BiLSTMs and BERT for Semantic Role Labeling

Hossam Arafat
Sapienza University of Rome
arafat.1803850@studenti.uniroma1.it

1 Introduction

Semantic Role Labelling (SRL) is the task of extracting a structured representation of a given sentence's meaning. SRL systems achieve this by identifying the predicate-argument relationships that exist within a sentence. As such, SRL is typically broken down into four sub-tasks: predicate identification, predicate sense disambiguation, argument identification, and argument classification. There are two methods for annotating arguments: span-based and dependency-based.

Simply put, SRL can be viewed as the task of taking a sentence as input and producing a representation that answers the question of *"who did what to whom (when and where) ?"*. This explicit representation of semantic information as produced by SRL systems has been shown to improve results in popular downstream NLP tasks such as dialog systems (Tur et al. 2005; Chen et al. 2013), machine translation (Shi et al., 2016), and question answering (Berant et al. 2013; Yih et al. 2016).

This work presents models that achieve near state-of-the-art performance on span-based argument identification and classification. Gold, disambiguated predicates are supplied as input to all our models. We frame SRL as a sequence learning problem where for every input $\overrightarrow{x} = (x_1, ..., x_T)$, our model should predict a sequence of output symbols $\overrightarrow{y} = (y_1, ..., y_T)$. Where every input x_i is a lemma drawn from an input vocabulary V and y_i is either the argument's relation with the predicate (called semantic role and specified by a semantic role inventory), if y_i has been identified as an argument, and the null-tag "_" if y_i has not.

2 VerbAtlas

In order to perform argument classification (and have gold disambiguated predicates), a verb inventory with clearly defined semantic roles is needed. Many such inventories exist such as Propbank (Palmer et al., 2005), FrameNet (Baker et al., 1998), and VerbAtlas [1] (Di Fabio et al., 2019). Since detailing the pros and cons of each is not the motivation behind this work, we point the reader to (Di Fabio et al., 2019) which discusses that topic in length. Briefly, this work chose to use VerbAtlas, as it was manually-crafted and contains a high coverage of verbs in English. Moreover, it defines the semantic roles in a human-readable, clearly-defined, and well-specified manner.

3 Bidirectional LSTMs

Bidirectional Long Short-term Memory Networks (Graves and Schmidhuber, 2005, BiLSTMs) have shown tremendous success in sequence learning problems in a variety of NLP tasks from Word Segmentation (Ma et al., 2018) to Word Sense Disambiguation (Raganato et al., 2017, WSD).
The overwhelming majority of the recent work pushing the state-of-the-art in SRL through syntax-agnostic or syntax-aware approaches utilized a BiLSTM-based model in order to do so (He et al. 2017; Li et al. 2018; Cai and Lapata 2019). Following in the footsteps of such work, this work elected to use a BiLSTM-based model as the baseline (Section 3) and we study the effects of augmenting it with various components (Sections 5 to 7).

BiLSTMs are particularly useful for SRL as clues for identifying whether or not a word has a semantic role with a verb, and classifying it, can be present after the word in context. For example, in order to identify that "*a stake*" has a semantic relation of "THEME" (as in the "object being held") in the sentence "*A stake* **held** *by my father's company.*", it is important to know the context occurring after it.

[1] https://www.verbatlas.org/.

4 Designing the Predicate Indicator

English sentences often contain $n > 1$ verbs. To classify the different arguments of each predicate (verb) separately, we simply feed the sentence n times into the network and provide the corresponding label for the current predicate under consideration each time.

In order to inform our SRL system with the location of all predicates that occur in a given input sentence, we designed a simple "binary flag", $\overrightarrow{b}$. Where for every sentence $\overrightarrow{x} = (x_1, ..., x_T)$, $\overrightarrow{b}$ has the same sequence length as $\overrightarrow{x}$ and b_i is 1 if x_i is a predicate and 0 if x_i is not. This binary flag is concatenated to the lemma embeddings of a given input sentence.

