An Empirical Study on Identifying Sentences with Salient Factual Statements

Damian Jimenez
Computer Science and Engineering
The University of Texas at Arlington
Arlington, Texas
damian.jimenez@mavs.uta.edu

Chengkai Li
Computer Science and Engineering
The University of Texas at Arlington
Arlington, Texas
cli@uta.edu

***Abstract*—In this paper, we show that by using a relatively simple neural network architecture and including edge (i.e., nonsensical) cases into a dataset we can more reliably identify factual claims than predecessor SVM models. Doping the dataset with these nonsensical example results in a more robust model overall that is resistant to being tricked into classifying sentences into a certain category based on easily met criteria. Furthermore, we show that the use of multiple word-embeddings makes little difference to the overall accuracy of the model, but particular embeddings perform differently on text that contains digits (i.e., 0-9) which can be leveraged by using multiple models to come to a conclusion on the score for a particular piece of text. Our results also show, that for our particular dataset trying to differentiate sentences into more than two categories might hurt the overall accuracy of the models, or at least not provide any substantial benefits compared to the binary classification scenario.**

I. Introduction

A. Background

Over the past few years, work in automated fact-checking has steadily increased and gained traction as our society begins to tackle the many challenges and issues that have arisen due to misinformation and an attack on the integrity of the news complex. Efforts in this area focusing on detecting "check-worthy" claims [1] and subsequently automating the process of fact-checking itself [2] [3] are gaining prominence. In order to make these efforts a reality, machine learning models have been trained to detect the check-worthiness of a claim. These methods have demonstrable short-comings in that they are susceptible to classifying sentences with digits as more "check-worthy" than those without. In essence, they are vulnerable to attacks of sorts that could be used to exploit them. The types of claims that we focus on and work with in this paper are extracted from political debates and center around the U.S. political complex. Websites such as PolitiFact[1] and more general news-websites regularly fact-check political statements made by entities in government. The pipeline for this work is essentially composed of gathering raw text from various sources that contains sentences spoken by a politician and then sifting through this data to identify claims that are important (i.e., in that they make claims that are important to the populace). Out of this ecosystem a tool, ClaimBuster[2], came to be which aims to simplify a fact-checkers work by scoring sentences according to how likely they are to be important. A subsequent effort to automate the process of fact-checking a claim itself is also in the works, and although we have a working system it is still in its infancy and will see improvement as our research progresses. In the fact-checking domain a model that is easily susceptible to failing, when not so uncommon requirements are met, can provide more work in the long run, as fact-checkers might need to sift through sentences that might have not even gone through a rudimentary filtering phase (i.e., if they rely on machine learning models heavily). Moreover, entities could easily exploit these weaknesses to mask their claims from or spam automated systems that make use of such models.

[1]http://www.politifact.com/
[2]http://idir-server2.uta.edu/claimbuster/

To tackle these issues, in this paper we discuss how to train a more robust and reliable model for this given task. Specifically, we propose a neural network model that takes advantage of pre-trained word embeddings available in the public domain, and an enhancing of the data-set used in training the previous machine learning models by doping it with troublesome sentences (i.e., sentences containing many numbers but are nonsensical). We show that by doing so we can mitigate some of the shortcomings of previous models and make progress towards better quantifying sentence salience (i.e., "check-worthiness"), which is of utmost importance in the fact-checking pipeline as it is the first step of the process.

B. Related Work

Our work follows and builds upon many others in the NLP realm that have explored either the construction of new embeddings or word modeling methods [4] [5] [6] [7]. We also make use of well-established neural network model baselines that served to guide us in constructing the network we used to train our models [8] [9] [10]. There have also been other efforts in fact-checking domain as well, FullFact [11] for one has ongoing efforts to automate fact-checking and claims to have made substantial progress in this respect. Other efforts include those to identify "check-worhty" tweets [12] and platforms for tracking misinformation [13]. Some of our collaberators at Duke also have substantial work done in this domain [14] as well as working systems[3]. Although, this field is somewhat of an emergent field that has come about due to the unforeseen interactions between the political and social-media complexes that envelop the daily lives of any person that lives in a nation with some sort of structured government. We see that both interact and influence the other in unique ways that lend themselves to outside entities introducing entropy into either of the systems, thereby warping perspectives and even changing narratives.

