An Empirical Evaluation of various Deep Learning Architectures for Bi-Sequence Classification Tasks

Anirban Laha
IBM Research India
anirlaha@in.ibm.com

Vikas Raykar
IBM Research India
viraykar@in.ibm.com

Abstract

Several tasks in argumentation mining and debating, question-answering, and natural language inference involve classifying a sequence in the context of another sequence (referred as bi-sequence classification). For several single sequence classification tasks, the current state-of-the-art approaches are based on recurrent and convolutional neural networks. On the other hand, for bi-sequence classification problems, there is not much understanding as to the best deep learning architecture. In this paper, we attempt to get an understanding of this category of problems by extensive empirical evaluation of 19 different deep learning architectures (specifically on different ways of handling context) for various problems originating in natural language processing like debating, textual entailment and question-answering. Following the empirical evaluation, we offer our insights and conclusions regarding the architectures we have considered. We also establish the first deep learning baselines for three argumentation mining tasks.

1 Introduction

Argumentation mining is a relatively new challenge in corpus-based discourse analysis that involves automatically identifying argumentative structures within a corpus. Many tasks in argumentation mining (Lippi and Torroni, 2015a) and debating technologies (Slonim et al., 2014) involve categorizing a sequence in the context of another sequence. For example, in *context dependent claim detection* (Levy et al., 2014), given a sentence, one task is to identify whether the sentence contains a claim relevant to a particular debatable topic (generally given as a context sentence). Similarly in *context dependent evidence detection* (Rinott et al., 2015), given a sequence (possibly multiple sentences), one task is to detect if the sequence contains an evidence relevant to a particular topic. We refer to such class of problems in computational argumentation as *bi-sequence classification* problems—given two sequences s and c we want to predict the label for the target sequence s in the context of another sequence c[1]. Apart from the debating tasks, several other natural language inference tasks fall under the same paradigm of having a pair of sequences. For example, *recognizing textual entailment* (Bowman et al., 2015), where the task is to predict if the meaning of a sentence can be inferred from the meaning of another sentence. Another class of problems originated from question-answering systems also known as *answer selection*, where given a question, a candidate answer needs to be classified as an answer to the question at hand or not.

Recently, deep learning approaches have obtained very high performance across many different natural language processing tasks. These models can often be trained in an end-to-end fashion and do not require traditional, task-specific feature engineering. For many single sequence classification tasks, the state-of-the-art approaches are based on recurrent neural networks (RNN variants like Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014)) and convolution neural network based models (CNN) (Kim, 2014). Whereas for bi-sequence classification, the context sentence c has to be explicitly taken into account when performing the classification for the

[1] In this paper, we shall ignore the subtle distinction between sentence and sequence and both will mean just a text segment composed of words.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2762–2773, Osaka, Japan, December 11-17 2016.

target sentence s. The context can be incorporated into the RNN and CNN based models in various ways. However there is not much understanding in current literature as to the best way to handle context in these deep learning based models. In this paper, we empirically evaluate(see Section 4) the performance of five different ways of handling context in conjunction with target sentence(see Section 3) for multiple bi-sequence classification tasks(see Section 2) using architectures composed of RNNs and(/or) CNNs.

In a nutshell, this paper makes the two major novel contributions:

1. We establish the first deep learning based baselines for three bi-sequence classification tasks relevant to argumentation mining with zero feature engineering.

2. We empirically compare the performance of several ways handling context for bi-sequence classification problems in RNN and CNN based models. While some of these variants are used in various other tasks, there has been no formal comparison of different variants and this is the first attempt to actually list all the variants and compare them on several publicly available benchmark datasets.

2 Bi-Sequence classification tasks

In this section, we will briefly mention the various bi-sequence tasks of interest in the literature of argumentation mining and in the broader natural language inference domain.

2.1 Argumentation Mining

We mainly consider two prominent tasks in argumentation mining, namely, detecting the claims (Levy et al., 2014) and evidences (Rinott et al., 2015), within a given corpus, which are related to a prespecified topic. These two tasks together helps to automatically construct persuasive arguments out of a given corpora. We will define the following four concepts:
Motion - The topic under debate, typically a short phrase that frames the discussion.
Claim - A general, typically concise statement that directly supports or contests the motion.
Motion text - A document/article/discourse that contain claims with high probability.
Evidence - A set of statements that directly supports the claim for a given motion.

2.1.1 Context Dependent Claim Detection (CDCD)

Given a sentence in a motion text the task is to identify whether the sentence contains a claim relevant to the motion or not. This is the claim sentence task introduced by Levy et al. (2014). For example, each of the following sentences includes a claim, marked in italic, for the motion topic in brackets.

1. (**the sale of violent video games to minors**) Recent research has suggested that some *violent video games may actually have a pro-social effect in some contexts, for example, team play*.

2. (**the right to bear arms**) Some gun control organizations say that *increased gun ownership leads to higher levels of crime, suicide and other negative outcomes*.

2.1.2 Context Dependent Evidence Detection (CDED)

Given a segment in a motion text the task is to identify whether the segment contains an evidence relevant to the motion or not (Rinott et al., 2015). We consider evidences of two types in this paper, *Study* and *Expert*. Evidences of type study are generally results of a quantitative analysis of data given as numbers, or as conclusions. The following are two examples for study evidence relevant to the motion topic in brackets.

- (**the sale of violent video games to minors**) A 2001 study found that exposure to violent video games causes at least a temporary increase in aggression and that this exposure correlates with aggression in the real world.