Concatenating a simple binary flag to the representation of each input sentence fails to supply the model with a key insight: The identification and classification of the semantic role that a given argument has with the predicate often depends on the sense that the predicate occurs in. Let us consider the two sentences: "*I* **ran** *this company.*" and "*I* **ran** *for 10 miles.*". In the first sentence, the verb "run" occurs in the sense "**to operate**" and as a result, "*I*" is classified as an "AGENT" (the individual that operates). While in the second sentence, "*I*" is classified as a "THEME" (the entity that runs) as a result of the verb "run" occurring in the sense "**to run / jog**".

To convey this key insight to the model, a "predicate embedding" indicator $\overrightarrow{p}$ was created. Where first, a vector $\overrightarrow{s}$ is created which has the same sequence length as $\overrightarrow{x}$ and s_i is equal to the specific sense the predicate occurs in if x_i is a predicate and 0 if x_i is not. $\overrightarrow{s}$ is then fed through a "predicate embedding layer" which produces a trainable embedding, $\overrightarrow{p}$, that is concatenated to the lemma embeddings of a given input sentence. Either $\overrightarrow{b}$ or $\overrightarrow{p}$ are used at any given time, as shown in Figures 1 and 3.

5 Incorporating Part-of-Speech Tags

Part-of-Speech (POS) tagging often improves a variety of downstream NLP tasks. SRL is one such task for the following reasons: Firstly, predicates will always have some kind of "verb" POS Tag. As such, POS Tags act as an "auxiliary predicate indictor" by giving predicates one of the "VB" POS tags listed in Table 1, and giving non-predicates other tags. Secondly, some of the VerbAtlas argument classes have very distinct POS Tags, as shown in Figure 2. This helps the network learn that, for example, words that are typically classified as an "AGENT" are often nouns. Other classes practically have one, unique POS Tag associated with them, like how words of the "NEGATION" class are always adverbs or how words of the "LOCATION" class are often prepositions.

6 Utilizing Pretrained embeddings

Pretrained embeddings allow the model to more quickly and reliably pick up semantic signals that are helpful for classifying each token than when training our own embedding layer from scratch. Consequently, the uncased version of GloVe (Pennington et al., 2014) pretrained embeddings was used.

7 Utilizing Contextualized Embeddings

Training our own embedding layer or even using pretrained embeddings as explained in Section 6 generates word embeddings that are independent of the context that occurs around a given word. For example, the GloVe word embedding for the word "bank" is the exact same in the sentence "*I will go to the bank tomorrow*" and "*I will have a picnic by the river bank tomorrow*" despite "*bank*" occurring with a different sense each time. Contextualized embeddings are ones that are a function of a given word's context. A famous model that generates such embeddings is BERT (Devlin et al., 2018).

BERT was pre-trained on the task of Next Sentence Prediction (NSP), where it receives two sentences as input, A and B, separated by a special separator token "[SEP]" and outputs a binary decision on whether or not sentence B is the next sentence in the corpus after A. This was especially advantageous in SRL when handling sentences with $n > 1$ predicates as it allowed us to do the following: feed the same sentence n times to BERT, but each time, insert a different predicate after the "[SEP]" token. This allows BERT, through its self-attention mechanisms, to produce contextualized embeddings for every lemma in that sentence that are different for each predicate we insert after the separator token, as shown in Figure 4. This helps the network better differentiate and classify the specific arguments of each predicate in the same sentence.

8 Experimental Setup

This section details all the experiments we conducted in our effort to study the effect of combining all the aforementioned concepts and intuitions.

8.1 Dataset

We use a proprietary dataset of English sentences that includes gold POS Tag, lemma, and predicate information. The data was split into training, development, and testing sets as shown in Table 2. We report the precision (P), recall (R), and F_1-score all our models obtained on that testing dataset in Table 5. We implemented no special preprocessing strategies.