C. Outline

Through this study we first intend to find out how multiple word-embeddings affect a model. Secondly, how enhancing the data-set that is used by doping it with examples that are explicitly nonsensical, but serve to curve the the importance of particular features (i.e., numerical characters), affect a model's accuracy. Most approaches make use of one embedding and consider data that is labeled but still pertinent to their task (i.e., most training examples have a high chance of appearing naturally in the domain they were extracted from). We present a thorough exploration of the affects of using one or more word embeddings from four different sources, in different combinations and how adding nonsensical data that might not be considered due to it's extreme nature (i.e., how unlikely it is to appear naturally and therefore deemed unrelated) can notably increase a model's robustness. Finally, We also explore how trying to further compartmentalize data into sub-categories impacts model accuracy and whether it is worth further discriminating data at the loss of some accuracy. These three aspects are explored in tandem with testing two different loss functions.

We limited the variations in loss functions and other parameters of the network in general, as our goal was to study the aforementioned three aspects: word-embeddings, data doping, and number of categories/classes. Our choice to explore the two different loss functions stemmed from observations during preliminary test runs in which both loss functions explored in this paper produced models that showed promise. Since we did use different loss functions for different models, we present the results of how these models perform, so that is an emergent aspect of this study that came out as an aside from our main efforts.

II. Problem Statement, Neural Network Architecture, and Data Preparation

A. Problem Statement

Our ultimate goal is to determine a sentence's factual-salience by using either two or three categories, where one of the categories is where all the sentences in the dataset containing factual statements are mapped to. In the three-category model we categorize sentences as containing a factual statement (CFS), containing a factual but unimportant statement (UFS), or containing a non-factual statement (NFS). In the two-category format we classify sentences as either

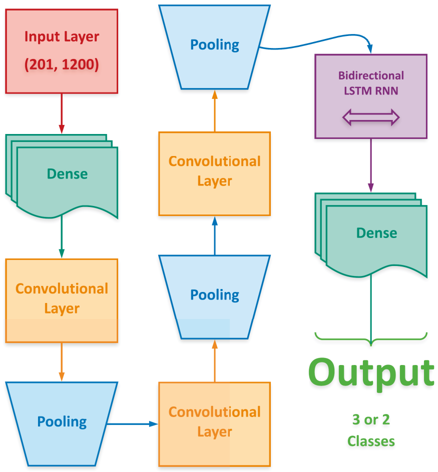


Fig. 1. Architecture of neural network used, where a input vector has a shape of $[201, 1200]$. The output can be a vector of length 3 or 2, depending on how many categories the input data was split into.

containing a factual statement (CFS) or containing a non-factual or unimportant statement (N-UFS). To this effect we take a hand-labeled dataset of 8231 sentences which have been labeled according to the three-category scenario. Our dataset is not available yet, but it is currently being prepared for public release. Interested parties can e-mail either of the authors. For the two-category scenario we simply concatenate the NFS and UFS classes into one. The overarching goal is to produce a model that can accurately identify salient-factual sentences versus non-factual sentences or unimportant factual sentences. The current baseline [1] achieves this to an extent, but has issues which have been mentioned previously. To that extent we not only want to produce a viable model, but one that addresses the issues present in the current baseline. To tackle these issues we chose to take a deep learning approach given that the current baseline is an SVM model, we wanted to explore how a deep learning framework would perform at this task and if it could indeed provide us with a better model.

