layer is passed to multiplication layer, where both are multiplied element wise. The output of the multiplication layer is then fed to the Bi-GRU having multiple hidden layers. Let us assume that the input to Bi-GRU is given by $X = [x_1, x_2, x_3, \ldots, x_k]$ where k is the padded length of the input sentence. This information is passed to Bi-GRU layer. In GRU, there are two types of gates: update gates and reset gates. The update gate determines how much previous data must be brought into the current state and how much new data must be introduced.

On the other hand, reset gate decides how much information from the previous steps is passed into the current state h_t. Here, h_t is the output from the GRU at time step t and z_t means the update gate. At a specific time step t, the new state h_t is given by:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h_t, \tag{F.5}$$

where $\odot$ represents the element-wise multiplication. To update z_t, we have

$$z_t = \sigma\left(W_{z_t} x_t + U_{z_t} h_{t-1} + b_{z_t}\right). \tag{F.6}$$

Here, x_t is each word of the sentence at time step t that is passed into the network unit which is then multiplied with its own weight W_{z_t}. Similarly, h_{t-1} represents the information of previous unit and is multiplied with its own weight U_{z_t} and b_{z_t} is the bias associated with update state. The current state h_t is updated using reset gate r_t by

$$h_t = \tanh\left(W_{h_t} x_t + r_t \odot \left(U_{h_t}\right) + b_{h_t}\right). \tag{F.7}$$

At r_t, the candidate state of step t can get the information of input x_t and the status of h_{t-1} of step $t-1$. The update function of r_t is given by

$$r_t = \sigma\left(W_{r_t} x_t + U_{r_t} h_{t-1} + b_{r_t}\right), \tag{F.8}$$

where W_{r_t} and U_{r_t} are the weights associated with the reset state and b_{r_t} is the bias.

The Bi-GRU consists the forward GRU layer $(\overrightarrow{h_t})$ that models the input sentence from step 0 to t and the backward GRU $(\overleftarrow{h_t})$ from t to 0.