8.2 Baseline Model

Our baseline model, shown in Figure 1, starts with a lemma embedding layer that given an input sentence $\overrightarrow{x} = (x_1, ..., x_T)$ returns an embedding vector that represents each lemma $\overrightarrow{w} = (w_1, ..., w_T)$. For this model, we test the effect of using either the "binary predicate" (Experiment 1) or the "predicate embedding" indicators (Experiment 2) described in Section 4. We then study how the model improves when we provide the POS Tags for each word and concatenate a POS embedding to the lemma and predicate indicator embeddings (Experiment 3). Moreover, we investigate the effect of using GloVe embeddings (Experiments 4 and & 5). We pass this concatenated embedding vector to a 2 layer BiLSTM with 256 hidden units. More hyper-parameters can be found in Table 3.

8.3 BERT-based Model

We utilize the power of BERT -specifically the `bert-base-uncased` version- as described in Section 7, by feeding the input sentence n times, once for each predicate x_p, in the following form: $\{$[CLS] $\overrightarrow{x} = (x_1, ..., x_T)$ [SEP] x_p [SEP]$\}$.

We then extract the 768-dimensional contextualized embeddings of every lemma in the sentence, dropping the embeddings of $\{$[CLS] [SEP] x_p [SEP]$\}$. This forms our vector $\overrightarrow{w} = (w_1, ..., w_T)$. As done in Section 8.2, we test the effect of using the "predicate embedding", and the POS embedding of each word (Experiments 6 & 7). We feed the vector $\overrightarrow{w}$ -concatenated with the various vectors we test- into a BiLSTM with 256 hidden units. More hyper-parameters can be found in Table 4. Finally, we observe if gradient clipping will improve the model's performance (Experiment 8). All the experiments can be found in Table 5. This model is shown in Figure 3. The loss plot for all experiments is shown in Figure 5.

9 Results

9.1 Baseline Model

Experiments 1 & 2 Using the "predicate embedding" indicator instead of the "binary indicator" clearly improves the model's performance when it comes to the classification sub-task as the precision (P), recall (R), and F_1-score all improve by 6 points or more. This confirms the intuition we had in Section 4 where we stated that in order to better classify a predicate's arguments, it is crucial to know the sense that the predicate occurs in. All three metrics improve in the identification sub-task as well.

Experiments 3 POS Tags improve the model's performance as we see all three metrics in both the identification and classification sub-tasks increase by 2%. Again, this confirms the reasoning we presented in Section 5 concerning all the additional information POS Tags provide to the model.

Experiments 4 & 5 Pretrained GloVe embeddings improved the performance of the model by a smaller amount than we anticipated. Providing an average increase of 1 point across all metrics compared to when we train our own embedding layer.

9.2 BERT-based Model

Experiment 6 Contextualized embeddings provided by BERT gave us the biggest boost in our performance, vastly improving the model's precision, recall and pushing the F_1-score above the 90 and 85 percent threshold in each task. Justifying the reason we used them (explained in detail in Section 7 and demonstrated in Figure 4).

Experiment 7 Similar to their effect on the performance of the baseline model, POS Tag information improves the precision of the BERT-based model by 2 points and F_1-score by 1 point, in both sub-tasks, respectively.

Experiment 8 Gradient clipping mitigates the problem of exploding gradients LSTMs usually face. A gradient clipping value of 3.0 gave us the model with the best F1 score of 93.6 and 88.0 in argument identification and classification respectively.

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Rui Cai and Mirella Lapata. 2019. Syntax-aware semantic role labeling without parsing. *Transactions of the Association for Computational Linguistics*, 7:343–356.

Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky. 2013. Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 120–125. IEEE.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Andrea Di Fabio, Simone Conia, and Roberto Navigli. 2019. Verbatlas: a novel large-scale verbal semantic resource and its application to semantic role labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 627–637.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483.

Zuchao Li, Shexia He, Jiaxun Cai, Zhuosheng Zhang, Hai Zhao, Gongshen Liu, Linlin Li, and Luo Si. 2018. A unified syntax-aware framework for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2401–2411.

