Cisco at AAAI-CAD21 shared task: Predicting Emphasis in Presentation Slides using Contextualized Embeddings

Sreyan Ghosh[1,3*], **Sonal Kumar**[2*], **Harsh Jalan**[3*], **Hemant Yadav**[3], **Rajiv Ratn Shah**[3]

[1]Cisco Systems, Bangalore, India
[2]Department of Computer Science and Engineering, CHRIST (Deemed to be University), Bangalore, India
[3]MIDAS, IIIT Delhi
{gsreyan, skbrahee}@gmail.com, harshjalan27@yahoo.com, {hemantya, rajivratn}@iiitd.ac.in

Abstract

This paper describes our proposed system for the AAAI-CAD21 shared task: Predicting Emphasis in Presentation Slides. In this specific task, given the contents of a slide we are asked to predict the degree of emphasis to be laid on each word in the slide. We propose 2 approaches to this problem including a BiLSTM-ELMo approach and a transformers based approach based on RoBERTa and XLNet architectures. We achieve a score of 0.518 on the evaluation leaderboard which ranks us 3rd and 0.543 on the post-evaluation leaderboard which ranks us 1st at the time of writing the paper.

Introduction

Emphasis Selection for written text in visual media from crowdsourced label distributions was first proposed by **?** and then by **?** in SemEval-2020 Task 10, Emphasis Selection for Written Text in Visual Media (**?**). AAAI-CAD21 shared task: Predicting Emphasis in Presentation Slides (**?**) builds on the same SemEval-2020 Task 10. Presentation slides have become quite common in workplace scenarios and researchers have previously developed resources that guide presenters on the aspects of overall style, color, and font sizes to ensure that the graphical representation of the slide creates an impact on the viewer's mind and the viewer can relate and understand the message that the presenter is trying to relay through the slide. This shared task aims at designing automated approaches to predict which word in the slide should be emphasized (making bold or italics) to improve the visual appeal of the slide. A pictorial example of what this shared task aims to achieve can be seen in Figure 1.

To solve this problem we treat this task as a sequence labelling problem. Given the contents of an entire slide $d = \{w_1, w_2, ..., w_n\}$ as the input text, we predict the emphasis probability for each word in the contents of the slide $e = \{e_1, e_2, ..., e_n\}$.

We mainly try two approaches to solve this problem. In our transformers approach, we experiment with two different transformer based model architectures, namely RoBERTa (**?**) and XLNet (**?**). Our choice of transformer architectures is inspired by the best performing architectures in SemEval-2020 Task 10, Emphasis selection for written text in visual media (**??**). Both these models were pre-trained on large amounts of unannotated data in an unsupervised manner. A particular token w in a slide is first passed through these transformer models to obtain embedding of each word in the form of vector representations, after which these vectors are passed through BiLSTM and fully connected layers for classification. We also keep the transformer part of the model trainable and fine-tune the weights on our downstream task of emphasis prediction.

Figure 1: The left slide is plain text. The right side shows the important emphasized words in the slide.

In our second approach, we use a BiLSTM + ELMo model inspired by the baseline paper (**?**). We modify the baseline model and use character embeddings together with pre-trained ELMo embeddings of each word and feed them to BiLSTM + Attention and fully connected layers for predicting emphasis scores. Additionally, we also concatenate some word-level features in the attention output before feeding it to the fully connected layers. A part of our modification is inspired by the team that stood 3rd on the SemEval-2020 Task 10 leaderboard (**?**).

Additionally, we employ 2 approaches to training all models , i.e, BCE or Binary Cross Entropy Loss to directly predict emphasis probabilities and also KL Divergence Loss (**?**) which uses Label Distribution Learning (LDL) (**?**) to learn the probabilities of both emphasis and non-emphasis as used in the baseline paper (**?**).

Literature Review

A lot of work in NLP has been done on keyphrase extraction in long texts from scientific articles or news (**??**). Key-

*These authors contributed equally to this work.

word detection mainly focuses on finding important nouns or noun phrases from the input text. Emphasis prediction on the other hand focuses on the automated emphasizing of words in the input text that increase the visual appeal of the text and makes it easier for the viewer of the text to understand the actual message trying to be relayed through it.

