RheFrameDetect: A Text Classification System for Automatic Detection of Rhetorical Frames in AI from Open Sources

Saurav Ghosh
Vista Consulting LLC
Arlington, Virginia, USA
saurav.ghosh.vt@gmail.com

Philippe Loustaunau
Vista Consulting LLC
Arlington, Virginia, USA
ploustauanu@conseil-vista.com

ABSTRACT

Rhetorical Frames in AI can be thought of as expressions that describe AI development as a competition between two or more actors, such as governments or companies. Examples of such Frames include robotic arms race, AI rivalry, technological supremacy, cyberwarfare dominance and 5G race. Detection of Rhetorical Frames from open sources can help us track the attitudes of governments or companies towards AI, specifically whether attitudes are becoming more cooperative or competitive over time. Given the rapidly increasing volumes of open sources (online news media, twitter, blogs), it is difficult for subject matter experts to identify Rhetorical Frames in (near) real-time. Moreover, these sources are in general unstructured (noisy) and therefore, detecting Frames from these sources will require state-of-the-art text classification techniques. In this paper, we develop RheFrameDetect, a text classification system for (near) real-time capture of Rhetorical Frames from open sources. Given an input document, RheFrameDetect employs text classification techniques at multiple levels (document level and paragraph level) to identify all occurrences of Frames used in the discussion of AI. We performed extensive evaluation of the text classification techniques used in RheFrameDetect against human annotated Frames from multiple news sources. To further demonstrate the effectiveness of RheFrameDetect, we show multiple case studies depicting the Frames identified by RheFrameDetect compared against human annotated Frames.

KEYWORDS

Rhetorical Frames, Document Classification, Paragraph Classification, Doc2Vec, Paragraph Vector, Self-Attention, Attention Networks, FrameSpan Identification

1 INTRODUCTION

Advances in Artificial Intelligence (AI) have at times been described as an AI development race among major nations in the world. World leaders, such as Russian President Vladimir Putin, Chinese President Xi Jinping and Chief Technology Officer of The Trump Administration have issued statements in the past calling for greater prioritization, swift action and increased investment in order to gain strategic advantage in this AI development race. This paper focuses on how AI development is framed as a competition, either between nations or private companies. Our goal is to capture the Rhetorical Frames of Competition between governments or companies in the field of Artificial Intelligence (AI). This Frame may be expressed in military terms (such as, arms race, Cold War, battle for supremacy, etc.) or non-military (such as, win, versus, race, compete with, etc.). In Table 1, we show some examples and counter-examples of Rhetorical Frames as well as the rationale behind them. For a more detailed understanding of Rhetorical Frames, please refer to Imbrie et al. [6]

With massive online availability of open sources (online news media, twitter, blogs), identifying Rhetorical Frames using human annotators can be very limited in scope as well as difficult to scale in real-time. Detecting Rhetorical Frames from open sources in (near) real-time can assist policy makers or analysts in assessing whether attitudes of world governments or private companies towards AI development are becoming more cooperative or competitive over time. However, these textual open sources are unstructured (noisy) and therefore, pose challenges in identifying the occurrence of Frames.

The availability of massive textual open sources coincides with recent developments in language modeling, such as Word2Vec [10, 11], GloVe [13], Doc2Vec [7], Sentence-BERT (SBERT) [16] and others. These neural network based language models when trained over a representative corpus can be used to convert words as well as any piece of text (documents, paragraphs and sentences) to dense low-dimensional vector representations, most popularly known as embeddings. These embeddings are then provided as input to downstream classifiers for classifying a piece of text.

Motivated by these techniques, we develop RheFrameDetect, a text classification system for automatic identification of Rhetorical Frames from textual open sources. Our main contributions are as follows:

- **Automated**: RheFrameDetect is fully automatic. Given an input news article, it will automatically locate all the occurrences of Frames mentioned in the context of AI.
- **Novelty**: To the best of our knowledge, there has been no prior systematic efforts at automatic detection of Rhetorical Frames from open sources.
- **Real-time**: RheFrameDetect is an end-to-end system and can be deployed in a (near) real-time setting.
- **Evaluation**: We present a detailed and prospective analysis of RheFrameDetect by evaluating the accuracies of text classifiers employed at document as well as paragraph level against manually annotated Frames at the corresponding level.
- **Case Studies**: We also present multiple case studies providing insights into how Frames located by RheFrameDetect in news articles compare against the human annotated ones.

2 PROBLEM OVERVIEW

In this paper, we intend to focus on detecting as well as locating Rhetorical Frames mentioned in the context of AI within an incoming news article. RheFrameDetect aims to solve both these problems...
by performing multiple tasks in a hierarchical fashion as given below.

- **DocContainsAI**: If the incoming article contains a discussion of AI, then RheFrameDetect assigns a value of Yes to DocContainsAI. Otherwise, RheFrameDetect assigns the value of No and do not proceed for the document.
- **DocContainsFrame**: If the incoming article contains at least one occurrence of Frames, then RheFrameDetect assigns a value of Yes to DocContainsFrame. Otherwise, RheFrameDetect assigns the value of No and do not proceed further.
- **ParContainsFrame**: If the incoming document contains both a discussion of AI and at least one occurrence of Frames, RheFrameDetect aims to identify the paragraphs of the article which contain the Frames. Therefore, for each paragraph, RheFrameDetect assigns a value of Yes to ParContainsFrame if the paragraph contains at least one occurrence of Frames. Otherwise, RheFrameDetect assigns No to the paragraph.
- **FrameSpan**: For the paragraphs assigned Yes for ParContainsFrame, RheFrameDetect identifies all occurrences of the Frames mentioned in the paragraph.

A flowchart depicting the tasks performed by RheFrameDetect for an incoming article is shown in Figure 1.