Ji Ma, Kuzman Ganchev, and David Weiss. 2018. State-of-the-art chinese word segmentation with bilstms. *arXiv preprint arXiv:1808.06511*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167.

Chen Shi, Shujie Liu, Shuo Ren, Shi Feng, Mu Li, Ming Zhou, Xu Sun, and Houfeng Wang. 2016. Knowledge-based semantic embedding for machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2245–2254.

Gokhan Tur, Dilek Hakkani-Tur, and Ananlada Chotimongkol. 2005. Semi-supervised learning for spoken language understanding semantic role labeling. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, pages 232–237. IEEE.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

A Tables

POS Tage	Details	Example
VB	base form	*take*
VBD	past tense	*took*
VBG	gerund / present participle	*taking*
VBN	past participle	*taken*
VBP	singular present, non-3rd person	*take*
VBZ	3rd person singular present	*takes*

Table 1: Type of "verb" POS Tags in the dataset we used.

Dataset	Number of Sentences
Training	39.2K
Development	1.30K
Testing	2.00K

Table 2: Train-dev-test split employed in all experiments

learning rate	batch size	dropout	embedding dim.	layers	hidden dim.	epochs
1x10^{-3}	256	0.4	300	2	256	20

Table 3: Hyper-parameters used throughout all experiments that test our $Baseline$ model. The embedding dimension is the same for the lemma, predicate, and POS embeddings (which are concatenated together to obtain our final embeddings). We train the Baeline Model for 20 epochs and early stopping halts training at a different point in every experiment.

learning rate	batch size	dropout	embedding dim	layers	hidden dim.	epochs (ES)
5x10^{-5}	32	0.4	300	2	256	3

Table 4: Hyper-parameters used throughout all experiments that test our $BERT$ model. The embedding dimension is the same for the predicate and POS embeddings (which are concatenated together to obtain our final embeddings). The model is able to produce excellent results in 3 epochs only thanks to two things: all the improvements we added and the fact that BERT-based models famously require very little fine-tuning in order to produce good results.

Experiment	Architecture	Identification			Classification		
		P	R	F1	P	R	F1
1	$Baseline^{binary}$	85.5	86.6	86.1	74.1	75.1	74.6
2	$Baseline^{embed}$	88.0	87.0	87.4	82.2	81.1	81.6
3	$Baseline^{embed}$ + POS	90.1	89.5	89.8	84.3	83.6	83.9
4	$Baseline^{embed}$ + GloVe	89.6	86.6	88.1	83.9	81.1	83.0
5	$Baseline^{embed}$ + POS + GloVe	90.4	90.6	90.0	84.0	84.2	84.4
6	$BERT^{embed}$	91.5	91.9	91.2	84.0	85.5	85.7
7	$BERT^{embed}$ + POS	93.1	91.6	93.0	87.2	85.4	86.8
8	$BERT^{embed}$ + POS + clip	**93.5**	**94.1**	**93.6**	**88.0**	**88.4**	**88.0**

Table 5: Precision (P), recall (R), and F_1-scores (%) obtained in each experiment. The "^{binary}" superscript means the "binary predicate" indicator, $\overrightarrow{b}$, was used. While the "^{embed}" superscript means that the "predicate embedding" indicator, $\overrightarrow{p}$, was used. "clip" stands for gradient clipping. All models were trained using early stopping (ES), where training is stopped if the validation loss increases.