Word emphasis prediction has also been explored in spoken data using acoustic and prosodic features (**??**). Emphasis Selection for written text in visual media was first proposed by **?** and then by **?** as a SemEval-2020 Task. The baseline paper (**?**) uses end-to-end label distribution learning (LDL) to predict emphasis scores on short text. The model has an embedding layer which is either Glove (**?**) or ELMo (Peters et al., 2018) followed by BiLSTM + Attention and fully connected layers. They used Adobe Spark Dataset[1] for their experiments. Hereon this model will be referred to as the "Baseline" model in our paper.

Team ERNIE (**?**) from Semeval-2020 Task 10 who stood 1st on the leaderboard, investigated the performance of several transformer-based models including ERNIE 2.0, XLM-RoBERTa, RoBERTa, ALBERT together with a combination of pointwise regression and pairwise ranking loss. The authors also tried some augmentation schemes and word-level lexical features and reported ERNIE 2.0 with the addition of lexical features to be the best performing model on the shared-task dataset.

Team IITK (**?**) that stood 3rd on the leaderboard also explored a number of transformer-based datasets including variations of BERT, RoBERTa, XLNet, GPT-2 and XLNet and also a modification on the baseline model. Parts of our modification on the baseline model are also based on this model reported here. Their final results were obtained from a simple ensemble of a number of their transformer-based models.

Team MIDAS (**?**) which stood 11th on the leaderboard also used BERT, RoBERTa and XLNet together with a combination of either BiLSTM and Dense or just Dense layers.

Learning from annotations from different annotators has been explored with majority voting (**?**) or by learning individual annotator expertise (**???**). Most work on this takes only one label sequence as correct. The baseline paper (**?**) was the first work to have used Label Distribution Learning (**?**) for a sequence labeling task. We also explore this learning scheme in the experiments mentioned in our paper together with Binary Cross Entropy loss on probabilities obtained from the dataset annotations.

Background

Problem Definition

Given a sequence of words or tokens $d = \{w_1, w_2, ..., w_n\}$ in a slide, the task is to compute a probabilistic score $e_i \epsilon [0, 1]$ for each w_i in d which indicates the degree of emphasis to be laid on the word.

Evaluation Metric

The evaluation metric for our problem is defined as follows:

[1]https://spark.adobe.com/

For a given m (1,5 and 10), we first define 2 sets, $S_m^{(x)}$ - set of m words with top m probabilities according to ground truth and $\hat{S}_m^{(x)}$ - set of m words with top m according to the model predictions. To get $S_m^{(x)}$, each word in the sentence has been manually annotated by 8 annotators. Based on these 2 sets, we define $Match_m$ as:

$$\text{Match}_m = \frac{\sum_{x \in D_{\text{test}}} \left| S_m^{(x)} \cap \hat{S}_m^{(x)} \right| / min(m, |x|)}{|D_{\text{test}}|} \quad (1)$$

where D_{test} is the dataset and x is the token instance. We find $Match_m$ for m $\in \{1, 5, 10\}$ and express our final score as a simple average over all 3 of them.

Dataset

Dataset Statistics for the dataset provided in the AAAI-CAD21 shared task is shown in Table 1. Each training instance is a complete slide with all the tokens present in the slide. Additionally, the sentence-wise divisions in the slides are also provided in the data. The entire training dataset was annotated by a total of 8 annotators on token-level emphasis. The dataset was annotated with a BIO tagging scheme where each annotator either annotated the token as an emphasized token $(B \, or \, I)$ or not (O). Thus, the probability of emphasis for each token was calculated as an average score of all annotations. The annotation scheme and the emphasis probability calculation has been shown with an example in Table 3. More information about the task and data creation can be found in **?**.