- (**the right to bear arms**) In the South region where there is the highest number of legal guns per citizen only 59% of all murders were caused by firearms in contrast to 70% in the Northeast where there is the lowest number of legal firearms per citizen.

Evidence of type expert is a testimony by a person/group/commitee/organization with some known expertise/authority on the topic. The following are two examples for expert evidence relevant to the motion topic in brackets.

1. (**the sale of violent video games to minors**) This was also the conclusion of a meta-analysis by psychologist Jonathan Freedman, who reviewed over 200 published studies and found that the majority did not find a causal link.

2. (**the right to bear arms**) University of Chicago economist Steven Levitt argues that available data indicate that neither stricter gun control laws nor more liberal concealed carry laws have had any significant effect on the decline in crime in the 1990s.

2.2 Textual Entailment (TE)

This task (Bowman et al., 2015) corresponds to a multiclass setting, where given a pair of sentences (*premise* and *hypothesis*), the task is to identify whether one of them (*premise*) entails, contradicts or is neutral with respect to the other sentence (*hypothesis*). Unlike the other debating tasks seen previously, we cannot call these pair of sentences as context and target as these are more symmetric in nature. Typical examples[2] are the following (premise followed by hypothesis):

- Entailment: A soccer game with multiple males playing - Some men are playing a sport.
- Contradiction: A black race car starts up in front of a crowd of people - A man is driving down a lonely road.
- Neutral: A smiling costumed woman is holding an umbrella - A happy woman in a fairy costume holds an umbrella.

2.3 Answer Selection for Questions

Question Answering (QA) System is a natural extension to the traditional commercial search engines as it is concerned with fetching answers to natural language queries and returning the information accurately in natural human language. A QA system can be either closed-domain or open-domain, the former being restricted to a particular domain while the latter is not. *Answer sentence selection is a crucial subtask of the open-domain question answering problem, with the goal of extracting answers from a set of pre-selected sentences* (Yang et al., 2015). This is again bi-sequence classification task where the pair of sequences being a question and a candidate answer to be selected.

3 Deep Learning models for sequence pairs

All the tasks described in the previous section can be formulated as bi-sequence classification problems where we have to predict the label for the given pair of sequences. For simpler single sequence text classification tasks, RNN or CNN based architectures have become standard baselines. In this section, we will briefly introduce RNN and CNN and then subsequently describe RNN and CNN based architectures for bi-sequence classification tasks. Specifically, we talk about five different ways of handling context along with the target sentence.

3.1 Continuous Bag of Words (CBOW)

One of the simplest forms of sequence representation is the CBOW model, where every word in the sequence produces some word embedding (say, based on word2vec (Mikolov et al., 2013)) and the average of the word embedding vectors over the words produces the representation of the sequence. As is evident, this form of representation totally disregards the word order of the sequence.

3.2 Recurrent neural networks (RNNs)

The RNN model provides a framework for conditioning on the entire history of the sequence without resorting to the Markov assumption traditionally used for modelling sequences. Unlike CBOW, RNNs encode arbitrary length sequences as fixed size vectors without disregarding the word order.

Given an ordered list of n input vectors $\mathbf{x}_1, ..., \mathbf{x}_n$ and an initial state vector $\mathbf{s}_0$, a RNN generates an ordered list of n state vectors $\mathbf{s}_0, ..., \mathbf{s}_n$ and an ordered list of n output vectors $\mathbf{y}_1, ..., \mathbf{y}_n$, that is, $RNN(\mathbf{s}_0, \mathbf{x}_1, ..., \mathbf{x}_n) = \mathbf{s}_1, ..., \mathbf{s}_n, \mathbf{y}_1, ..., \mathbf{y}_n$. The input vectors $\mathbf{x}_i$ (which corresponds to a fixed dimensional representation for each word in the sequence) are presented to the RNN in a sequential fashion and $\mathbf{s}_i$ represents the state of the RNN after observing the inputs $\mathbf{x}_1, ..., \mathbf{x}_i$. The output vector $\mathbf{y}_i$ is a

[2] http://nlp.stanford.edu/projects/snli/

function of the corresponding state vector $\mathbf{s}_i$ and is then used for further prediction. An RNN is given by the following update equations:

$$\mathbf{s}_i = R(\mathbf{x}_i, \mathbf{s}_{i-1}) \quad (1)$$

$$\mathbf{y}_i = O(\mathbf{s}_i) \quad (2)$$

The recursively defined function R takes as input the previous state vector $\mathbf{s}_{i-1}$ and the current input vector $\mathbf{x}_i \in \mathbb{R}^{d_x}$ and results in an updated state vector $\mathbf{s}_i \in \mathbb{R}^{d_s}$. An additional function O maps the state vector $\mathbf{s}_i$ to an output vector $\mathbf{y}_i \in \mathbb{R}^{d_y}$. Different instantiations of R and O will result in the different network structures (Simple RNN, LSTM (Hochreiter and Schmidhuber, 1997), GRU (Cho et al., 2014), etc.). The final state vector $\mathbf{s}_n$ can be thought of as encoding the entire input sequence into a fixed size vector, which can be passed to a softmax layer to produce class probabilities.

3.3 Convolutional Neural Networks (CNNs)

CNNs are built on the premise of locality and parameter sharing which has proven to produce very effective feature representation for images. Following the groundbreaking work by Kim (2014), there has been a lot of interest shown by the text community towards applying CNNs for modelling text representation.