B Figures

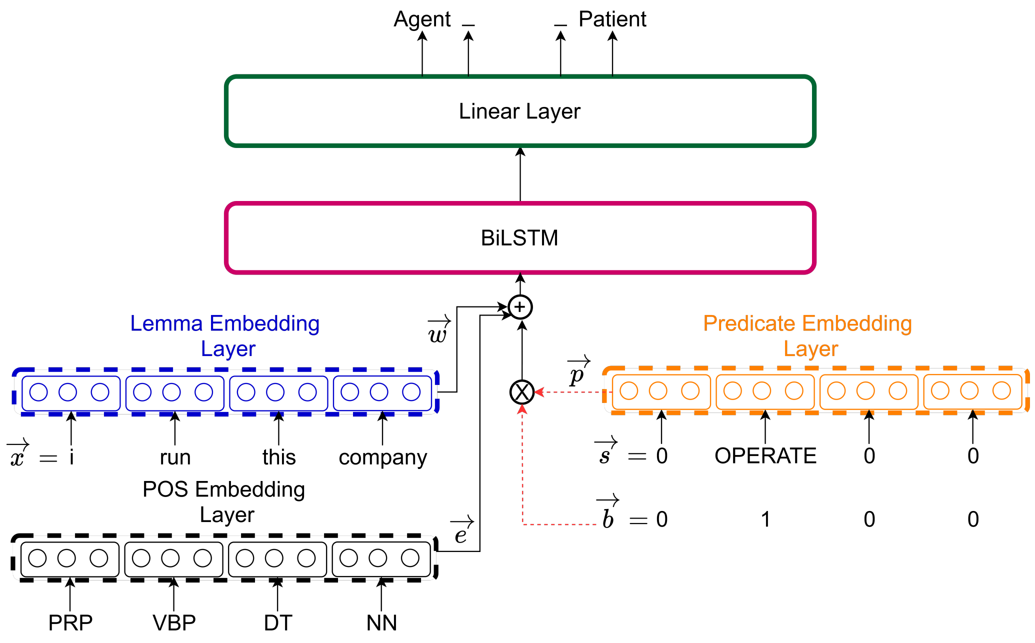


Figure 1: Overview of our Baseline model's architecture. The model has a lemma embedding layer that takes as input a sentence of lemmas $\overrightarrow{x}$ and produces an embedding for each lemma $\overrightarrow{w}$. The model also receives the POS Tag of each word and produces a representation for them, $\overrightarrow{e}$, via a POS embedding layer. The $\bigotimes$ symbol represents the XOR operation as the model receives either the binary predicate indicator, $\overrightarrow{b}$, or the predicate embedding indicator, $\overrightarrow{p}$. The $\bigoplus$ symbol represents the concatenation operation where the vectors $\overrightarrow{w}$, $\overrightarrow{e}$, and either predicate indicator $\overrightarrow{b}$ or $\overrightarrow{p}$ are concatenated before being fed into a BiLSTM. The output of which is fed to a linear layer that outputs the network's classification.

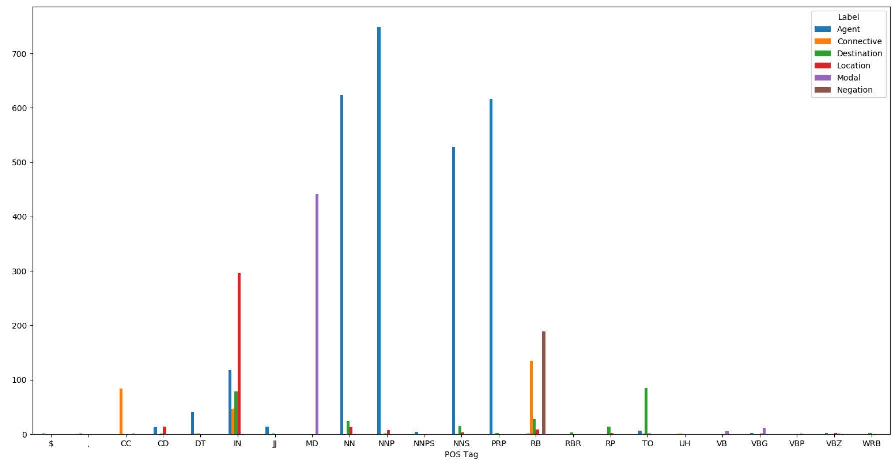


Figure 2: The frequency of occurrence of each POS Tag for the ground truth VerbAtlas argument classes. We can see that: lemmas with the "Agent" class are predominantly nouns (NNS, NNP, NNS) and personal pronouns (PRP). This makes sense, as in the English language, entities that take actions are often people like "*I* or *Roberto*". Lemmas with the "Destination" class are often the word "to" (POS Tag "TO") as in *"I go to ..."*. Lemmas of the "Location" class are almost exclusively pronouns (IN) like "in" and "at". The "Modal" and "Connective" classes have the modal and connective conjuction POS Tags (MD) and (CC) respectively. And the "Negation" class which VerbAtlas assigns to the word "not" always has the adverb POS Tag (RB).