	Total Slides	**Total Sentences**	**Total Tokens**
Train	1241	8849	96934
Dev	180	1175	12822
Test	355	2569	28108

Table 1: Train, Development and Test Dataset Description

	Min	**Max**	**Average**
Train	13	180	78
Dev	15	164	71
Test	17	181	79

Table 2: Token length description

System Description

Token Level Features

We tried investigating the data to find token level features that can enhance the performance of our BiLSTM-ELMo model. We tried finding features by analyzing a particular feature's average emphasis score and the number of times a word with that feature occurred in our dataset. The average emphasis scores of the token with these features and the total count can be found in Table 4. Initially, we tried only shape and syntactic features of words by concatenating them with

Words	A1	A2	A3	A4	A5	A6	A7	A8	Freq.[B+I,O]	Probs.
	O	O	O	O	O	O	O	O	[0,8]	0.0
Have	O	O	O	O	O	O	O	O	[0,8]	0.0
population	O	B	O	O	B	O	O	O	[2,6]	0.25
counts	O	O	O	O	O	O	O	O	[0,8]	0.0
for	O	O	O	O	O	O	O	O	[0,8]	0.0
three	O	B	O	O	O	O	O	B	[2,6]	0.25
key	O	I	O	O	O	O	O	I	[2,6]	0.25
species	O	I	B	O	B	O	O	I	[4,4]	0.5

Table 3: Annotation scheme and emphasis probability calculation on a sample sentence from the Train dataset.

the attention output as described in our system description. The only feature that had given us an improvement over the baseline score was POS (Parts of Speech) tags.

Figure 2: Word Cloud of tokens having emphasis probability of ≥ 0.5

.

Type	Train (Avg/Nos.)	Dev (Avg/Nos.)
Punctuation	0.031/14726	0.034/2082
UpperCase start	0.136/21195	0.157
Contain numbers	0.045/2893	0.055/308
All Upper Case	0.092/4498	0.116/523
Inside Brackets	0.002/3598	0.009/7560
Keyphrase Tags	**0.25/12723**	**0.35/1179**
Overall	0.102/96934	0119/12822

Table 4: Average Emphasis Scores and Count

Upon analysis of words with an emphasis score ≥ 0.5 we noticed that most of them were scientific keywords. Thus we created our own feature by training a sequence labeling BiLSTM-CRF model with BERT (**?**) word embeddings as input to the model with the information extraction datasets used for scientific keyphrase extraction by **?** . We use the python flair[2] library for this task. The results of the model trained for this task are given in Table 5. The model was trained and inferred with a BIO tagging scheme and it was processed to a binary feature where B and I tags were termed as 1 and O as 0. This feature when used together with POS tags gives us a decent improvement on the baseline results. We name this feature the Keyphrase Feature in all our experiments.

	Precision	Recall	f1-score	Support
B	0.6359	0.5510	0.5904	5294
I	0.6413	0.6572	0.6492	6561
O	0.9313	0.9395	0.9354	61941
Macro avg	0.7362	0.7159	0.7250	73796
Weighted avg	0.8844	0.8866	0.8852	73796

Table 5: Keyphrase Extraction Model Results

Our Approach

BiLSTM-ELMo Approach Our BiLSTM-ELMo approach is inspired by the baseline paper (**?**) where we extract the ELMo embeddings (**?**) E_{ELMo} for each word in a sequence and additionally, we pass the input through a character-level BiLSTM Network where the combined forward and backward embedding for the last character of each word is then passed through a Highway Layer (**?**) which effectively provides us with contextual word-level embeddings E_{hway} for our entire sequence. These contextual word-level embeddings are then concatenated with the extracted ELMo Embeddings for each word to produce the final word embeddings E.

We pass E through a BiLSTM Layer followed by an Attention Layer. The output of the attention layer is then concatenated with the POS tags (**?**) and Keyphrase Feature. for the corresponding word at each time-step. Now combined, the attention output and the external features are fed to a Time Distributed Dense Layer followed by our Time Distributed Output Layer. The activation function g of the Output Layer is either Sigmoid or Softmax depending on whether the Loss Criterion is Binary Cross-Entropy Loss (in case of Sigmoid Activation) or Kullback-Liebler Divergence Loss (in case of Softmax Activation) used for Label Distribution Learning (LDL).