### 3 RheFrameDetect

In this section, we provide a brief description of the techniques used by RheFrameDetect for solving the tasks shown in Figure 1.

#### 3.1 DocContainsAI

Given an incoming news article, the first task that RheFrameDetect performs is to detect whether the incoming article contains a discussion of AI. It is not necessary that AI be the main focus of the document. If there is any mention of AI or related keywords, then the field DocContainsAI for the article is set to Yes. For this task, RheFrameDetect employs a standard keyword search technique to determine whether the article text contains the term AI or related keywords.

#### 3.2 DocContainsFrame

If DocContainsAI is assigned the value of Yes, then the next task performed by RheFrameDetect is DocContainsFrame. If the article contains at least one occurrence of the Rhetorical Frame with reference to AI, the value Yes is assigned to DocContainsFrame, otherwise it is allocated the value No. Therefore, DocContainsFrame is a binary document classification task. To solve this task, RheFrameDetect employs multiple classifiers as follows: Logistic Regression, SVM, Multi-layer Perceptron and Random Forest. However, all these classifiers require converting the input document into a low-dimensional feature embedding. For generating document embeddings, RheFrameDetect employs two most popular and efficient techniques,

### Table 1: Examples and Counter-Examples of Rhetorical Frames mentioned in the context of AI and the rationale behind them

| Text                                                                 | Frame? | Rationale                                                                 |
|---------------------------------------------------------------------|--------|---------------------------------------------------------------------------|
| It is a great power rivalry focusing on technological supremacy    | Yes    | The phrase rivalry focusing on technological supremacy occurs in a discussion about AI and indicates a competition between two or more actors |
| Chinese AI company iFlytek often beats Facebook, Alphabet’s DeepMind, and IBM’s Watson in competitions to process natural speech | Yes    | The phrase beats Facebook, Alphabet’s DeepMind, and IBM’s Watson in competitions describes a competition between two or more actors in the context of AI. |
| China is outpacing other countries in the development of 5G today  | Yes    | Although one country is outpacing, this term entails a race, or competition between more than one actor, with the relative gains accruing to one side over another. |
| US-China trade war is making China stronger                        | No     | The phrase trade war is NOT an AI competition frame, because it is not used in the context of AI. |
| His company finds itself under fire, besieged by a U.S. effort to get key allies to ban its networking equipment. | No     | This is NOT an AI competition frame, because under fire and besieged only describe the actions of one actor; they do not entail competition or relative advantage. 

![Figure 1: Flow Chart depicting the tasks performed by RheFrameDetect for incoming news articles as input up to FrameSpan identification at the paragraph level.](image-url)
RheFrameDetect: A Text Classification System for Automatic Detection of Rhetorical Frames in AI from Open Sources

namely Doc2Vec [7] and SBERT [16] for converting documents into real-valued embeddings in a low-dimensional space. Document embeddings generated by Doc2Vec and SBERT are provided as input features to the downstream Machine Learning classifiers (Logistic Regression, SVM, Random Forest and Multi-layer Perceptron) for classifying DocContainsFrame.

3.3 ParContainsFrame

If DocContainsFrame is assigned the value of Yes, then the final binary classification task executed by RheFrameDetect is ParContainsFrame after segmenting the article into paragraphs. If each paragraph contains an instance of the Frame, RheFrameDetect assigns Yes to ParContainsFrame, otherwise No is assigned to ParContainsFrame. For ParContainsFrame, RheFrameDetect employs two types of classifiers as given below.

- **Paragraph Vector based Classifiers:** One of the properties of the Doc2Vec algorithm that make it so versatile, unique and powerful is that it can convert text (document or paragraph or sentence) of any length to low-dimensional real-valued embeddings. The same goes for the SBERT algorithm in that it can convert documents or paragraphs or sentences of any length into low-dimensional vectors. Therefore, both Doc2Vec and SBERT are applicable to paragraphs too. Paragraph vectors generated by Doc2Vec and SBERT algorithms are provided as input features to the downstream classifiers (Logistic Regression, SVM, Random Forest and Multi-layer Perceptron) for classifying ParContainsFrame.

- **Self-Attention based Classifiers:** Self-Attention [8] is an intra-attention mechanism that allows the words in a sentence or paragraph to interact with each other and discover which words in the sentence or paragraph should be paid more attention. RheFrameDetect applies Self-Attention mechanism on top of bi-directional LSTM [4] for classifying ParContainsFrame. Apart from classification, Self-Attention mechanism also allows us to locate the most important words in paragraphs in terms of contribution to the classification decision. Self-Attention based Classifiers take as input pre-trained GloVe [13] embeddings.

3.4 FrameSpan Identification (GuidedSelfAttention)

If ParContainsFrame is assigned a value of Yes, RheFrameDetect attempts to identify the FrameSpan within the paragraph as the final task in the overall process of detection of Frames from incoming news articles. Motivated by Rei et al. [15], RheFrameDetect employs a modified version of Self-Attention mechanism, namely GuidedSelfAttention on top of bi-directional LSTM to accomplish the task of identifying FrameSpan. In GuidedSelfAttention, the attention mechanism is modified in order to guide the attention values based on existing word-level FrameSpan annotations as follows.

- Firstly, GuidedSelfAttention converts the word-level FrameSpan annotation within a paragraph to a 0-1 encoding, E.g., a paragraph of five words with first and third word being Frames will be represented as [1, 0, 1, 0, 0].
- In the second step, the 0-1 encoding corresponding to word-level FrameSpan annotation is transformed into a probability distribution via normalization. E.g., [1, 0, 1, 0, 0] is converted to [0.5, 0, 0.5, 0, 0].
- Finally, GuidedSelfAttention minimizes the KL divergence between the probability distributions of attention weights and word-level FrameSpan annotation. As KL divergence is a measure of the difference between two probability distributions, minimizing it will encourage GuidedSelfAttention to apply more attention to the areas in paragraphs containing the FrameSpan.