As in case of RNNs defined above, an n-word sentence consists of embedding vectors $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathbb{R}^{d_x}$, one for each word in the sentence. Let $\mathbf{x}_{i:i+j}$ denote the concatenation of words $\mathbf{x}_i, \mathbf{x}_{i+1}, ..., \mathbf{x}_{i+j}$. A convolution operation defined by a non-linear function f applies a filter $\mathbf{w} \in \mathbb{R}^{hd_x}$ to a window of h words to produce a single feature value as given below:

$$c_i = f(\mathbf{w}.\mathbf{x}_{i:i+h-1} + b) \quad (3)$$

$$\mathbf{c} = [c_1, c_2, ..., c_{n-h+1}] \quad (4)$$

In the next step, max-pooling is applied which essentially produces a single feature value $\hat{c} = max\{\mathbf{c}\}$, corresponding to one filter that has been used. The model can have multiple feature values, one for each applied filter, thus producing a feature representation for the input sentence, which can again be passed to a softmax layer to produce class probabilities.

3.4 Bi-Sequence RNN models

For bi-sequence classification tasks we use two RNNs, one RNN to encode the context sentence (*context RNN*) and another separate RNN encode the target sentence (*target RNN*). We define the following five different variants of combining these two RNNs for bi-sequence classification tasks (see Figure 1 for illustration of these variants).

1. **conditional-state**: The final state of the context RNN is fed as the initial state of the target RNN. This way of handling context for RNNs has been previously used in conversational systems (Vinyals and Le, 2015), image description (Vinyals et al., 2015) and image question answering (Ren et al., 2015) systems.

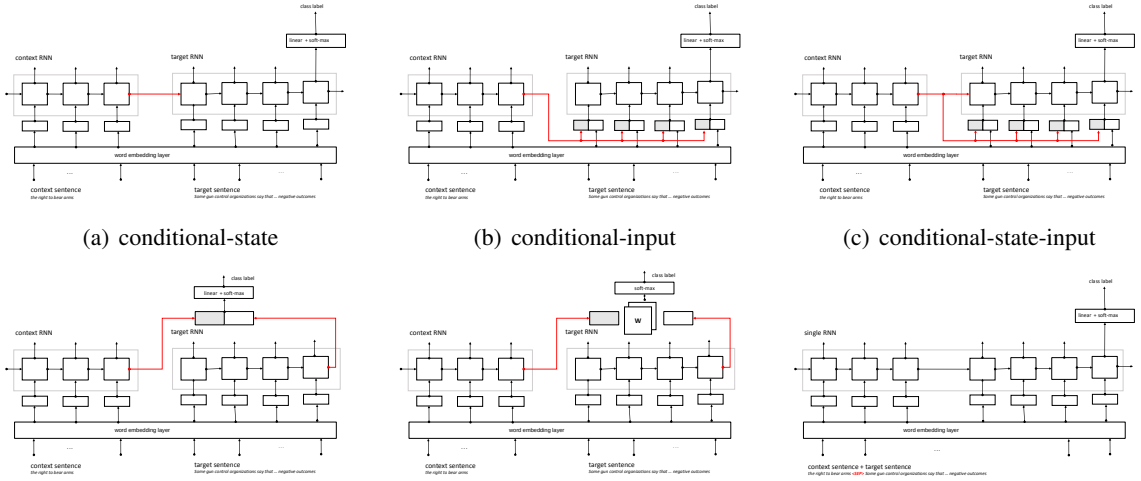
2. **conditional-input**: The final state of the context RNN is fed as auxiliary input (concatenated with every input) for the target RNN. This way of handling context has been previously used in machine translation tasks (Sutskever et al., 2014).

3. **conditional-state-input**: The final state of the context RNN is fed as the initial state of the target RNN and also fed as input for target RNN concatenated with every input.

4. **concat**: The final states of both the context and the target RNN are concatenated and then fed to a softmax layer for the label prediction.

5. **bi-linear**: The final states of both the context and the target RNN are combined using a bi-linear form ($\mathbf{x}^\top \mathbf{W}\mathbf{y}$) with a softmax function for the final prediction. There are different $\mathbf{W}$ for different classes under consideration.

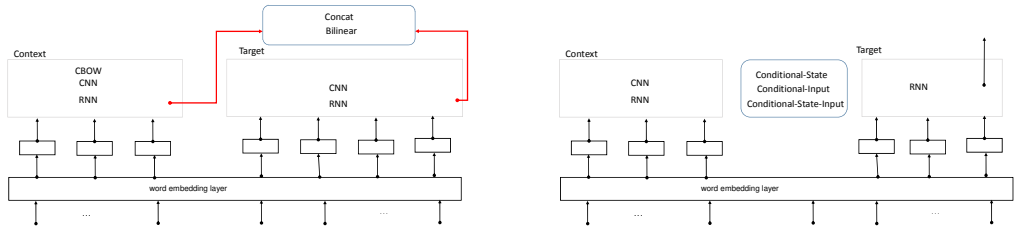
From here on, we would refer to architecture types 1, 2 and 3 as *conditional* variants while the others will be addressed as is. In addition, we consider another baseline variant **concat-sentence**, in which we concatenate the context and the sentence with a special separator token and feed the entire concatenated sequence to a single RNN. For all these variants we use a common embedding layer. Also note that the conditional variants require a common RNN size for both the context and the target RNNs. Even though that restriction is not there for other variants, we choose the same RNN size anyways for convenience.



(a) conditional-state (b) conditional-input (c) conditional-state-input

(d) concat (e) bilinear (f) concat-sentence

Figure 1: RNN based Architectures for bi-sequence classification.