$$\overrightarrow{h_t} = \overrightarrow{GRU}(x_t),\ t \in [1, T], \tag{F.9}$$

$$\overleftarrow{h_t} = \overleftarrow{GRU}(x_t),\ t \in [T, 1], \tag{F.10}$$

$$h_t = \left[\overrightarrow{h_t}, \overleftarrow{h_t}\right]. \tag{F.11}$$

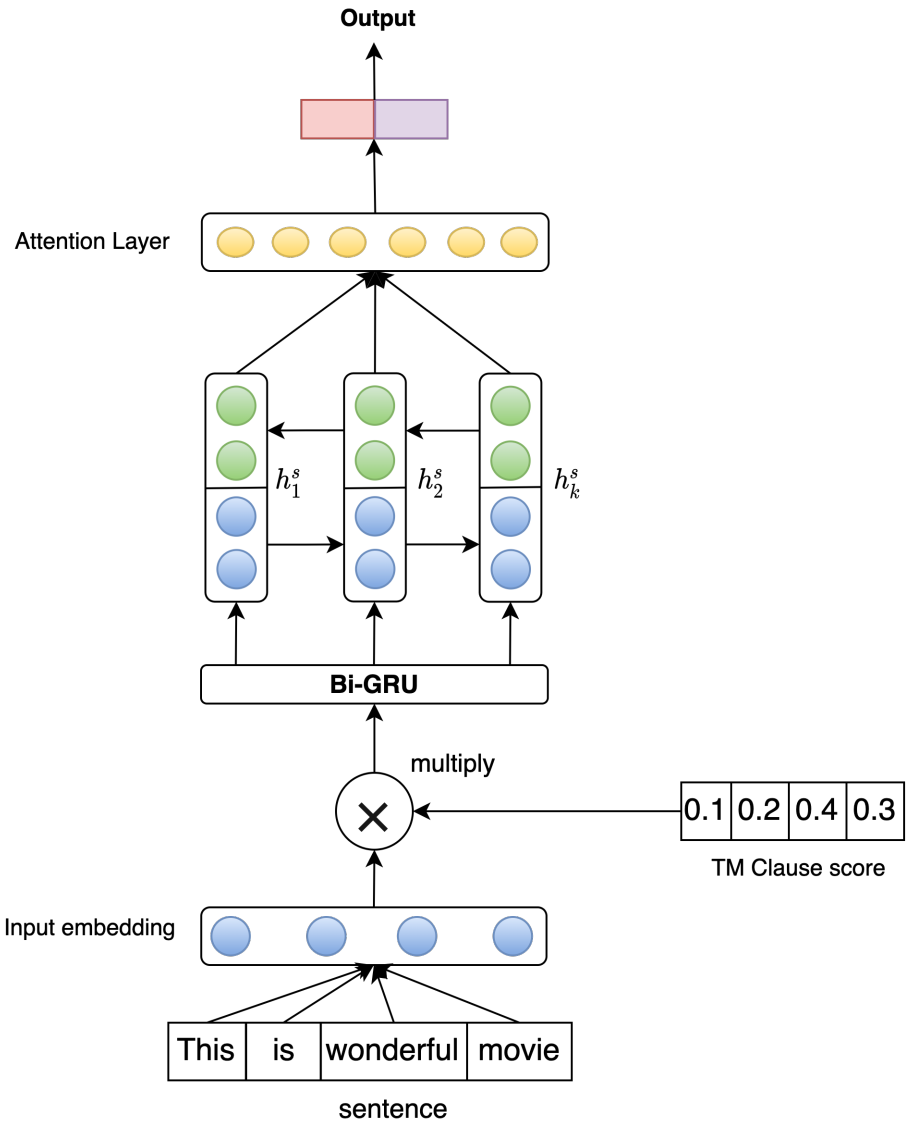


Figure F.2: The two-action TA and its transition in TM.

As we all know, not all of the words in the context contribute equally to text categorization. As a result, an attention layer is allocated to the context to prioritize significant words. Attention layer is fed on top of $Bi-GRU$ to learn the weight α_t for each hidden state h_t obtained at time step t. Since there are k inputs in the padded sequences, time step t will be from 1 to k. The weighting vector $\alpha = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_k)$ is calculated based on the output sequence $H = (h_1, h_2, h_3, \ldots, h_k)$. The attention vector s_1 for AL_1 is calculated based on the weighted sum of these hidden states, as:

$$s_1 = \sum_{t=1}^{k} (\alpha_t h_t), \tag{F.12}$$

where the weighted parameter α_t^1 is calculated by:

$$\alpha_t = \frac{\exp\left(u_t^T u_w\right)}{\sum_t \exp\left(u_t^T u_w\right)}, \tag{F.13}$$

where $u_t = \tanh(W_w h_t + b_w)$. Here W_w and h_t are the weight matrices and b_w represents the bias. The parameter u_w demonstrates context vector that is different at each time step, which is randomly initialized and learned jointly during the training process.

F.4 Experiments and Results

Here, we demonstrate the experiments and the result on the proposed model for enhancing the explanation of attention layer in text classification. We use two sentiment classification datasets for evaluation. They are:

- **MR** is a movie review dataset for binary sentiment classification with just one sentence per review [41]. There are 5331 positive reviews and 5331 critical reviews in the corpus. In this study, we used a training/test split from [42] (`https://github.com/mnqu/PTE/tree/master/data/mr` (24th Feb, 2022).).
- **Reuters** The Reuters 21,578 dataset has two subsets: R52 and R83 (all-terms version). R8 is divided into eight categories, including 5485 training and 2189 exam papers. R52 is divided into 52 categories and 6532 training and 2568 test papers.

We employ Keras [43] to implement our model. Adam [44] is used as the models' optimization method with the learning rate of $1 \times e^{-3}$. Additionally, we adopted Dropout [45] as the regularization strategy and the probability of Dropout was kept to be 0.25. Words are initialized with Glove [26] of 300-dimension word embedding. The batch size was 128 and was run for 100 epochs in the test datasets for obtaining the best results.

Since, the main purpose of this paper is to enhance the explanation of the attention layer, we demonstrate the performance of the proposed model with the relatable models to show the impact of each model. The comparable state-of-the-arts are explained below:

- **TF-IDF+LR**: Bag-of-words model with inverse document frequency weighting for term frequency. The classifier is based on logistic regression.
- **CNN**: CNN-rand uses arbitrarily initialized word embeddings [46].
- **LSTM**: The LSTM model that we employ here is from [47], representing the entire text using the last hidden layer. We used both the model that is using pretrained embeddings and without using.
- **Bi-LSTM**: Bi-directional LSTMs are widely used for text classification that models both forward and backward information.
- **PV-DBOW**: PV-DBOW is a paragraph vector model where the word order is not considered and is trained with Logistic Regression used as a softmax classifier [48].
- **PV-DM**: PV-DM is a paragraph vector model, with word ordering taken into consideration [48].
- **fastText**: This baseline uses the average of the word embeddings provided by fastText as document embedding. The embedding is then fed to a linear classifier [49].