Therefore, GuidedSelfAttention optimizes an additional objective of KL divergence between attention weights distribution and FrameSpan annotation distribution, apart from the ParContainsFrame classification objective. Minimizing this joint objective encourages GuidedSelfAttention to focus more on the tokens within FrameSpan by applying more attention weights to them.

4 EXPERIMENTAL EVALUATION

In this section, we provide a brief description of our experimental setup, including the dataset of news articles, the annotation process and the classifiers used by RheFrameDetect for the tasks DocContainsFrame and ParContainsFrame.

4.1 News Articles Corpus

To explore the use of Rhetorical Frames, we analyzed a corpus of news articles related to AI from four different sources, namely Reuters News Agency, Defense One, Foreign Affairs and LexisNexis news database [1] over the period 2012 to 2019. These news sources were selected based on three criteria: firstly, they have a broad, global news coverage; secondly, they offered a diversity of news analysis, commentary, and opinion; and thirdly, they are more consistently editorially structured than speeches, blogs, or social media posts. In Table 2, we show the total number of news articles containing AI and Frames as well as the total number of paragraphs across articles containing Frames for each source. As can be seen in Table 2, both binary classification tasks DocContainsFrame and ParContainsFrame have a class imbalance of 14:1 and 13:1 respectively, with Yes being the minority class. Therefore, it is extremely challenging to predict whether a document or a paragraph contains Frames as classification data is heavily skewed towards documents or paragraphs not having any Frames. All articles were provided by the Center for Security and Emerging Technology (CSET) at Georgetown University [6].

4.2 Human Annotation

Analysts at CSET helped identify Rhetorical Frames in an initial small set of news articles. From this, we developed an initial annotation guide that we tested with our annotation service provider (Spi Global). This involved multiple back and forth with the analysts at CSET. We finalized the annotation guidelines and provided 19K documents to Spi Global for annotation. We also asked Spi Global to doubly and independently annotate 10% of these documents to assess inter-coder agreement. For each article the annotators were asked to determine whether the article discussed AI, if so, whether it contained the Rhetorical Frame. For documents that contained the Frame, and for each paragraph in that document, the annotators were asked to determine whether the paragraph contained the
As per Figure 1, we have two binary classification tasks: 1) Doc-contains-frame and 2) Par-contains-frame as follows.

4.3 Text Pre-processing

The textual content of each article in the corpus was pre-processed using SpaCy [5], Gensim [14], NLTK [9] and Keras [3]. Pre-processing steps involve sentence splitting, tokenization, token lower-casing, noise removal, and ignoring too short or too long tokens.

4.4 Classifiers

As per Figure 1, we have two binary classification tasks: 1) Doc-contains-frame at the article level and 2) Par-contains-frame at the paragraph level.

For Doc-contains-frame, RheFrameDetect evaluated two types of classifiers as follows.

- **Doc2Vec based Classifiers**: Doc2Vec embeddings provided as input features to downstream classifiers, namely Logistic Regression, SVM, Random Forest and Multi-layer Perceptron. We compared four variants of Doc2Vec: 1) Distributed Bag of Words with hierarchical softmax (DV-DBOW-NS) and negative sampling (DV-DBOW-NEG), 2) Distributed Memory with hierarchical softmax (DV-DM-NS) and Negative Sampling (DV-DM-NEG). For each of these models, we used 300 embedding dimension, a window size of 5 and 5 negative samples if negative sampling is used. Our corpus of 18972 AI related news articles was used for training the Doc2Vec models to generate the embeddings.

- **SBERT based Classifiers**: SBERT embeddings for documents provided as input features to downstream classifiers, namely Logistic Regression, SVM, Random Forest and Multi-layer Perceptron. For SBERT model, we used 768 embedding dimension. We used the implementation of SBERT model all-mpnet-base-v2 within the SentenceTransformer package of HuggingFace. SBERT model all-mpnet-base-v2 maps each sentence/paragraph/document to a dense 768-dimensional vector space and was trained on a large and diverse dataset of over 1 billion training sentence pairs.

For Par-contains-frame, RheFrameDetect compared the following classifiers.

- **Paragraph Vector based Classifiers**: Paragraph Vectors [7] provided as input features to the same downstream classifiers as used in Doc2Vec based classifiers for Doc-contains-frame. Similar to Doc2Vec, we also compared four variants of Paragraph Vector: PV-DBOW-NS, PV-DBOW-NEG, PV-DM-NS and PV-DM-NEG. We used the same parameter settings for generation of Paragraph Vectors (300 embedding dimension, window size of 5 and 5 negative samples if negative sampling is used) as in Doc2Vec. We extracted all the sentences from our corpus of 18972 AI related news articles and generated a sentence corpus comprising of nearly 1 million sentences for training the Paragraph Vector models.

- **SBERT based classifiers**: SBERT 768-dimensional embeddings for paragraphs generated using all-mpnet-base-v2 model within the SentenceTransformer package of HuggingFace provided as input features to the same downstream classifiers as used for Doc-contains-frame.

- **SelfAttention**: Self-Attention mechanism [8] that learns Par-contains-frame with GloVe word embeddings as input.

- **GuidedSelfAttention**: Self-Attention mechanism [8] that jointly learns Par-contains-frame and FrameSpan with GloVe word embeddings as input.

Doc2Vec and Paragraph Vectors were implemented using Gensim [14]. SBERT embeddings were generated using the Sentence-Transformer package of HuggingFace. For each of the four downstream classifiers (Logistic Regression, SVM, Random Forest and Multi-layer Perceptron), we used a grid of parameter settings as shown below and then used scikit-learn [12]’s GridSearchCV (10-fold cross-validation) to perform an exhaustive search over the parameter grid for each classifier.