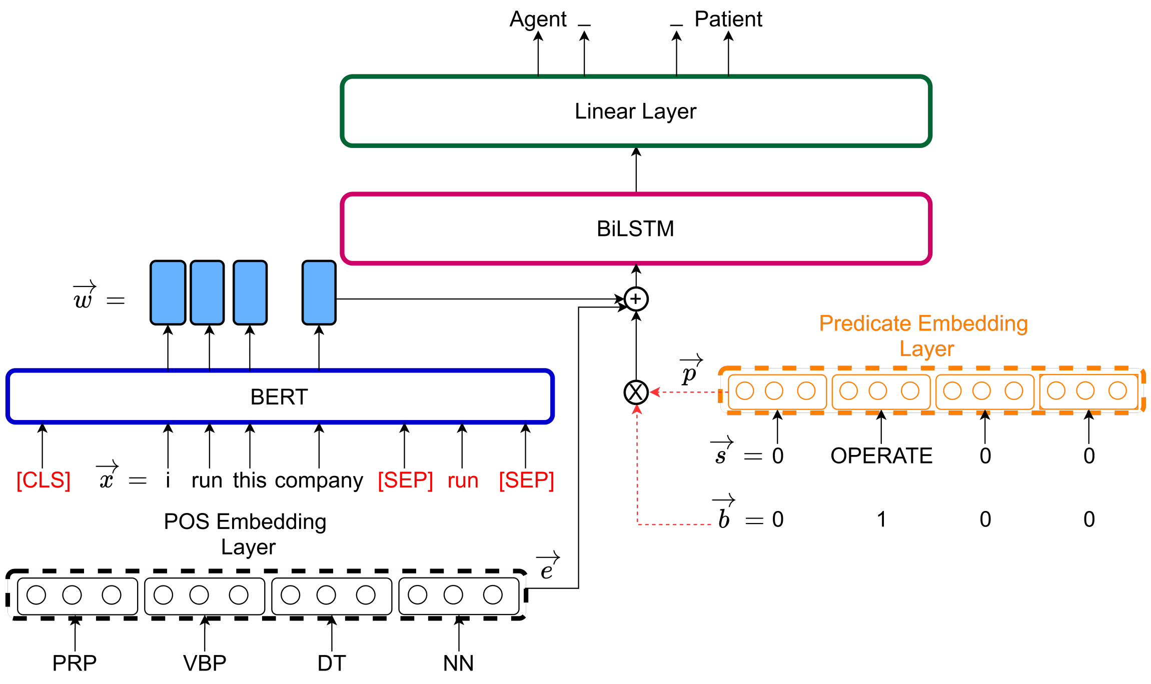


Figure 3: Overview of our BERT-based model architecture. BERT receives an input sentence of lemmas $\overrightarrow{x}$ preceded by the special "[CLS]" token and super-seceded by the predicate currently under consideration, x_p ("run" in the example in this figure), surrounded by the special separator token: "[SEP]x_p[SEP]". We consider only the contextualized embeddings of the lemmas $\overrightarrow{x}$ in order to produce $\overrightarrow{w}$ (ignoring all the tokens that have the red color in the figure). We also supply the POS Tag of each word and the model produces a representation for them, $\overrightarrow{e}$, via a POS embedding layer. The $\bigotimes$ symbol represents the XOR operation as the model receives either the binary predicate indicator, $\overrightarrow{b}$, or the predicate embedding indicator, $\overrightarrow{p}$. The $\bigoplus$ symbol represents the concatenation operation where the vectors $\overrightarrow{w}$, $\overrightarrow{e}$, and either predicate indicator $\overrightarrow{b}$ or $\overrightarrow{p}$ are concatenated before being fed into a BiLSTM. The output of which is fed to a linear layer that outputs the network's classification.