- **SWEM**: SWEM applies simple pooling techniques over the word embeddings to obtain a document embedding [50].

- **Graph-CNN-C**: A graph CNN model uses convolutions over a word embedding similarity graph [51], employing a Chebyshev filter.

- **Tsetlin Machine**: Simple BOW model for Tsetlin Machine without feature enhancement.

- **Bi-GRU+Attn**: Bi-directional GRUs are widely used for text classification. We compare our model with Bi-GRU fed with pre-trained word embeddings along with attention layer on top of it.

- **TM+Bi-GRU+Attn**: Proposed model with Bi-GRU model with pretrained word embedding initialized with pretrained TM score in its input layer.

F.4.1 Performance Comparison with State-Of-The-Arts

Table F.3 shows the comparison of performance for selected datasets. As we can see that traditional method such as TF-IDF with Logistic Regression (TF-IDF+LR) performs decently in MR with 74.59, R8 with 93.74, and R8 with 86.95. Some sophisticated language model such as CNN, and LSTM performs quite similarly. The only improvement seen among them is Bi-LSTM which incorporates both past and future information for better input representation thereby reaching 77.06% in MR, 96.68% in R8, and 90.54% in R52. Slightly different than language models PV-DBOW and PV-DM performs poorly in all three datasets. Similarly, Graph-based CNN and SWEM perform on par with the state-of-the-arts baselines. On the other hand, the rule-based method TM performs quite comparable to baseline by reaching 75.14% in MR, 96.16% in R8, and 84.62% in R52. The performance is slightly below Bi-LSTM/GRU-based model because of its restriction to use pre-trained word embedding. However, Yadav et al. [22] show that embedding similar words using a pre-trained word embedding significantly enhances the performance and outperforms the baselines. However, our proposed model only uses TM explainability to generate prerequisite word weightage to replace human attention input into neural network language models. Hence, this is demonstrated in the table as well. Even though the motive of this task does not necessarily impact the accuracy but there is a slight increase in performance anyway. This is due to the fact that the TM score gives additional weightage to the model's input thereby reaching 77.95% in MR, 97.53% in R8, and 95.71% in R52 for TM+Bi-GRU+Attn. This shows an increment of about 1% in average throughout the selected datasets.

Models	MR	R8	R52
TF-IDF+LR	74.59	93.74	86.95
CNN	74.98	94.02	85.37
LSTM	75.06	93.68	85.54
Bi-LSTM	77.68	96.31	90.54
PV-DBOW	61.09	85.87	78.29
PV-DM	59.47	52.07	44.92
SWEM	76.65	95.32	92.94
Graph-CNN-C	77.22	96.99	92.75
Tsetlin Machine	75.14	96.16	84.62
Bi-GRU+Attn	77.15	96.20	94.85
TM+Bi-GRU+Attn	77.95	97.53	95.71

Table F.3: Performance of the proposed model (TM+Bi-GRU+Attn) with selected baselines.

In addition to this, we also evaluate some more metrics that supports the performance of the proposed model. Since MR is only binary classification dataset, R8 and R52 is multiclass dataset. Hence, there is need of the evaluation of the performance of each class. Usually unbalanced or multiclass datasets sometimes suffers with low F-scores because the model greedily learns the majority classes. Hence to have a clear picture of our proposed model, we evaluate precision, recall, and f-scores of main baseline TM, Bi-GRU+Attn with our proposed model TM+Bi-GRU+Attn as shown in Tables F.4–F.6 respectively. The results clearly indicate the our proposed model also performance superior on all three selected metrics for macro, micro, and weighted form of measurement compared to baselines TM and Bi-GRU+Attn. The performance of our proposed model is significantly higher in case of R8 and MR across all metrics. However the difference in performance for R52 is very marginal. In case of comparison with TM, our proposed outperforms all the measures for all three datasets.