(a) concat-bilinear-all-combinations (b) conditional-all-combinations

Figure 2: Multiple model variants for bi-sequence classification.

3.5 Bi-Sequence model variants

In this paper, we consider multiple ways of extending the bi-sequence architectures mentioned in section 3.4, by replacing RNN with CBOW or CNN either for context or target or both. For the variations *concat* and *bi-linear* (see Fig. 2(a)), we have considered CBOW, RNN and CNN for context representation whereas we have RNN and CNN for target representation. In tasks where context has very few words (say debating tasks or question-answering), a simple representation like CBOW may work for context. However, we haven't considered modelling target using CBOW as targets are usually of larger length. For the *conditional* variants (see Fig. 2(b)), we haven't considered CBOW due to their limited modelling capacity and there is no softmax layer directly on top of context representation to compensate for it (even though softmax is only on top of target RNN). Moreover, target can only be RNN as there is no concept of hidden state for CNNs. Hence, we use CNN and RNN for context whereas RNN for target. In addition, we consider the concat-sentence (mentioned in section 3.4) as a baseline. This leads to 19 architectures for empirical comparison (12 from Fig. 2(a) and 6 from Fig. 2(b) and the baseline).

4 Experiments

We have carried out extensive evaluation of the above architecture variants over a wide range of datasets related to argumentation mining as well as datasets appealing to the larger natural language community

like textual entailment and question answering data. We consider data with class imbalance problem as well as balanced and we do not restrict ourselves to binary classification by working with multiclass dataset as well. As can be found in the Table 1, we consider the following tasks related to the domain of argumentation mining(Aharoni et al., 2014), which is available here [3] :

- Claim Sentence : This is the dataset for the CDCD task defined in section 2.1.1. This is the current benchmark dataset for the Claim Detection task. There are a total 47183 canditate claims distributed among 33 motions.
- EXPERT Evidence : This is corresponding to the CDED task defined in section 2.1.2 for evidence type EXPERT. There are 56985 labelled candidates for 57 different motion topics.
- STUDY Evidence : For evidence type STUDY, the dataset consists of 33534 labelled candidates for 49 motion topics.

Table 1 summarizes all of the datasets above. Interesting point to be noted here is that all the datasets above have very low number of positives and the architectures we are evaluating need to be resilient to the class imbalance problem for these datasets. Other than the debating datasets listed above, we also consider two datasets related to more popular problems in the natural language processing community:

- Textual Entailment (TE) [4] (Bowman et al., 2015) :This dataset consist of around 500K instances evenly distributed across all three classes. So, here we have a multiclass problem in a balanced setting.
- WikiQA [5] (Yang et al., 2015) : There are around 29K labelled question/answer pairs at our disposal.

Task	Motions	Data Size	Positives
Claim Sentence	33	47183	2.77%
EXPERT Evidence	57	56985	4.56%
STUDY Evidence	49	33534	3.74%

Table 1: Argument Mining Datasets.

Task	Train	Dev	Test	Problem	Class
TE	549367	9842	9824	Multiclass	Balance
WikiQA	20360	2733	6165	Binary	Imbalance

Table 2: More Datasets.

4.1 Experimental Setup

For each of the architectures mentioned in section 3.5, we choose the best configuration of hyperparameters based on the validation portion of the particular dataset (For Claim and Evidence datasets, we consider a train:valid:test split of 60:10:30 while for the TE and WikiQA datasets we consider their corresponding given split).

Performance of the best performing configuration for every architecture is reported on the test data using the appropriate metric. As the claim and expert and study Evidences had similar data characteristics (in terms of data size and context and target lengths), we did extensive hyperparam tuning only on the claim dataset and applied the best configurations without further tuning to the expert and study datasets. Exhaustive hyperparam tuning was done on the TE dataset as well because its data characteristics are very different from other datasets.

In addition to reporting the test metrics for argumentation datasets, we carried out Leave-One-Out(leaving one motion out for testing) Mode training and evaluation which is more appropriate for this problem setting as it is crucial that we generalize well to totally unseen motion topics. In this case, we report the macro-average metrics over all motions.

4.2 Hyperparameter Tuning

Considering the number of variations of combinations of architectures we have considered, we have a huge hyperparameter space to deal with. Hence, we decided to fix insignificant hyperparameters and focus only on the relevant ones. We have decided to use word2vec (Mikolov et al., 2013) pretrained models for initializing the word embeddings across CBOW, CNN and RNNs and made them trainable specific to task at hand. In addition we have found through minimal tuning that the Adam (Kingma and Ba, 2015) optimizer seems to work best. We have also found that a learning rate of 0.001 works best

[3] https://www.research.ibm.com/haifa/dept/vst/mlta_data.shtml
[4] http://nlp.stanford.edu/projects/snli/
[5] http://aka.ms/WikiQA

in most scenarios except when the parameter space in some architectures (for ex, *bi-linear*) is large, in which case lower learning rates of 0.0001 or 0.00001 worked well. Rather than tuning the maximum sequence length for context and target sentence, we tried to fix it by getting reasonable values by plotting histogram of sequence lengths and had a cut-off at around 98-99 percentile. The max lengths turned out to be 14 for context and 60 for target for Claim, Evidence and WikiQA datasets while for textual entailment, they turned out to be 30 in both due to the symmetrical nature between premise and hypothesis. We tuned the following hyperparameters:

RNN Model : GRU or LSTM.

RNN Size : 50,100,200,300,400,500,1000.

CNN Filter Sizes : 3,3+4,3+4+5,2+3+4+5.