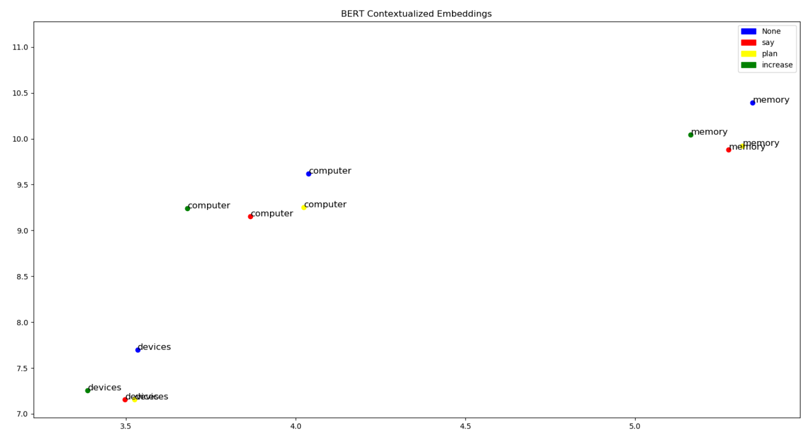
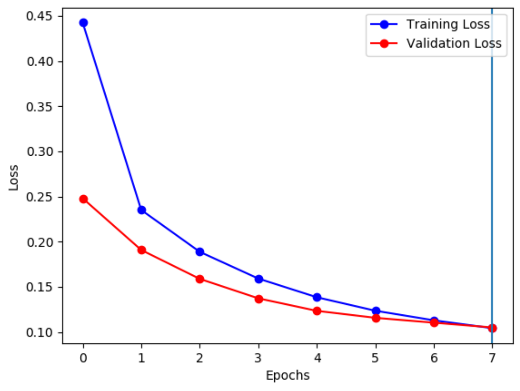
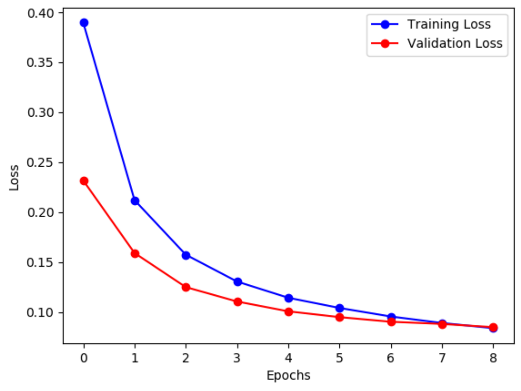


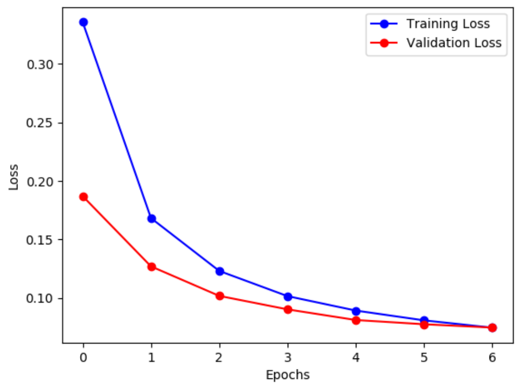
Figure 4: Contextualized embeddings produced for some lemmas of a sentence with the following 3 verbs: "say", "plan", and "increase". The sentence is fed into BERT 3 times as such: {[CLS] $sentence$ [SEP] x_p [SEP]}, each time replacing x_p with one of the aforementioned verbs. If we hadn't added [SEP] x_p [SEP] to the sentence each time we feed it to BERT, it would produce the exact same contextualized embeddings (blue) all 3 times. The BiLSTM that receives the output of BERT would see the exact same embeddings 3 times, and each time, they have a different label. Using our approach instead, we are able to obtain embeddings that are a function of the predicate we're currently considering, as the embeddings for a word like "computer" are different when we consider the predicate "say" (red) than "plan" (yellow) than "increase" (green).