Models	MR	R8	R52
Precision (macro)	73.22	86.12	79.18
Recall (macro)	70.42	87.44	75.44
F-Score (macro)	69.32	88.32	76.66
Precision (micro)	70.42	94.82	85.28
Recall (micro)	70.42	94.82	85.28
F-Score (micro)	70.42	94.82	85.28
Precision (weighted)	73.22	95.02	85.51
Recall (weighted)	70.42	95.12	85.12
F-Score (weighted)	69.32	95.02	85.28

Table F.4: Performance of TM for various evaluation metrics.

Models	MR	R8	R52
Precision (macro)	75.21	88.69	82.32
Recall (macro)	72.20	90.66	79.26
F-Score (macro)	71.34	89.26	79.87
Precision (micro)	72.20	95.52	95.63
Recall (micro)	72.20	95.52	95.63
F-Score (micro)	72.20	95.52	95.63
Precision (weighted)	75.21	95.60	95.33
Recall (weighted)	72.20	95.23	95.63
F-Score (weighted)	71.34	95.49	95.34

Table F.5: Performance of Bi-GRU+Attn for various evaluation metrics.

Models	MR	R8	R52
Precision (macro)	75.63	94.70	83.81
Recall (macro)	74.62	93.32	80.23
F-Score (macro)	74.61	93.39	80.67
Precision (micro)	74.62	96.52	96.82
Recall (micro)	74.62	96.52	96.82
F-Score (micro)	74.62	96.52	96.85
Precision (weighted)	75.63	96.58	96.51
Recall (weighted)	74.62	96.52	96.52
F-Score (weighted)	74.61	96.51	96.49

Table F.6: Performance of TM+Bi-GRU+Attn for various evaluation metrics.

F.4.2 Explainability

Here, we explore the proposed model's explainability by visualizing the respective attention weight. The attention weight usually gives the impact of each individual feature for a particular prediction. However, such weight usually gives the relationship between input and the output, such method of interpreting model can be beneficial for system to understand the impact of each features. Since neural network are already an established blackbox models, one can use this interpretation to generate explainability for the understanding the context of prediction. Hence we define interpretation of the model as the weights obtained from attention layer and explainability as use-case of interpretation to design the reasoning for a particular prediction that is easily understandable to humans. For ease of illustration, we visualize the attention weight of the Bi-GRU model and the attention weight of the Bi-GRU model initialized with TM's word score. We use the red color gradient to demonstrate the weightage of each input word in the context. Dark color represents the higher weightage with light color representing lower weightage. As we can see from Figure F.3, only using Bi-GRU, the model recognizes mostly important words for predicting correct sentiment class. However, it is not perfect as the human level. However, Figure F.4 shows the visualization of attention weight using Bi-GRU and TM's score.

Here we can see that the model focus on more significant words than the previous model. For instance, in the first example, the later model captures “look”, “away” with higher weightage which is an important context for negative sentiment than “directing” and “attempt”. This is more clearly seen in the third sample as the first model focus on “easily”, “best”, and “film” however our proposed model shifts the higher weightage to “best”, “Korean”, “film” for predicting the positive sentiment. One of the most peculiar cases where there are ambiguities in the context consisting of both positive and negative sentiment words as in the last example. Here using only Bi-GRU, the model captures “forgettable”, “rip”, and “work” as thigh-impact words. However, it does not give high weightage to the word “cheerful” which is also sentiment carrying word. However, using our proposed model, the weightage changes drastically and the model assigns higher weightage to “forgettable”, “cheerful”, “but”, and “earlier”. This makes more sense to human understanding because the context the has word “cheerful” and it is contradicted with the word “but” which eventually leads to a negative sentiment the carrying word “forgettable” thereby making the whole context negative sentiment.

Figure F.3: Visualization of attention weights with Bi-GRU only. Dark red to light red color represents the color gradients based on the attention weights in descending order.

Figure F.4: Visualization of attention weights with Bi-GRU and TM Score. Dark red to light red color represents the color gradients based on the attention weights in descending order.

F.5 Conclusions

Recently, attention weights have been a great tool for visualization of the weightage of input rationales in the model. However, their weightage sometimes gives higher weightage to unwanted

tokens that did not make sense to humans. This led to the requirement of human-annotated rationales that are embedded into the models. Even, such human annotators are not a very extensive task to obtain while annotating new datasets, the problems come annotating human rationales to existing datasets. It takes high time and cost to re-annotate human rationales for explainability. Hence, in this paper, we propose an alternative approach to get human explainable rationales using interpretable Tsetlin Machine (TM). Since TM can be explained using logical rules, it provides human-level interpretation and is used as a prerequisite annotation of input rationales. The proposed model shows that embedding such information in attention-based models not only increases the accuracy but also enhances the weightage of attention layer for each input rationales thereby making the explanation more sensible to humans. The visualization also shows that the proposed model is capable of capturing the ambiguity of the context much better than traditional models.