B. Architecture

The architecture of the network we used is not too unconventional. As can be seen in Figure 1, we start with an input layer which takes a sentence that can be of up to 200 words in length (i.e., that sentence is converted to a vector of length 200 where each index is the mapped word-embedding for a given word). The longest sentence in our dataset was 167 words, so we added some padding, which was simply a vector filled with zeroes, to fill up empty indices in the vector which was necessary as the input needed to be standardized to a set length. We also added an extra vector, for a total of 201 feature vectors per sentence, that contained some features that were extracted from the words using Google's NLP service[4]. These features include the sentiment of the sentence (i.e., a value determining if the sentence has a negative, positive or neutral

[3]icheckuclaim.org

[4]https://cloud.google.com/natural-language/

sentiment overall), how many entities were found [5], and what parts of speech were found in text. Each word is described by a 300-1200 element long feature vector (since each embedding is 300 elements long) whose entries are comprised of word embedding values. Our dense layer takes the input and feeds this to a convolution layer with no activation function applied to it. From here we have 3 sets of convolutional and pooling layers, where each convolutional layer has a different kernel, we found that kernel sizes of 2, 3, and 4 worked best for our dataset by experimenting with several different kernel sizes, large and small. Our observations during this trial and error period showed that there was a degradation in performance when we had only 1-2 convolutional layers with small kernel sizes. Having many layers with large kernel sizes either displayed a degradation in performance or no significant increase in performance to merit the inclusion of more layers. The output of the convolutional layers is fed to a bi-directional LSTM network that feeds its output to a dense layer to generate the final output vector of the model. We performed experiments where we classified sentences into either two or three classes, so depending on the experiment the output vector of the last dense layer changed.

With our data we train two types of models, one whose loss function is categorical cross-entropy (CCE) defined in Eq. 1, and another which is the logarithm of the hyperbolic cosine of the prediction error (LHC) and is defined in Eq. 2, as defined in the Keras[6] library and documentation[7] [8] [9] [10]. The activation function used in the final output layer for the model is the softmax function in Eq. 3. An L2 regularizer (Eq. 5 is also applied to the kernel weights matrix of the output layer, and an L1 regularizer (Eq. 4) to the output layer itself. All of these methods are well established and based in literature [15]. We wanted to see how the two different approaches handle this task and how these results compare to the baseline predecessor model this work is based on. Our use of the two main losses, LHC and CCE, is based on our own observations training different models with several losses and concluding that these two were the best performing ones for our task. Likewise, with the L1 and L2 loss functions the lambda for these functions was chosen based on running several test runs and observing the the accuracy of the models fluctuated with each change. Finally, our choice of the Softmax function as the activation function is somewhat standard, but as with the other parameters we tested different activation functions and Softmax consistently produced the best performing models relative to the other activation functions.

[5]https://cloud.google.com/natural-language/docs/reference/rest/v1/Entity

[6]https://keras.io/losses/#logcosh

[7]https://github.com/keras-team/keras/blob/master/keras/losses.py#L48

[8]https://github.com/keras-team/keras/blob/master/keras/backend/tensorflow_backend.py

[9]https://www.tensorflow.org/api_docs/python/tf/nn/softplus

[10]https://www.tensorflow.org/api_docs/python/tf/log

$$\begin{aligned}\mathcal{L}(\theta) &= -\frac{1}{N}\sum_{i=1}^{N}\left[q_i \log(p_i) + (1-q_i)\log(1-p_i)\right] \\ \text{where } N &= \text{the number of samples} \\ p_i &= \text{the predicted value of the model} \\ q_i &= \text{the ground truth for the data}\end{aligned} \tag{1}$$

$$\begin{aligned}\mathcal{L}(\theta) &= (p-q) + \log_{10}(e^{-2\times(p-q)}+1) - \ln(2) \\ \text{where } p &= \text{the predicted value of the model} \\ q &= \text{the ground truth for the data}\end{aligned} \tag{2}$$

$$\begin{aligned}softmax(x) &= \frac{e^x}{\sum_{k=1}^{K} e^{x_k}} \\ \text{where } x &= \text{the tensor to be evaluated} \\ K &= \text{the number of dimensions of tensor x}\end{aligned} \tag{3}$$

$$\begin{aligned}Loss_{l1}(\lambda, x) &= \sum_{i=1}^{n} \lambda \times |x_i| \\ \text{where } \lambda &= \text{the weight of the regularization} \\ x &= \text{the tensor to be evaluated} \\ n &= \text{the number of dimensions of tensor x}\end{aligned} \tag{4}$$

$$\begin{aligned}Loss_{l2}(\lambda, x) &= \sum_{i=1}^{n} \lambda \times (x_i)^2 \\ \text{where } \lambda &= \text{the weight of the regularization} \\ x &= \text{the tensor to be evaluated} \\ n &= \text{the number of dimensions of tensor x}\end{aligned} \tag{5}$$