[2]https://github.com/flairNLP/flair

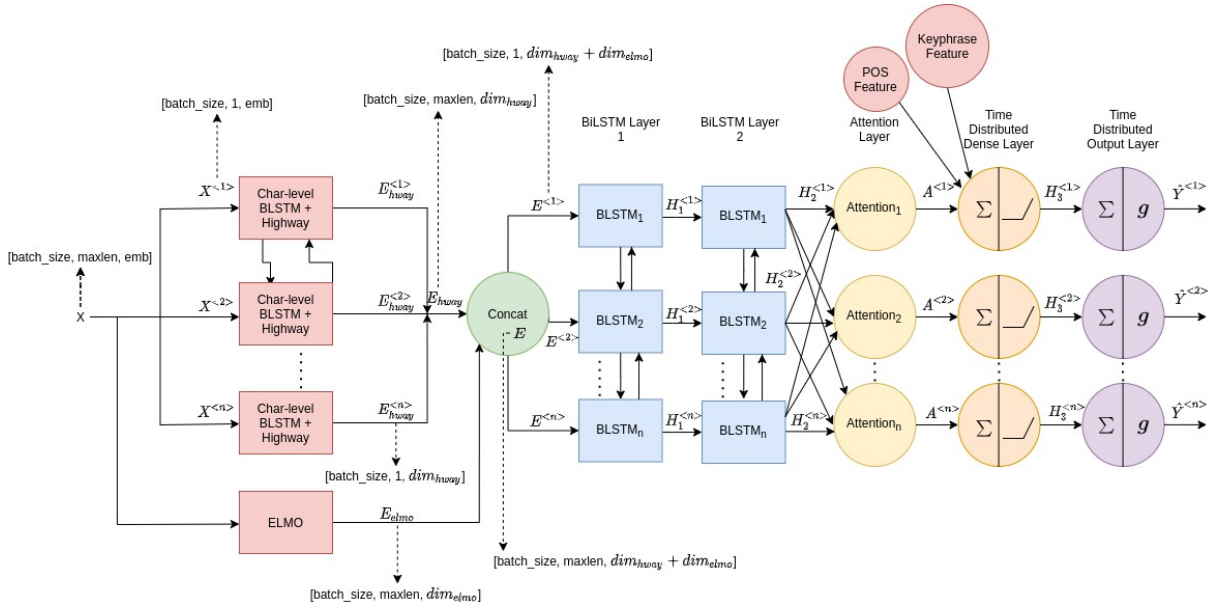

Figure 3: The BiLSTM-ELMo Model

Transformers Approach Our Transformers approach makes use of one of two Transformer Architectures that is, XLNet or RoBERTa. First, the tokenized word input is passed through the transformer architecture and the outputs of all encoding layers of the transformer are concatenated together to get the final embedding E for any given word. This embedding E is now fed through a BiLSTM Layer followed by a set of Time Distributed Dense Layer. Finally, the output of the Time Distributed Dense Layers are fed to our Time Distributed Output Layer with Activation Function g. Here, g can either be Sigmoid or Softmax when Loss Function is Binary Cross-Entropy or KL-Divergence respectively.

Experimental Setup

We use PyTorch [3] Framework for our Deep Learning models along with the Transformer implementations, pre-trained models and, specific tokenizers in the HuggingFace library[4].

In the BiLSTM-ELMo Approach, we use a hidden size of 300 for the character-level LSTM Layers and on top of that, we use one highway layer which gives us word-level embeddings E_{hway}. These embeddings are then concatenated with their corresponding ELMo embeddings E_{ELMo} where the embeddings have 2048 dimensions. This concatenated vector is passed through a BiLSTM Layer with an output size of 512 dimensions in each direction. The Attention Layer uses a self-attention mechanism, the output of the attention mechanism is concatenated with the POS tags and Keyword Feature for each word so that this information can be used by the classifier to make better predictions. The final stage of the classifier consists of a Time Distributed Dense Layer with a hidden size of 20 and ReLU Activation. Finally, the output layer has 1 output neuron if the activation function is Sigmoid and the loss function is Binary Cross-Entropy and 2 output neurons in case of a Softmax activation function and KL-Divergence Loss Function. The dropout layer probabilities were set to 0.3 for all layers to avoid overfitting.