However, the concern with current study of explainability in AI is the subjectivity of explainability. Even though the mode of interpreting a model has been very sophisticated with proof of concept. It still fails to align with human understanding because of the subjectivity of opinion. Hence, as a future work, one can collect the human rationales annotation while manually labelling the particular datasets. This can be used as an evaluation criteria on how explainability of ML models align with various human understanding.

Bibliography

[1] Y. Zhang, I. J. Marshall, and B. C. Wallace, "Rationale-augmented convolutional neural networks for text classification," *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, vol. 2016, pp. 795–804, 2016.

[2] W. Wang, N. Yang, F. Wei, B. Chang, and M. Zhou, "Gated self-matching networks for reading comprehension and question answering," in *55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Vancouver, Canada), pp. 189–198, Association for Computational Linguistics, July 2017.

[3] H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction," in *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), p. 1675–1684, Association for Computing Machinery, 2016.

[4] C. J. Mahoney, J. Zhang, N. Huber-Fliflet, P. Gronvall, and H. Zhao, "A framework for explainable text classification in legal document review," *2019 IEEE International Conference on Big Data (Big Data)*, pp. 1858–1867, 2019.

[5] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[6] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *34th International Conference on Machine Learning - Volume 70*, ICML'17, p. 3319–3328, JMLR, 2017.

[7] O.-M. Camburu, T. Rocktäschel, T. Lukasiewicz, and P. Blunsom, "e-snli: Natural language inference with natural language explanations," in *NeurIPS*, 2018.

[8] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2015.

[9] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," in *Conference on Empirical Methods in Natural Language Processing*, (Austin, Texas), pp. 2249–2255, Association for Computational Linguistics, Nov. 2016.

[10] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *ArXiv*, vol. abs/1706.03762, 2017.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019.

[12] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *CoRR*, vol. abs/1312.6034, 2014.

[13] S. Jain and B. C. Wallace, "Attention is not Explanation," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 3543–3556, Association for Computational Linguistics, June 2019.

[14] S. Wiegreffe and Y. Pinter, "Attention is not not explanation," in *Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (Hong Kong, China), pp. 11–20, Association for Computational Linguistics, Nov. 2019.

[15] Z. C. Lipton, "The mythos of model interpretability," *Queue*, vol. 16, pp. 31 – 57, 2018.

[16] O.-C. Granmo, "The tsetlin machine - a game theoretic bandit driven approach to optimal pattern recognition with propositional logic," 2018.

[17] O.-C. Granmo, S. Glimsdal, L. Jiao, M. Goodwin, C. W. Omlin, and G. T. Berge, "The Convolutional Tsetlin Machine," 2019.

[18] R. K. Yadav, L. Jiao, O.-C. Granmo, and M. Goodwin, "Human-Level Interpretable Learning for Aspect-Based Sentiment Analysis," in *AAAI, Vancouver, Canada*, AAAI, 2021.

[19] B. Bhattarai, O.-C. Granmo, and L. Jiao, "Explainable tsetlin machine framework for fake news detection with credibility score assessment," 2021.

[20] K. D. Abeyrathna, B. Bhattarai, M. Goodwin, S. R. Gorji, O.-C. Granmo, L. Jiao, R. Saha, and R. K. Yadav, "Massively parallel and asynchronous tsetlin machine architecture supporting almost constant-time scaling," in *ICML*, pp. 10–20, PMLR, 2021.

[21] R. K. Yadav, L. Jiao, O.-C. Granmo, and M. Goodwin, "Interpretability in Word Sense Disambiguation using Tsetlin Machine," in *13th International Conference on Agents and Artificial Intelligence (ICAART), Vienna, Austria*, INSTICC, 2021.

[22] R. K. Yadav, L. Jiao, O.-C. Granmo, and M. Goodwin, "Enhancing interpretable clauses semantically using pretrained word representation," in *Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, (Punta Cana, Dominican Republic), pp. 265–274, Association for Computational Linguistics, 2021.