CNN Number of Filters : 10,20,40,64,128.

L2 Reg coeff for CNN : 0, 0.01, 0.001, 0.0001.

For every architecture type, we carried out the optimization of the relevant hyperparams from the above list over the whole grid. One point to note is that for certain architectures like conditional-state, as output of context is fed in to the hidden state of target RNN, there are some restrictions in the allowable context RNN/CNN hyperparam configurations based on the target RNN settings as the output dimension of the context RNN/CNN needs to match the hidden state dimension of the target RNN.

4.3 Evaluation Metrics

For the datasets Claim, Expert, Study Evidence and WikiQA datasets, we have used standard evaluation measures like Average Precision (Area under Precision Recall Curve) and AUC to choose best hyperparam configurations based on validation data as well as report test metrics. For argument mining specific tasks, we have reported other additional metrics like P@200, R@200, F1@200, P@50, R@50 and F1@50 (Levy et al., 2014) in addition to reporting AUC and Average Precision. Please note for leave-one-out mode, the reported metrics are macro-average over all motion topics. For Textual entailment, since it is a more balanced dataset, reporting valid and test accuracies are standard in the literature and we have done the same.

Task	Context	Target	Architecture	Test AVGP
Claim Sentence	RNN	CNN	**Concat**	**0.307**
	CNN	CNN	Concat	0.304
	Concat-Sentence baseline			0.17
EXPERT Evidence	RNN	RNN	**Conditional-State-Input**	**0.257**
	CNN	CNN	Concat	0.254
	Concat-Sentence baseline			0.225
STUDY Evidence	CNN	CNN	**Concat**	**0.297**
	RNN	CNN	Concat	0.29
	Concat-Sentence baseline			0.236
WikiQA	CBOW	RNN	**Concat**	**0.187**
	CNN	RNN	Conditional-State-Input	0.186

Table 3: Empirical evaluation based on Average Precision on assymetric datasets.

Task	Context	Target	Architecture	Test AUC
Claim Sentence	CNN	CNN	**Concat**	**0.873**
	CNN	RNN	Conditional-State	0.873
	Concat-Sentence baseline			0.831
EXPERT Evidence	RNN	RNN	**Conditional-State**	**0.832**
	RNN	RNN	Conditional-State-Input	0.823
	Concat-Sentence baseline			0.805
STUDY Evidence	CNN	CNN	**Concat**	**0.87**
	CBOW	CNN	Concat	0.864
	Concat-Sentence baseline			0.844
WikiQA	CNN	CNN	**Concat**	**0.74**
	CBOW	RNN	Concat	0.74

Table 4: Empirical evaluation based on AUC on assymetric datasets.

Method	Model	TrainAcc(%)	TestAcc(%)
(Bowman et al., 2015)	Use of features incl unigrams and bigrams	99.7	78.2
(Vendrov et al., 2016)	1024D GRU encoders w/ unsupervised 'skip-thoughts' pre-training	98.8	81.4
(Mou et al., 2016)	300D Tree-based CNN encoders	83.3	82.1
(Cheng et al., 2016)	450D LSTMN with deep attention fusion	88.5	86.3
(Parikh et al., 2016)	200D decomposable attention model with intra-sentence attention	90.5	**86.8**
Conditional-State-RNN-RNN	Simple architecture with RNNs without attention	89.97	82.36

Table 5: Comparison with the state-of-the-art in Textual Entailment dataset.

Method	P@200	R@200	F1@200	P@50	R@50	F1@50	AVGP	AUC
CDCD (Levy et al., 2014)**	9.0	**73.0**	-	18.0	40.0	-	-	-
BoW (Lippi and Torroni, 2015b)	8.2	51.7	14.2	-	-	-	0.117	0.771
TK (Lippi and Torroni, 2015b)	9.8	58.7	16.8	-	-	-	0.161	0.808
TK+Topic (Lippi and Torroni, 2015b)	**10.5**	62.9	**18.0**	-	-	-	0.178	0.823
Concat-CNN-CNN	9.64	61.5	15.8	17.1	27.7	19.2	0.173	0.812
Conditional-State-Input-RNN-RNN	9.56	60.0	15.6	16.6	26.9	18.5	0.162	0.801

Table 6: Results in Leave-One-Motion-Out mode for Claim Sentence Task. **Levy et al. (2014) used a smaller version of the dataset consisting of only 32 motions and also less number of claims. For fair comparison, we also use the same version of dataset as in CDCD and report the results in Appendix A.

Task	P@200	R@200	F1@200	P@50	R@50	F1@50	P@20	P@10	P@5	AVGP	AUC
Claim Sentence Task	9.64	61.5	15.8	17.1	27.7	19.2	22.4	27.9	28.5	0.173	0.812
EXPERT Evidence Task	9.53	64.0	14.5	14.5	35.0	15.7	18.6	21.1	22.5	0.160	0.750
STUDY Evidence Task	8.33	79.5	13.5	15.5	53.9	18.9	20.8	25.3	31.8	0.298	0.836

Table 7: Numbers in Leave-One-Motion-Out mode for all three debating tasks using our approach.

4.4 Results and Discussion

Tables 3 and 4 report the two top ranking architectures for four datasets based on Test AUC and Test Avg Precision. We find that **Concat** is the winning architecture variant across majority of the datasets considered. Moreover, the runner-up architecture type **Conditional-State-Input** is also similar to 'Concat' in the sense that concatenation of context representation is done at the input of the sentence RNN. Now the four datasets we considered are asymmetric in nature as there are significantly fewer contexts (motions or questions) than the targets. Hence the context model does not see enough data for learning and hence, if the learnt context model is fed directly to the hidden state of the target RNN, the improperly learnt context model can play a big role. In contrast if a Concat kind of architecture is used, the linear plus softmax layer can decide on how much importance to give to the context model. Hence, Concat is doing better in this case.