In the transformers approach, we used the RoBERTa and XLNet transformers without freezing any layers of the network and the output of all encoder layers are concatenated to make word-level embeddings E. These word embeddings are then passed through two BiLSTM Layers with an output size of 256 dimensions in each direction. The output is then fed to a pair of BiLSTM Layers with 256-dimensional output in both directions. In the classifier, the output of the BiLSTM is fed to a pair of Time Distributed Dense Layers with a hidden layer size of 20 and ReLU Activation and finally to the Time Distributed Output Layer which has either 1 or 2 output neurons depending on whether the activation function used is Sigmoid or Softmax respectively. Dropout Layers with Dropout Probability 0.3 are also added to prevent overfitting.

When using Sigmoid activation, we aim to predict a single output $\hat{y}^{<t>}$ which represents the probability of emphasis to be laid on the t^{th} token. This probability is used with the Binary Cross-Entropy Loss to train the model. However, in the case of Softmax, we predict a probability distribution $\hat{Y}^{<t>}$ over 2 classes $\{0 = no\ emphasis, 1 = emphasis\}$. This distribution is used with the KL-Divergence Loss function to perform Label Distribution Learning. The equations for $\hat{y}^{<t>}$ and $\hat{Y}^{<t>}$ are as follows:

$$\hat{y}^{<t>} = Sigmoid(z^{<t>}) \tag{2}$$

Where $z^{<t>}$ is the logit of the last output layer for the t^{th} token.

[3] https://pytorch.org/

[4] https://huggingface.co/transformers/

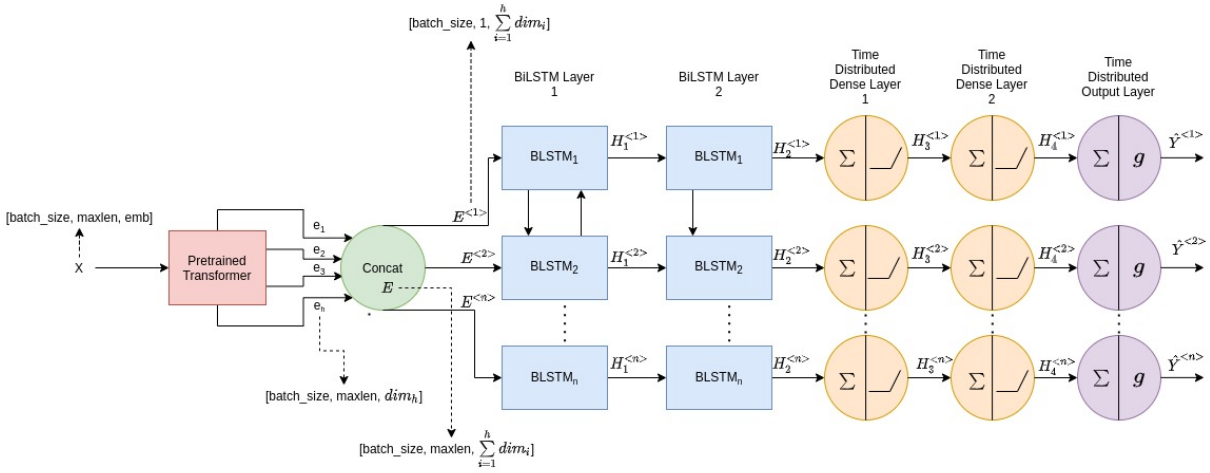

Figure 4: The Transformer-Based Model

$$\hat{Y}^{<t>} = \{Softmax(z_0^{<t>}), Softmax(z_1^{<t>})\} \quad (3)$$

Where $z_i^{<t>}$ is the logit of the last output layer for the t^{th} token and i^{th} class.