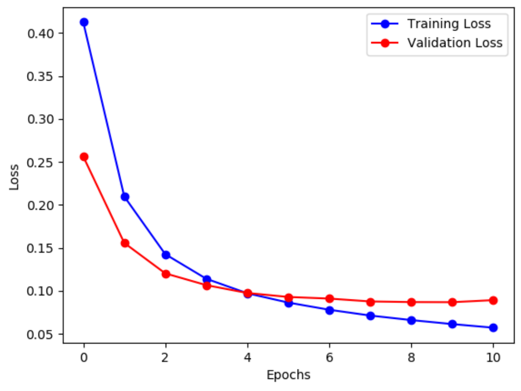
(a) Experiment 1 Training and Validation Loss



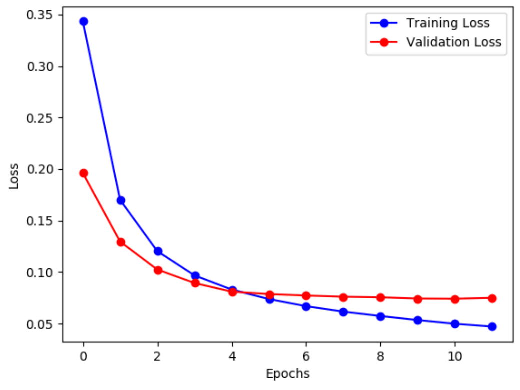
(b) Experiment 2 Training and Validation Loss



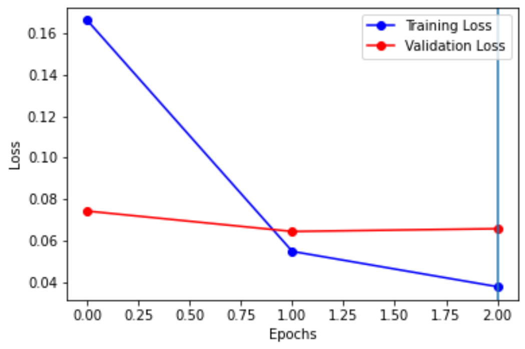
(c) Experiment 3 Training and Validation Loss



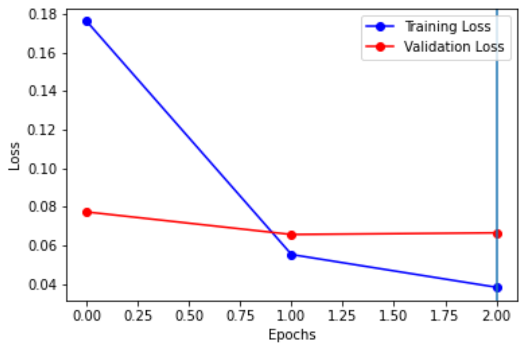
(d) Experiment 4 Training and Validation Loss



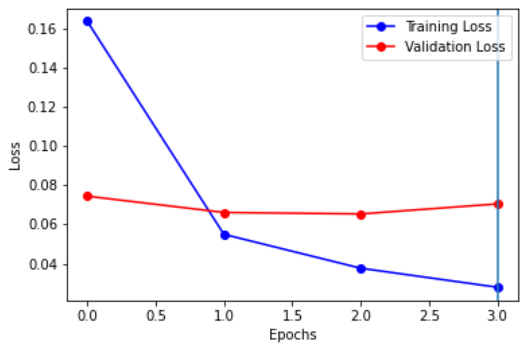
(e) Experiment 5 Training and Validation Loss



(f) Experiment 6 Training and Validation Loss



(g) Experiment 7 Training and Validation Loss



(h) Experiment 8 Training and Validation Loss

Figure 5: Training and Validation Losses for all 8 Experiments. Experiments 1 to 5 concern the Baseline model while 6 to 8 are the ones pertaining to BERT.