- **Logistic Regression**: penalty: ['l1', 'l2', 'elasticnet'], C: [0.1, 1., 10., 100., 1000.]
- **SVM**: kernel: ['rbf', 'poly', 'linear'], C: [0.1, 1., 10., 100., 1000.], degree: [3, 5, 7, 8]
- **Multi-layer Perceptron**: hiddenlayersizes: [(100,)], learningrate: ['constant', 'invscaling', 'adaptive'], activation: ['logistic', 'identity', 'tanh', 'relu'], alpha: [0.01, 0.01, 0.001, 0.0001]
- **Random Forest**: estimators: [10, 50, 100, 200, 500, 1000], maxfeatures: ['log2', 'sqrt']

For all the other parameters (not part of GridSearchCV) in each classifier, we used the default parameter setting as in scikit-learn. The GridSearchCV procedure gives us the optimal parameter setting for each downstream classifier and we refit each classifier on the entire training set using the optimal parameter setting following the GridSearchCV procedure. To account for the class imbalance in both Doc-contains-frame and Par-contains-frame, we used the parameter classweight: balanced in all of the classifiers. SelfAttention and GuidedSelfAttention were implemented using Keras [3] and Tensorflow [2]. For both SelfAttention and GuidedSelfAttention, we employed EarlyStopping for monitoring cross-entropy loss on the validation dataset (10 percent of training set). If the validation loss is no longer decreasing for 2-3 iterations consecutively, we stop the training, thereby minimizing the risk of potential overfitting for both SelfAttention and GuidedSelfAttention.

4.5 Performance Metrics

We evaluated the classifiers for each binary classification task (Doc-contains-frame and Par-contains-frame) in terms of the following performance metrics.

- **Precision** for each class (Yes and No) as well as macro-average across the classes
- **Recall** for each class as well as macro-average across the classes
- **F1-score** for each class as well as macro-average across the classes

We used scikit-learn’s Repeated Stratified 5-Fold cross validator for splitting the annotation data corresponding to each binary classification task into 5 folds. For each fold, we calculate the score for each metric and report the average metric score across the folds.
We also show the counts of paragraphs across articles for each source with ParContainsFrame assigned Yes or No. We also show the counts of paragraphs across articles for each source with ParContainsFrame assigned Yes or No.

| Classification Task | Class | Reuters | Defense One | Foreign Affairs | LexisNexis |
|---------------------|-------|---------|-------------|-----------------|------------|
| DocContainsAI       | Yes   | 3496    | 537         | 55              | 9984       |
|                     | No    | 4205    | 667         | 11              | 16         |
| DocContainsFrame    | Yes   | 249     | 43          | 1               | 649        |
|                     | No    | 3247    | 494         | 54              | 9335       |
| ParContainsFrame    | Yes   | 391     | 79          | 1               | 1032       |
|                     | No    | 4934    | 798         | 29              | 13998      |

## 5 RESULTS

In this section, we try to ascertain the efficacy and applicability of RheFrameDetect by investigating some of the pertinent questions related to the problem of automated detection of Rhetorical Frames.

**Doc2Vec based Classifiers vs SBERT based classifiers - Which is a better method for detecting Rhetorical Frames at the document level?**

In Tables 4 and 5, we evaluated Doc2Vec and SBERT based classifiers for the binary classification task DocContainsFrame. For DocContainsFrame, we are mostly interested in the performance metrics (Precision, Recall and F1-score) for Yes class as it is the minority class and therefore, hard to predict. Moreover, performance metrics for the Yes class will inform us whether classifiers can detect presence of Frames at the document level. We are also interested in the macro-average metrics across the classes. Overall, in terms of F1-score for Yes class and macro-average, Multi-layer Perceptron with embedding DV-DBOW-NEG and SVM with embedding SBERT emerge out to be the best performing classifiers. In terms of Recall, Logistic Regression consistently emerges out to be the best classifier but performs worse in terms of Precision leading to low F1-score than SVM and Multi-layer Perceptron. Random Forest outperforms other classifiers in terms of Precision but performs worse in terms of Recall leading to very poor F1-score compared to SVM and Multi-layer Perceptron.

If we compare Doc2Vec vs SBERT in terms of the F1-score, we do not notice any significant performance difference. However, in terms of Recall, Logistic Regression performs better with Doc2Vec embeddings than SBERT. Similarly, for Precision, Random Forest performs better with Doc2Vec embeddings than SBERT. This depicts that Doc2Vec generates slightly better embeddings at the document level than SBERT. Since SBERT technique is trained over a data set of nearly billion sentences, it may not be the ideal technique for generating embeddings at the document level.

**Paragraph Vector and SBERT based Classifiers vs Self-Attention based Classifiers - Which is a better method for detecting Rhetorical Frames at the paragraph level?**

In Tables 6, 7 and 8, we evaluated Paragraph Vector and SBERT based classifiers against the Self-Attention based classifiers for the binary classification task ParContainsFrame.

Firstly, we compare Paragraph Vector based classifiers against SBERT based classifiers for this task. Similar to DocContainsFrame, we are mostly interested in the performance metrics (Precision, Recall and F1-score) for Yes class as it is the minority class and therefore, hard to predict. Moreover, performance metrics for the Yes class will inform us whether classifiers can detect presence of Frames at the paragraph level. As can be clearly seen, SVM with SBERT embeddings outperform all the other techniques in terms of F1-Score for the Yes class as well as macro-average. Overall, with both Doc2Vec and SBERT embeddings, Multi-layer Perceptron emerges out to be the best performing classifier in terms of F1-Score followed by SVM. Similar to DocContainsFrame, Logistic Regression is the best performing classifier in terms of Recall for ParContainsFrame and Random Forest outperforms other classifiers in terms of Precision.