From Textual entailment dataset(which is symmetric in nature), we found that conditional type of architectures are doing better at the Test accuracies. In fact, the winning architecture was **Conditional-State** with RNN-RNN combo, which did better in terms of test accuracy than the feature based models (Bowman et al., 2015) and one tree-based model (Mou et al., 2016). However, it came close to the state-of-the-art attention based model (Parikh et al., 2016). In our work we are empirically evaluating simple architectures for bisequence classification without using more sophisticated tree-based or attention-based models. It is possible that adding attention on top of this will improve the results further.

The *bi-linear* model, is supposed to capture the interaction between the context and target reps via a quadratic form (section 3.4). For the asymmetric datasets, this is not doing well again due to insufficient data for context. Whereas, it does well for the TE data. However, due to the huge parameter space for bi-linear, training times are considerably higher and requires lower learning rate than other architecture types. The runtimes are comparable for the other architecture variants.

From Table 6, the main takeaway is that we are the only deep learning based method with zero feature engineering and we have come very close to the state-of-the-art systems (Levy et al., 2014) and (Lippi and Torroni, 2015b), which are heavily feature-engineered. Here again the winner is a 'Concat' based combination of architecture. Moreover, Tables 6 and 7 are the first deep learning zero feature engineered baselines for all argument mining datasets. Appendix A contains the details of the exhaustive

experiments on all architectures on the different datasets in terms of the best hyperparameters used.

5 Conclusion

In this work, we have considered taking up multiple architectures for bisequence classification tasks, for which not much understanding is there in the current literature. In addition to suggesting winning architecture recipes for different kinds of datasets, we have established deep learning based baselines for argument mining tasks with zero feature engineering. As future work, it remains to be seen how adding attention on top of winning simple architectures fare in terms of benchmark performance.

6 Acknowledgements

We would like to thank Mitesh M. Khapra for the multiple discussions that we had with him leading to this paper. In addition, we are also grateful to our colleagues at IBM Debating Technologies for the numerous suggestions and feedback at different points of time.

References

Ehud Aharoni, Anatoly Polnarov, Tamar Lavee, Daniel Hershcovich, Ran Levy, Ruty Rinott, Dan Gutfreund, and Noam Slonim. 2014. A benchmark dataset for automatic detection of claims and evidence. In *in the Context of Controversial Topics", in Proceedings of the First Workshop on Argumentation and Computation, ACL 2014*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations, San Diego, 2015*.

Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 1489–1500.

Marco Lippi and Paolo Torroni. 2015a. Argument mining: A machine learning perspective. In *Theory and Applications of Formal Argumentation - Third International Workshop, TAFA 2015, Buenos Aires, Argentina, July 25-26, 2015, Revised Selected Papers*, pages 163–176.

Marco Lippi and Paolo Torroni. 2015b. Context-independent claim detection for argument mining. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 185–191.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 130–136, Berlin, Germany, August. Association for Computational Linguistics.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, US., November. Association for Computational Linguistics.

Mengye Ren, Ryan Kiros, and Richard S. Zemel. 2015. Exploring models and data for image question answering. In *In Advances in Neural Information Processing Systems*, pages 2935–2943.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M. Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence - an automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 440–450.

Noam Slonim, Ehud Aharoni, Carlos Alzate Perez, Roy Bar-Haim, Yonatan Bilu, Lena Dankin, Iris Eiron, Daniel Hershcovich, Shay Hummel, Mitesh M. Khapra, Tamar Lavee, Ran Levy, Paul Matchen, Anatoly Polnarov, Vikas C. Raykar, Ruty Rinott, Amrita Saha, Naama Zwerdling, David Konopnicki, and Dan Gutfreund. 2014. Claims on demand - an initial demonstration of a system for automatic detection and polarity identification of context dependent claims in massive corpora. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference System Demonstrations, August 23-29, 2014, Dublin, Ireland*, pages 6–9.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *4th International Conference for Learning Representations, Puerto Rico, 2016*.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, Portugal, September. Association for Computational Linguistics.

A Appendix

Method	**P@200**	**R@200**	**F1@200**	**P@50**	**R@50**	**F1@50**	**AVGP**	**AUC**
CDCD (Levy et al., 2014)	**9.0**	**73.0**	-	**18.0**	**40.0**	-	-	-
Concat-CNN-CNN	7.29	60.5	12.4	14.7	31.5	18.3	0.166	0.810
Conditional-State-Input-RNN-RNN	6.87	58.1	11.7	13.5	29.5	17.0	0.163	0.789

Table 8: Results in Leave-One-Motion-Out mode for Claim Sentence Task according to the dataset used by Levy et al. (2014).