[23] O. Zaidan, J. Eisner, and C. Piatko, "Using annotator rationales to improve machine learning for text categorization," in *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, (Rochester, New York), pp. 260–267, Association for Computational Linguistics, Apr. 2007.

[24] Y. Zhang, I. Marshall, and B. C. Wallace, "Rationale-augmented convolutional neural networks for text classification," in *Conference on Empirical Methods in Natural Language Processing*, (Austin, Texas), pp. 795–804, Association for Computational Linguistics, Nov. 2016.

[25] K. Cho, B. van Merrienboer, Çaglar Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *EMNLP*, 2014.

[26] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, p. 1532–1543, 2014.

[27] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv: Machine Learning*, 2017.

[28] H. Liu, Q. Yin, and W. Y. Wang, "Towards explainable NLP: A generative explanation framework for text classification," in *57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 5570–5581, Association for Computational Linguistics, July 2019.

[29] A. A. Freitas, "Comprehensible classification models: a position paper," *SIGKDD Explor.*, vol. 15, pp. 1–10, 2014.

[30] Q. Zhang, Y. N. Wu, and S.-C. Zhu, "Interpretable convolutional neural networks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8827–8836, 2018.

[31] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, 2015.

[32] Y. Bao, S. Chang, M. Yu, and R. Barzilay, "Deriving machine attention from human rationales," in *Conference on Empirical Methods in Natural Language Processing*, (Brussels, Belgium), pp. 1903–1913, Association for Computational Linguistics, Oct.-Nov. 2018.

[33] A. K. Mohankumar, P. Nema, S. Narasimhan, M. M. Khapra, B. V. Srinivasan, and B. Ravindran, "Towards transparent and explainable attention models," in *58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 4206–4216, Association for Computational Linguistics, July 2020.

[34] T. McDonnell, M. Lease, M. Kutlu, and T. Elsayed, "Why is that relevant? collecting annotator rationales for relevance judgments," in *HCOMP*, 2016.

[35] X. Zhang, L. Jiao, O.-C. Granmo, and M. Goodwin, "On the Convergence of Tsetlin Machines for the IDENTITY- and NOT Operators," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[36] J. Sharma, R. Yadav, O.-C. Granmo, and L. Jiao, "Human Interpretable AI: Enhancing Tsetlin Machine Stochasticity with Drop Clause," *arXiv preprint arXiv:2105.14506*, 2021.

[37] J. Lei, A. Wheeldon, R. Shafik, A. Yakovlev, and O.-C. Granmo, "From Arithmetic to Logic Based AI: A Comparative Analysis of Neural Networks and Tsetlin Machine," in *27th IEEE International Conference on Electronics Circuits and Systems (ICECS2020)*, IEEE, 2020.

[38] J. Lei, T. Rahman, R. Shafik, A. Wheeldon, A. Yakovlev, O.-C. Granmo, F. Kawsar, and A. Mathur, "Low-Power Audio Keyword Spotting Using Tsetlin Machines," *Journal of Low Power Electronics and Applications*, vol. 11, 2021.

[39] T. Mikolov, M. Karafi, and S. Khudanpur, "Recurrent neural network based language model," in *INTERSPEECH*, 2010.

[40] J. Chung, Çaglar Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *ArXiv*, vol. abs/1412.3555, 2014.

[41] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *ACL*, (Michigan, USA), p. 115–124, ACL, 2005.

[42] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," in *21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, (Sydney, NSW, Australia), p. 1165–1174, Association for Computing Machinery, 2015.

[43] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[45] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[46] Y. Kim, "Convolutional neural networks for sentence classification," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1746–1751, Association for Computational Linguistics, Oct. 2014.

[47] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," in *IJCAI*, p. 2873–2879, 2016.

[48] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *31st International Conference on Machine Learning*, vol. 32 of *Machine Learning Research*, (Bejing, China), pp. 1188–1196, PMLR, 22–24 Jun 2014.

[49] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *EACL: Volume 2, Short Papers*, (Valencia, Spain), pp. 427–431, ACL, 2017.

[50] D. Shen, G. Wang, W. Wang, M. R. Min, Q. Su, Y. Zhang, C. Li, R. Henao, and L. Carin, "Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms," in *ACL (Volume 1: Long Papers)*, (Melbourne, Australia), pp. 440–450, ACL, 2018.

[51] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.