In both the Transformers and BiLSTM-ELMo approaches, the Binary Cross-Entropy (BCE) Loss as well as the KL-Divergence (KLD) Loss were used to train the models. The $Match_m$ score is used as an evaluation for all our models. The equations for both the loss functions are as follows:

$$BCE(y^{<t>}, \hat{y}^{<t>}) = -y^{<t>}.\log(\hat{y}^{<t>}) - (1 - y^{<t>}).\log(1 - \hat{y}^{<t>}) \quad (4)$$

Where $y^{<t>} \in \{0, 1\}$ is the true label for emphasis laid on each token and $\hat{y}^{<t>}$ is the output of the sigmoid activation for each token.

$$KLD(Y^{<t>}||\hat{Y}^{<t>}) = \sum Y^{<t>}.\log\left(\frac{Y^{<t>}}{\hat{Y}^{<t>}}\right) \quad (5)$$

Where $Y^{<t>}$ is the true probability distribution for the emphasis laid on each token and $\hat{Y}^{<t>}$ is the output distribution of the softmax activation for each token.

We use the Adam Optimizer for training the models with a learning rate of 1e-4 for the BiLSTM-ELMo model for 100 epochs and 2e-5 for the Transformer-based models for 100 epochs. The training was performed on 1 NVIDIA Titan X GPU. Our code is available on Github[5].

Results

In Table 6 we present scores for both our BiLSTM-ELMo and Transformers approach trained on both BCE Loss and KLDivergence Loss for LDL. As we can see in the results, LDL as used by (Shirani et. al 2019) doesn't give a huge improvement over results and at times even diminishes the results.

Model	Dev	Test
BiLSTM-ELMo (Baseline)	-	0.475
BiLSTM-ELMo (POS)	0.497	0.484
BiLSTM-ELMo (POS) (LDL)	0.501	0.506
BiLSTM-ELMo (POS, Keyphrase)	0.515	0.496
BiLSTM-ELMo (POS, Keyphrase) (LDL)	0.504	0.501
XLNet	**0.536**	**0.514**
XLNet (LDL)	0.529	0.491
RoBERTa	0.51	0.485
RoBERTa (LDL)	0.515	0.47

Table 6: Performance of BiLSTM-ELMo and Transformers approach on development and Test set. The results are expressed in terms of average $Match_m$ for m $\in \{1, 5, 10\}$. LDL indicates that label distribution learning was employed to train the model with KL-Divergence as the loss function, Binary Cross Entropy otherwise. For BiLSTM-ELMo model the extra features concatenated at the attention layer have been mentioned with each experiment. Baseline. indicates the scores by the baseline model defined by **?**

For our final submissions, we tried an ensemble of scores from different models shown in Table 7. Our best scores on the Evaluation leaderboard were obtained using an ensemble of XLNet and RoBERTa with LDL where we stood 3rd. Meanwhile, our best scores on the Post-Evaluation leaderboard were obtained using an ensemble of XLNet and BiLSTM-ELMo approach with POS tags and Keyphrase Feature where we currently stand 1st on the leaderboard.

Additionally, we also ran experiments by dividing the presentations into their constituent sentences in the train and development data. Thus each training instance now corre-

[5]https://github.com/realsonalkumar/CAD21-AAAI21

Model	Dev	Test
XLNet + RoBERTa (LDL)	0.547	0.518
XLNet + BiLSTM-ELMo (Keyphrase)	0.538	0.532
XLNet + BiLSTM-ELMo (LDL)	**0.55**	**0.543**

Table 7: Performance of different ensemble models

sponds to a particular sentence belonging to a presentation slide in the original corpus. The development set results can be found in Table 8. The evaluation scheme used in this experiment uses the same Match$_m$ as described in the Evaluation Metric section but with m = $1, 2, 3, 4$ as used in **?**.