When we compared Paragraph Vector against SBERT across the four classifiers, SBERT based classifiers provide better F1-score than the corresponding Paragraph Vector based classifiers. SBERT model is trained over a dataset of billion sentences in comparison to Paragraph Vector trained over our corpus dataset of nearly 1 million sentences and therefore, SBERT model is expected to generate better quality embeddings for sentences/paragraphs than Paragraph Vector.

Finally, we compared the Self-Attention based classifiers (SelfAttention and GuidedSelfAttention) against the Paragraph Vector and SBERT based classifiers. As expected, Self-Attention based classifiers (specifically GuidedSelfAttention) outperform Paragraph Vector and SBERT based classifiers in terms of Recall and F1-Score due to the powerful Self-Attention mechanism guided by the FrameSpan annotations. For the Yes class of ParContainsFrame, GuidedSelfAttention achieves the highest Recall (0.87) and F1-Score (0.81) among all the classifiers. Superior performance of Self-Attention based classifiers is expected as they are sequence based models. Therefore, they learn the sequential patterns of Frames in paragraphs.
more efficiently compared to the Paragraph Vector based classifiers. Moreover, Self-Attention mechanism provides dual benefits to the classifier: not only does it result in better classification metrics, but it also drives the classifier to locate the most relevant words in paragraphs/sentences which contribute to the classification decision.

6 CASE STUDIES: FRAMESPAN IDENTIFICATION

Finally, we also depict five case studies related to FrameSpan identification after ParContainsFrame classification. For each case study, we chose a paragraph from the test dataset of ParContainsFrame classification and then, we visualized the attention weights of tokens in each paragraph assigned by SelfAttention and GuidedSelfAttention models. In Table 3, we show the selected paragraphs and the FrameSpan mentioned within each paragraph. In Figure 2, we show the visualization of the attention weights for both SelfAttention and GuidedSelfAttention. As can be seen clearly in Figure 2, GuidedSelfAttention is able to locate the tokens within FrameSpan more prominently in comparison to SelfAttention. Specifically, in paragraphs corresponding to case studies 2 and 5, GuidedSelfAttention captures accurately the FrameSpan ‘compete’ and ‘competition’ respectively. This is expected as the attention weights in GuidedSelfAttention are supervised based on the FrameSpan annotations. This motivates GuidedSelfAttention to apply more attention to the portions in paragraphs which may contain FrameSpan. On the other hand, attention weights in SelfAttention model are unsupervised, i.e. it does not have any information about the FrameSpan location in paragraphs. Therefore, GuidedSelfAttention not only outperforms SelfAttention in terms of the metrics (Recall and F1-Score) for the Yes class in ParContainsFrame but also has the capability to accurately locate FrameSpan information within the paragraph leading to enhanced intelligibility from a human analyst perspective.

7 CONCLUSIONS AND FUTURE WORK

In this manuscript, we have introduced RheFrameDetect, the first automated end-to-end system for detecting and locating Rhetorical Frames in AI from Open Sources. RheFrameDetect takes an incoming news article as input and performs multiple tasks (Text Pre-processing, Keyword matching and binary classifications) in a hierarchical manner. Binary classification tasks performed by RheFrameDetect were used to detect Frames at both document level (DocContainsFrame) and paragraph level (ParContainsFrame). We evaluated a wide variety of classifiers for both these tasks. For DocContainsFrame classification task, we found that Multi-layer Perceptron and SVM emerge out to be the best performing classifiers in terms of F1-Score for the Yes class and macro-average. In terms of Recall and Precision for DocContainsFrame, Doc2Vec based classifiers perform slightly better than SBERT based classifiers indicating better quality of embeddings generated at the document level by Doc2Vec than SBERT. For the ParContainsFrame classification task, GuidedSelfAttention classifier outperforms all other classifiers achieving a macro-average F1-Score of 0.90. For the Yes class, GuidedSelfAttention achieves the highest Recall of 0.87 and F1-Score of 0.81. We showed in Figure 2 that GuidedSelfAttention also has the capability to accurately locate FrameSpan information within the paragraph compared to SelfAttention. Superior performance of GuidedSelfAttention can be attributed to the fact that it directs the Self-Attention mechanism to put more emphasis (higher weights) on words/tokens within FrameSpan during training, ignoring the other words/tokens in paragraphs.

Our future work will focus on expanding RheFrameDetect to detect the purpose of Rhetorical Frames in AI, such as Motivation, Critique, Structured and Explanation. We also aim to quantify the FrameSpan identification task in terms of Precision, Recall and F1-Score using rankings based on attention weights of words/tokens in each paragraph. Finally, we also aim to extract named entities mentioned in the context of Frames within a paragraph so that we can provide human analysts more context around the detected Frames.

8 ACKNOWLEDGEMENTS

This work was performed under contract with the Center for Security and Emerging Technology (CSET) at Georgetown University, contract number CON-0011700. We would also like to thank SPI Global for providing us AI Rhetorical Frame annotations at the document and paragraph level.