Architecture	**Context**	**Target**	**Setting**	**Test AVGP**
Concat	CBOW	CNN	FilterSize:3,Filters:40	0.3
Concat	CBOW	RNN	Cell:GRU,Size:300	0.276
Concat	RNN	CNN	Cell:LSTM,Size:200,FilterSize:3+4+5,Filters:20	**0.307**
Concat	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:64	0.28
Concat	RNN	RNN	Cell:GRU,Size:300	0.27
Concat	CNN	CNN	FilterSize:3+4,Filters:64,L2:0.01	0.304
Bilinear	CBOW	CNN	FilterSize:3+4+5,Filters:40	0.237
Bilinear	CBOW	RNN	Cell:GRU,Size:300	0.263
Bilinear	RNN	CNN	Cell:GRU,Size:200,FilterSize:3+4+5,Filters:64	0.254
Bilinear	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:10	0.268
Bilinear	RNN	RNN	Cell:GRU,Size:200	0.263
Bilinear	CNN	CNN	FilterSize:3+4,Filters:20	0.237
Conditional-State	CNN	RNN	Cell:GRU,Size:200,FilterSize:3+4,Filters:100	0.248
Conditional-State	RNN	RNN	Cell:GRU,Size:200	0.266
Conditional-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3+4+5,Filters:100	0.254
Conditional-Input	RNN	RNN	Cell:GRU,Size:100	0.246
Conditional-State-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3,Filters:300	0.264
Conditional-State-Input	RNN	RNN	Cell:GRU,Size:100	0.247
Concat-Sentence baseline			Cell:GRU,Size:200	0.17

Table 9: Best configurations of all architectures on Claim Sentence Dataset tuning based on AVGP

Architecture	Context	Target	Setting	Test AUC
Concat	CBOW	CNN	FilterSize:3+4+5,Filters:64,L2:0.01	0.863
Concat	CBOW	RNN	Cell:GRU,Size:200	0.868
Concat	RNN	CNN	Cell:GRU,Size:200,FilterSize:3,Filters:40	0.867
Concat	CNN	RNN	Cell:LSTM,Size:200,FilterSize:3+4+5,Filters:20	0.855
Concat	RNN	RNN	Cell:GRU,Size:100	0.864
Concat	CNN	CNN	FilterSize:3,Filters:128,L2:0.01	**0.873**
Bilinear	CBOW	CNN	FilterSize:3,Filters:128,L2:0.01,LR:0.0001	0.831
Bilinear	CBOW	RNN	Cell:GRU,Size:500	0.832
Bilinear	RNN	CNN	Cell:LSTM,Size:100,FilterSize:3+4,Filters:128,LR:0.00001	0.828
Bilinear	CNN	RNN	Cell:LSTM,Size:200,FilterSize:3+4+5,Filters:10,LR:0.0001	0.857
Bilinear	RNN	RNN	Cell:LSTM,Size:300,LR:0.0001	0.855
Bilinear	CNN	CNN	FilterSize:3+4,Filters:64,L2:0.001,LR:0.0001	0.82
Conditional-State	CNN	RNN	Cell:GRU,Size:48,FilterSize:3+4+5,Filters:16,LR:0.0001	0.873
Conditional-State	RNN	RNN	Cell:GRU,Size:100	0.86
Conditional-Input	CNN	RNN	Cell:GRU,Size:50,FilterSize:3,Filters:50,LR:0.0001	0.873
Conditional-Input	RNN	RNN	Cell:GRU,Size:200	0.86
Conditional-State-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3+4,Filters:150,LR:0.00001	0.856
Conditional-State-Input	RNN	RNN	Cell:GRU,Size:200	0.862
Concat-Sentence baseline			Cell:GRU,Size:200	0.83

Table 10: Best configurations of all architectures on Claim Sentence Dataset tuning based on AUC

Architecture	Context	Target	Setting	Test AVGP	Test AUC
Concat	CBOW	CNN	FilterSize:3,Filters:40	0.239	0.81
Concat	CBOW	RNN	Cell:GRU,Size:300	0.251	0.819
Concat	RNN	CNN	Cell:LSTM,Size:200,FilterSize:3+4+5,Filters:20	0.242	0.812
Concat	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:64	0.231	0.794
Concat	RNN	RNN	Cell:GRU,Size:300	0.241	0.811
Concat	CNN	CNN	FilterSize:3+4,Filters:64,L2:0.01	0.254	0.819
Bilinear	CBOW	CNN	FilterSize:3+4+5,Filters:40	0.218	0.79
Bilinear	CBOW	RNN	Cell:GRU,Size:300	0.202	0.788
Bilinear	RNN	CNN	Cell:GRU,Size:200,FilterSize:3+4+5,Filters:64	0.219	0.789
Bilinear	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:10	0.229	0.791
Bilinear	RNN	RNN	Cell:GRU,Size:200	0.214	0.788
Bilinear	CNN	CNN	FilterSize:3+4,Filters:20	0.233	0.792
Conditional-State	CNN	RNN	Cell:GRU,Size:200,FilterSize:3+4,Filters:100	0.226	0.797
Conditional-State	RNN	RNN	Cell:GRU,Size:200	0.254	**0.832**
Conditional-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3+4+5,Filters:100	0.229	0.797
Conditional-Input	RNN	RNN	Cell:GRU,Size:100	0.231	0.817
Conditional-State-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3,Filters:300	0.211	0.796
Conditional-State-Input	RNN	RNN	Cell:GRU,Size:100	**0.257**	0.823
Concat-Sentence baseline			Cell:GRU,Size:200	0.225	0.805