Model	Dev
XLNet	**0.758**
XLNet (LDL)	0.757
RoBERTa	0.743
RoBERTa (LDL)	0.745
BiLSTM-ELMo	0.751
BiLSTM-ELMo (LDL)	0.752

Table 8: Sentence-wise results on the Development set

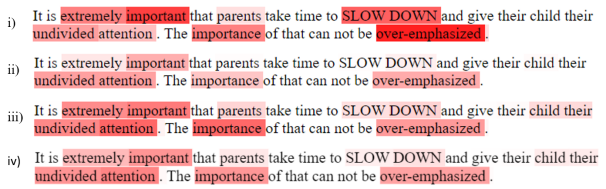

Figure 5: Emphasis Heatmaps i) Ground Truth ii) BiLSTM-ELMo iii) XLNet iv) Best Ensemble Model

Analysis

Length vs Performance

We wanted to understand how the performance of our models was affected by the length of the instances. Table 9 summarizes the performance of our best performing single model, i.e, XLNet on the development set divided into three sets, Short ($\leq$ 40 tokens, 80 samples), Medium (40 to 90 tokens, 262 samples), and Long (>90 tokens, 50 samples). As we can see, the model performance deteriorates with the increasing length of the instances.

	XLNet
Small ($\leq$ 40)	**0.648**
Medium (>40 and $\leq$ 90)	0.549
Large (>90)	0.42

Table 9: Average $Match_m$ for best performing XLNet model on different size of instances in the development set

Emphasis vs Parts of Speech

Table 10 shows POS (Parts of Speech) tags vs. average emphasis on the development dataset. We did this experiment to understand how our model predictions performed on each POS tag when compared to the actual human-annotated emphasis scores on the development set. We noticed that the original average emphasis scores were highest on Adjectives followed by Noun. On comparing our models, we found that XLNet was able to almost accurately predict the emphasis scores on Adjectives and Noun respectively, and BiLSTM-ELMo also had the highest predictions on Adjectives and Noun respectively. We also noticed that XLNet did a better job on predicting the emphasis score on different POS tags where the predictions were either very close to the human scores or marginally lesser. On the other hand, we noticed that BiLSTM-ELMo's predictions fell short by bigger margins when compared to XLNet and gave more emphasis to Adverbs than that in the development set.

POS	Count	Human	BiLSTM	XLNet
Noun	4719	0.169	0.134	0.168
Verb	1420	0.118	0.083	0.113
Adjectives	982	0.186	0.140	0.181
Det	634	0.062	0.029	0.042
Adverbs	347	0.111	0.068	0.103
Pronouns	165	0.040	0.068	0.022
Punct	2082	0.034	0.015	0.025

Table 10: POS tags vs. average emphasis on development dataset

Conclusion

In this paper, we present our approach to AAAI-CAD21 shared task: Predicting Emphasis in Presentation Slides. Our best submission gave us an average $Match_m$ of 0.518 placing us 3rd on the Evaluation phase leaderboard and an average $Match_m$ of 0.543 placing us 1st on the Post-Evaluation leaderboard at the time of writing the paper. Future work includes using a hierarchical approach to emphasis prediction as a sequence labeling task using both sentence-level (individual sentence in a slide) and slide-level representations of a word (**?**).

Acknowledgement

Rajiv Ratn Shah is partly supported by the Infosys Center for AI at IIIT Delhi. We also thank Sunny Dsouza and Gautam Maurya for their detailed and valuable feedback.

Neque tenetur asperiores repellendus maiores aspernatur molestiae placeat, distinctio quod tempore ab sit eos id repellat enim quis soluta tempora, quisquam praesentium similique ea error, voluptate blanditiis harum architecto aliquid recusandae omnis, nihil eum culpa dicta ut repudiandae facere dolorum obcaecati asperiores sequi est?Quod sapiente laboriosam libero maiores perspiciatis aliquam, omnis dolorem excepturi, esse possimus debitis dolore animi laudantium quaerat dicta excepturi mollitia nam ducimus?Porro minus maxime nisi consequatur ab necessitatibus nesciunt, odit harum commodi, laudantium quas tenetur inventore beatae aperiam laboriosam, quaerat dolor quam modi, quidem porro sequi aliquam