REFERENCES

[1] 2020. Nexis Metabase. (2020). https://www.lexisnexis.com/en-us/professional/data-as-a-service/DaaS-Metabase-Product page.
[2] Martin Abadi, Ashish Agarwal, and Paul Barham et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. http://tensorflow.org/ Software available from tensorflow.org.
[3] François Chollet. 2015. Keras. https://github.com/fchollet/keras.
[4] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. Neural Comput. 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
[5] M. Honnibal and M. Johnson. 2015. An Improved Non-monotonic Transition System for Dependency Parsing. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Lisbon, Portugal, 1373–1378. https://aclweb.org/anthology/D/D15/D15-1162
[6] Andrew Imbrie, James Dunham, Rebecca Gelles, and Catherine Aiken. 2020. Mainframes: A Provisional Analysis of Rhetorical Frames in AI. Center for Security and Emerging Technology (August 2020). https://doi.org/10.51953/20190046
[7] Q. V. Le and T. Mikolov. 2014. Distributed Representations of Sentences and Documents. In IJCAI, Vol. 14:1188–1196.
[8] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130 (2017).
[9] Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. In In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics.
[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient Estimation of Word Representations in Vector Space. CoRR abs/1301.3781 (2013). http://arxiv.org/abs/1301.3781
[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In 26th Annual Conference on Neural Information Processing Systems. 3111–3119.
[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12 (2011), 2825–2830.
[13] J. Pennington, R. Socher, and C. D. Manning. 2014. Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. 1532–1543. http://aclweb.org/anthology/D/D14/D14-1162.pdf
[14] Radim Rehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. ELRA, Valletta, Malta, 45–50. http://is.muni.cz/publication/884893/en.
[15] Marek Rei and Anders Søgaard. 2019. Jointly learning to label sentences and tokens. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33. 6916–6923.
[16] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084 (2019).
When you speak about artificial intelligence, machine learning, everyone admits that Amazon is so far ahead of everybody else,” Cramer added.

Amazon, Microsoft and IBM are investing billions in virtualizing video services technologies and artificial intelligence. It is arguably the biggest battle on the internet given that video accounts for nearly 80 percent of internet traffic.

Enterprise Monkey has established itself as a leader in web and app development space and now growing its leadership in Artificial Intelligence, Augmented Reality and Internet of Things space. The client list includes NASDAQ listed companies as well as seed-stage startups.

The car’s motor controls were modified so driving was possible at that angle and the robot slipped easily out of its vehicle in record time. Many of the other robots in the competition failed to dismount and had the indignity of their teams wheeling over the safety harness to tether and lift them out, losing points and time before continuing other tasks.

The client list includes NASDAQ listed companies as well as seed-stage startups.

Traditional automakers from Detroit and around the world are responding by trying to build a greater presence in the Silicon Valley region to compete for both technology and engineering talent.

Amazon, Microsoft and IBM are investing billions in virtualizing video services technologies and artificial intelligence. Augmented Reality and Internet of Things space. The client list includes NASDAQ listed companies as well as seed-stage startups.

Microsoft and IBM are investing billions in virtualizing video services technologies and artificial intelligence. Augmented Reality and Internet of Things space.

The car’s motor controls were modified so driving was possible at that angle and the robot slipped easily out of its vehicle in record time. Many of the other robots in the competition failed to dismount and had the indignity of their teams wheeling over the safety harness to tether and lift them out, losing points and time before continuing other tasks.

Enterprise Monkey has established itself as a leader in web and app development space and now growing its leadership in Artificial Intelligence, Augmented Reality and Internet of Things space. The client list includes NASDAQ listed companies as well as seed-stage startups.

Table 3: Case studies for FrameSpan identification where we show the paragraphs and the Frames mentioned within the paragraphs for each case study

| Case Study | Paragraph | FrameSpan |
|------------|-----------|-----------|
| 1          | When you speak about artificial intelligence, machine learning everyone admits that Amazon is so far ahead of everybody else, Cramer added. | ahead of everybody else |
| 2          | Traditional automakers from Detroit and around the world are responding by trying to build a greater presence in the Silicon Valley region to compete for both technology and engineering talent. | compete |
| 3          | Enterprise Monkey has established itself as a leader in web and app development space and now growing its leadership in Artificial Intelligence, Augmented Reality and Internet of Things space. The client list includes NASDAQ listed companies as well as seed-stage startups. | growing its leadership in Artificial Intelligence |
| 4          | Amazon, Microsoft and IBM are investing billions in virtualizing video services technologies and artificial intelligence. Augmented Reality and Internet of Things space. It is arguably the biggest battle on the internet given that video accounts for nearly 80 percent of internet traffic. | biggest battle on the internet |
| 5          | The car’s motor controls were modified so driving was possible at that angle and the robot slipped easily out of its vehicle in record time. Many of the other robots in the competition failed to dismount and had the indignity of their teams wheeling over the safety harness to tether and lift them out, losing points and time before continuing other tasks. | competition |

Figure 2: Comparing the weights of words in paragraphs for each case study as shown in Table 3 assigned by SelfAttention and GuidedSelfAttention models. Words in paragraphs are highlighted corresponding to their weights. We can see that GuidedSelfAttention is able to locate the Frames in each paragraph more accurately than SelfAttention.
Table 4: Comparing Doc2Vec (DV-DM-NEG and DV-DBOW-NEG) based Classifiers for DocContainsFrame in terms of Precision, Recall and F1-score for each class (Yes and No) as well as macro-average across the classes.