Table 11: Performance of all architectures on EXPERT Evidence Dataset

Architecture	Context	Target	Setting	Test AVGP	Test AUC
Concat	CBOW	CNN	FilterSize:3,Filters:40	0.281	0.864
Concat	CBOW	RNN	Cell:GRU,Size:300	0.279	0.851
Concat	RNN	CNN	Cell:LSTM,Size:200,FilterSize:3+4+5,Filters:20	0.29	0.863
Concat	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:64	0.262	0.829
Concat	RNN	RNN	Cell:GRU,Size:300	0.28	0.842
Concat	CNN	CNN	FilterSize:3+4,Filters:64,L2:0.01	**0.297**	**0.869**
Bilinear	CBOW	CNN	FilterSize:3+4+5,Filters:40	0.271	0.831
Bilinear	CBOW	RNN	Cell:GRU,Size:300	0.202	0.788
Bilinear	RNN	CNN	Cell:GRU,Size:200,FilterSize:3+4+5,Filters:64	0.271	0.833
Bilinear	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:10	0.254	0.839
Bilinear	RNN	RNN	Cell:GRU,Size:200	0.257	0.84
Bilinear	CNN	CNN	FilterSize:3+4,Filters:20	0.275	0.835
Conditional-State	CNN	RNN	Cell:GRU,Size:200,FilterSize:3+4,Filters:100	0.254	0.835
Conditional-State	RNN	RNN	Cell:GRU,Size:200	0.267	0.861
Conditional-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3+4+5,Filters:100	0.245	0.838
Conditional-Input	RNN	RNN	Cell:GRU,Size:100	0.28	0.854
Conditional-State-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3,Filters:300	0.257	0.839
Conditional-State-Input	RNN	RNN	Cell:GRU,Size:100	0.25	0.849
Concat-Sentence baseline			Cell:GRU,Size:200	0.236	0.844

Table 12: Performance of all architectures on STUDY Evidence Dataset

Architecture	Context	Target	Setting	Test AVGP	Test AUC
Concat	CBOW	CNN	FilterSize:3,Filters:40	0.162	0.735
Concat	CBOW	RNN	Cell:GRU,Size:300	**0.187**	0.74
Concat	RNN	CNN	Cell:LSTM,Size:200,FilterSize:3+4+5,Filters:20	0.15	0.727
Concat	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:64	0.119	0.66
Concat	RNN	RNN	Cell:GRU,Size:300	0.171	0.705
Concat	CNN	CNN	FilterSize:3+4,Filters:64,L2:0.01	0.179	**0.74**
Bilinear	CBOW	CNN	FilterSize:3+4+5,Filters:40	0.129	0.672
Bilinear	CBOW	RNN	Cell:GRU,Size:300	0.119	0.656
Bilinear	RNN	CNN	Cell:GRU,Size:200,FilterSize:3+4+5,Filters:64	0.122	0.676
Bilinear	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:10	0.131	0.681
Bilinear	RNN	RNN	Cell:GRU,Size:200	0.149	0.688
Bilinear	CNN	CNN	FilterSize:3+4,Filters:20	0.129	0.712
Conditional-State	CNN	RNN	Cell:GRU,Size:200,FilterSize:3+4,Filters:100	0.122	0.681
Conditional-State	RNN	RNN	Cell:GRU,Size:200	0.171	0.739
Conditional-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3+4+5,Filters:100	0.141	0.713
Conditional-Input	RNN	RNN	Cell:GRU,Size:100	0.184	0.729
Conditional-State-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3,Filters:300	0.186	0.726
Conditional-State-Input	RNN	RNN	Cell:GRU,Size:100	0.169	0.714

Table 13: Performance of all architectures on WikiQA Dataset

Architecture	Context	Target	Setting	TrainAcc(%)	ValidAcc(%)	TestAcc(%)
Concat	CBOW	CNN	FilterSize:3+4+5,Filters:128	74.33	69.43	68.44
Concat	CBOW	RNN	Cell:LSTM,Size:400	72.75	69.4	69.02
Concat	RNN	CNN	Cell:GRU,Size:200,FilterSize:3+4+5,Filters:20	74.01	69.34	68.96
Concat	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:20	72.93	69.99	69.69
Concat	RNN	RNN	Cell:LSTM,Size:200	72.74	69.96	69.46
Concat	CNN	CNN	FilterSize:3+4+5,Filters:64	74.55	69.49	68.97
Bilinear	CBOW	CNN	FilterSize:3+4,Filters:128	83.86	77.1	77.07
Bilinear	CBOW	RNN	Cell:GRU,Size:300	84.78	79.07	78.19
Bilinear	RNN	CNN	Cell:GRU,Size:500,FilterSize:2+3+4+5,Filters:200	84.42	77.68	77.18
Bilinear	CNN	RNN	Cell:GRU,Size:500,FilterSize:2+3+4+5,Filters:200	83.71	78.72	78.6
Bilinear	RNN	RNN	Cell:GRU,Size:1000,LR:0.0001	84.91	81.1	80.3
Bilinear	CNN	CNN	FilterSize:3+4,Filters:128,LR:0.0001	84.51	76.58	76.81
Conditional-State	CNN	RNN	Cell:GRU,Size:500,FilterSize:3,Filters:500	87.77	80.87	80.81
Conditional-State	RNN	RNN	Cell:GRU,Size:500	**89.97**	**82.38**	**82.36**
Conditional-Input	CNN	RNN	Cell:GRU,Size:500,FilterSize:3+4,Filters:250	87.36	80.81	81.1
Conditional-Input	RNN	RNN	Cell:GRU,Size:500	89.02	81.45	80.92
Conditional-State-Input	CNN	RNN	Cell:GRU,Size:500,FilterSize:3,Filters:500	85.78	80.05	79.61
Conditional-State-Input	RNN	RNN	Cell:GRU,Size:500	89.03	81.93	81.38

Table 14: Performance of all architectures on Textual Entailment Dataset