| Embedding          | Classifier           | Class  | Precision | Recall | F1-score |
|--------------------|----------------------|--------|-----------|--------|----------|
|                    | Logistic Regression  | No     | 0.97      | 0.75   | 0.84     |
|                    |                      | Yes    | 0.15      | **0.63**| 0.24     |
|                    |                      | Macro-average | 0.56 | **0.69** | 0.54     |
| DV-DM-NEG          | SVM                  | No     | 0.95      | 0.93   | 0.94     |
|                    |                      | Yes    | 0.24      | 0.29   | **0.26** |
|                    |                      | Macro-average | 0.59 | 0.61   | **0.60** |
|                    | Multi-layer Perceptron | No | 0.94      | 0.99   | 0.97     |
|                    |                      | Yes    | 0.53      | 0.08   | 0.13     |
|                    |                      | Macro-average | 0.73 | 0.54   | 0.55     |
|                    | Random Forest        | No     | 0.93      | 1.0    | 0.97     |
|                    |                      | Yes    | **1.0**   | 0.01   | 0.02     |
|                    |                      | Macro-average | **0.97** | 0.50 | 0.49     |
|                    | Logistic Regression  | No     | 0.97      | 0.77   | 0.86     |
|                    |                      | Yes    | 0.17      | **0.67**| 0.27     |
|                    |                      | Macro-average | 0.57 | **0.72** | 0.56     |
| DV-DBOW-NEG        | SVM                  | No     | 0.95      | 0.98   | 0.96     |
|                    |                      | Yes    | 0.37      | 0.23   | 0.29     |
|                    |                      | Macro-average | 0.66 | 0.61   | 0.63     |
|                    | Multi-layer Perceptron | No | 0.95      | 0.99   | 0.97     |
|                    |                      | Yes    | 0.54      | 0.24   | **0.33** |
|                    |                      | Macro-average | 0.75 | 0.61   | **0.65** |
|                    | Random Forest        | No     | 0.93      | 1.00   | 0.97     |
|                    |                      | Yes    | **0.85**  | 0.01   | 0.03     |
|                    |                      | Macro-average | **0.89** | 0.51 | 0.50     |
Table 5: Comparing Doc2Vec (DV-DM-HS and DV-DBOW-HS) based Classifiers for DocContainsFrame in terms of Precision, Recall and F1-score for each class (Yes and No) as well as macro-average across the classes.

| Embedding   | Classifier       | Class | Precision | Recall | F1-score |
|-------------|------------------|-------|-----------|--------|----------|
|             |                  |       |           |        |          |
| DV-DM-HS    | Logistic Regression | No | 0.96 | 0.74 | 0.84 |
|             |                  | Yes | 0.15 | **0.61** | 0.24 |
|             |                  | Macro-average | 0.55 | **0.68** | 0.54 |
|             | SVM              | No | 0.94 | 0.99 | 0.96 |
|             |                  | Yes | 0.08 | 0.06 | 0.07 |
|             |                  | Macro-average | 0.51 | 0.53 | 0.52 |
|             | Multi-layer Perceptron | No | 0.95 | 0.99 | 0.97 |
|             |                  | Yes | 0.51 | 0.20 | **0.29** |
|             |                  | Macro-average | 0.73 | 0.59 | **0.63** |
|             | Random Forest    | No | 0.93 | 1.00 | 0.97 |
|             |                  | Yes | **0.97** | 0.02 | 0.03 |
|             |                  | Macro-average | **0.95** | 0.51 | 0.50 |
|             | Logistic Regression | No | 0.97 | 0.75 | 0.85 |
|             |                  | Yes | 0.16 | **0.66** | 0.26 |
|             |                  | Macro-average | 0.57 | **0.71** | 0.55 |
|             | SVM              | No | 0.95 | 0.98 | 0.96 |
|             |                  | Yes | 0.37 | 0.23 | 0.28 |
|             |                  | Macro-average | 0.66 | 0.61 | 0.62 |
|             | Multi-layer Perceptron | No | 0.95 | 0.99 | 0.97 |
|             |                  | Yes | 0.57 | 0.22 | **0.31** |
|             |                  | Macro-average | 0.76 | 0.60 | **0.64** |
|             | Random Forest    | No | 0.93 | 1.00 | 0.97 |
|             |                  | Yes | **0.75** | 0.03 | 0.06 |
|             |                  | Macro-average | **0.84** | 0.51 | 0.51 |
|             | Logistic Regression | No | 0.95 | 0.85 | 0.90 |
|             |                  | Yes | 0.15 | **0.40** | 0.20 |
|             |                  | Macro-average | 0.55 | **0.62** | 0.55 |
|             | SVM              | No | 0.95 | 0.98 | 0.96 |
|             |                  | Yes | 0.50 | 0.25 | **0.33** |
|             |                  | Macro-average | 0.72 | **0.62** | **0.65** |
|             | Multi-layer Perceptron | No | 0.94 | 1.00 | 0.97 |
|             |                  | Yes | 0.65 | 0.08 | 0.14 |
|             |                  | Macro-average | 0.79 | 0.54 | 0.55 |
|             | Random Forest    | No | 0.94 | 1.00 | 0.97 |
|             |                  | Yes | **0.75** | 0.08 | 0.14 |
|             |                  | Macro-average | **0.84** | 0.54 | 0.56 |
Table 6: Comparing Paragraph Vector (PV-DM-NEG and PV-DBOW-NEG) based Classifiers for ParContainsFrame in terms of Precision, Recall and F1-score for each class (Yes and No) as well as macro-average across the classes

| Embedding   | Classifier       | Class      | Precision | Recall | F1-score |
|-------------|------------------|------------|-----------|--------|----------|
| PV-DM-NEG   | Logistic Regression | No         | 0.97      | 0.84   | 0.90     |
| PV-DM-NEG   | Logistic Regression | Yes        | 0.25      | 0.71   | 0.37     |
| PV-DM-NEG   | Logistic Regression | Macro-average | 0.61      | 0.77   | 0.64     |
| PV-DM-NEG   | SVM               | No         | 0.95      | 0.97   | 0.96     |
| PV-DM-NEG   | SVM               | Yes        | 0.40      | 0.29   | 0.34     |
| PV-DM-NEG   | SVM               | Macro-average | 0.67      | 0.63   | 0.65     |
| PV-DM-NEG   | Multi-layer Perceptron | No       | 0.95      | 0.99   | 0.97     |
| PV-DM-NEG   | Multi-layer Perceptron | Yes     | 0.61      | 0.29   | 0.39     |
| PV-DM-NEG   | Multi-layer Perceptron | Macro-average | 0.78      | 0.64   | 0.68     |
| PV-DM-NEG   | Random Forest     | No         | 0.93      | 1.00   | 0.96     |
| PV-DM-NEG   | Random Forest     | Yes        | 0.83      | 0.01   | 0.01     |
| PV-DM-NEG   | Random Forest     | Macro-average | 0.88      | 0.50   | 0.49     |
| PV-DBOW-NEG | Logistic Regression | No         | 0.98      | 0.84   | 0.91     |
| PV-DBOW-NEG | Logistic Regression | Yes        | 0.28      | 0.80   | 0.41     |
| PV-DBOW-NEG | Logistic Regression | Macro Average | 0.63      | 0.82   | 0.66     |
| PV-DBOW-NEG | SVM               | No         | 0.95      | 0.97   | 0.96     |
| PV-DBOW-NEG | SVM               | Yes        | 0.48      | 0.40   | 0.43     |
| PV-DBOW-NEG | SVM               | Macro-average | 0.72      | 0.68   | 0.70     |
| PV-DBOW-NEG | Multi-layer Perceptron | No     | 0.95      | 0.99   | 0.97     |
| PV-DBOW-NEG | Multi-layer Perceptron | Yes    | 0.67      | 0.37   | 0.47     |
| PV-DBOW-NEG | Multi-layer Perceptron | Macro-average | 0.81      | 0.68   | 0.72     |
| PV-DBOW-NEG | Random Forest     | No         | 0.93      | 1.00   | 0.96     |
| PV-DBOW-NEG | Random Forest     | Yes        | 0.85      | 0.01   | 0.03     |
| PV-DBOW-NEG | Random Forest     | Macro-average | 0.89      | 0.51   | 0.49     |
Table 7: Comparing Paragraph Vector (PV-DM-HS and PV-DBOW-HS) and SBERT based Classifiers for ParContainsFrame in terms of Precision, Recall and F1-score for each class (Yes and No) as well as macro-average across the classes.

| Embedding         | Classifier     | Class       | Precision | Recall | F1-score |
|-------------------|----------------|-------------|-----------|--------|----------|
|                   | Logistic Regression | No        | 0.97      | 0.81   | 0.88     |
|                   |                 | Yes        | 0.22      | 0.73   | 0.34     |
| PV-DM-HS          |                 | Macro-average | 0.60      | 0.77   | 0.61     |
|                   | SVM            | No         | 0.96      | 0.94   | 0.95     |
|                   |                 | Yes        | 0.37      | 0.45   | 0.40     |
|                   |                 | Macro-average | 0.66      | 0.70   | 0.67     |
|                   | Multi-layer Perceptron | No     | 0.95      | 0.98   | 0.97     |
|                   |                 | Yes        | 0.61      | 0.35   | 0.44     |
|                   |                 | Macro-average | 0.78      | 0.67   | 0.70     |
|                   | Random Forest  | No         | 0.93      | 1.0    | 0.96     |
|                   |                 | Yes        | 1.0       | 0.03   | 0.06     |
|                   |                 | Macro-average | 0.97      | 0.52   | 0.51     |
|                   | Logistic Regression | No        | 0.98      | 0.82   | 0.89     |
|                   |                 | Yes        | 0.25      | 0.80   | 0.38     |
|                   |                 | Macro-average | 0.62      | 0.81   | 0.64     |
| PV-DBOW-HS        | SVM            | No         | 0.96      | 0.96   | 0.96     |
|                   |                 | Yes        | 0.49      | 0.54   | 0.51     |
|                   |                 | Macro-average | 0.72      | 0.75   | 0.73     |
|                   | Multi-layer Perceptron | No     | 0.96      | 0.99   | 0.97     |
|                   |                 | Yes        | 0.68      | 0.39   | 0.49     |
|                   |                 | Macro-average | 0.82      | 0.69   | 0.73     |
|                   | Random Forest  | No         | 0.93      | 1.00   | 0.96     |
|                   |                 | Yes        | 0.92      | 0.04   | 0.08     |
|                   |                 | Macro-average | 0.92      | 0.52   | 0.52     |
|                   | Logistic Regression | No        | 0.98      | 0.86   | 0.92     |
|                   |                 | Yes        | 0.31      | 0.79   | 0.44     |
| SBERT             |                 | Macro-average | 0.65      | 0.82   | 0.68     |
|                   | SVM            | No         | 0.97      | 0.97   | 0.97     |
|                   |                 | Yes        | 0.57      | 0.56   | 0.57     |
|                   |                 | Macro-average | 0.77      | 0.77   | 0.77     |
|                   | Multi-layer Perceptron | No     | 0.95      | 0.99   | 0.97     |
|                   |                 | Yes        | 0.72      | 0.35   | 0.46     |
|                   |                 | Macro-average | 0.84      | 0.67   | 0.72     |
|                   | Random Forest  | No         | 0.94      | 1.00   | 0.97     |
|                   |                 | Yes        | 0.82      | 0.23   | 0.35     |
|                   |                 | Macro-average | 0.88      | 0.61   | 0.66     |
Table 8: Comparing Self-Attention based Classifiers (SelfAttention and GuidedSelfAttention) for ParContainsFrame in terms of Precision, Recall and F1-score for each class (Yes and No) as well as macro-average across the classes.

| Classifier       | Class | Precision | Recall | F1-score |
|------------------|-------|-----------|--------|----------|
| **SelfAttention**| No    | 0.98      | 0.99   | 0.98     |
|                  | Yes   | **0.80**  | 0.67   | 0.72     |
|                  | Macro-average | **0.89**  | 0.83   | 0.85     |
| **GuidedSelfAttention** | No    | 0.99      | 0.98   | 0.98     |
|                  | Yes   | 0.76      | **0.87** | **0.81** |
|                  | Macro-average | 0.88      | **0.93** | **0.90** |