C. *Use of Multiple Word Embeddings*

We had initially tried to train models without using any pre-trained embeddings, but found that the resultant models were very poor. For this reason we began to explore the use of pre-trained embeddings and became intrigued by the question of whether using embeddings from more than one source would make a difference. We test four different word embeddings, which are Google [16] [17], the GloVe [18], Facebook [19], and dependency based word embeddings [20]. For a given word, each embedding source returns a vector with a length of 300. We simply concatenate each of the vectors from the different sources when testing multiple embeddings. If the word happens to be missing from the lexicon of a source, we simply ignore that source and place a dummy vector filled with 300 zeros.

We looked for several different embeddings to include in our study, but these four seemed to be representative of the state-of-the-art with respect to representational word vectors. Currently, count-based and predict models [21] are the most prominent (e.g., GloVe and word2vec (Google), respectively), and from here different embeddings try incorporating different contexts into the training process which produces embeddings with different weights.

D. Constructing Negative Examples

Apart from running experiments on the original dataset (which was used in the predecessor study to this paper) we used an enhanced version of this dataset, which we show results in an increased robustness to the model. The base (i.e., original) dataset contains about 8000 hand labeled sentences that have been given a label describing them as factual, factual but unimportant, or non-factual in nature. The collection was not part of the efforts in this paper, but rather work from the paper that the baseline model we will be comparing our results to came out of. We saw short-comings in the baseline model and wanted to try and address these by giving the neural network model some more negative examples that were exaggerated cases of what was giving the original SVM model trouble to try and see if it could improve the accuracy of the model. We had previously attempted to use the dataset as was with only one embedding but had only results that were about just as good as the baseline model. To enhance the dataset we wrote a simple script that generated 2000 random sequences of text that contained mostly numbers but also some words we thought might be used to exploit a model and confuse it. We labeled these sentences as non-factual and added them to our dataset processing them as all the other data, using the word embeddings mentioned above. Such training examples would have never been collected in the wild (i.e., by tuning into broadcasts and capturing closed caption text, scraping transcripts, etc.), also considering such extreme examples which can only be described as nonsensical data is not generally what one considers when training a model. Usually, we strive to obtain high-quality data that is representative of the problem task and contains good examples of the positive label. Yet, we can learn from mistakes or negative examples as much as we can learn from success, so we wanted to try and introduce that paradigm to model training process.

III. Experimental Evaluation

A. Results

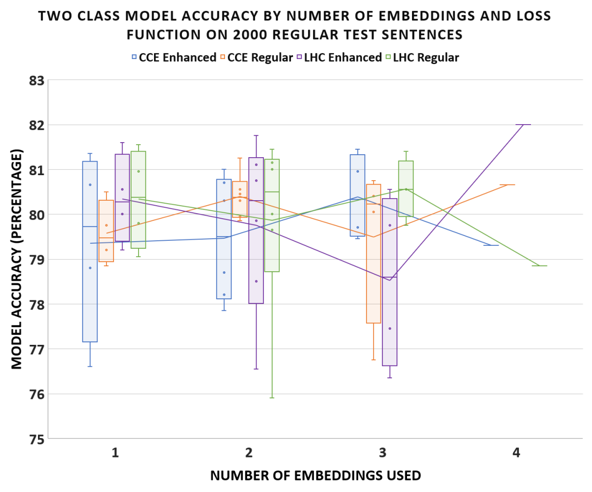
We trained the models on the 8231-sentence base dataset and the base dataset supplemented with 2000 nonsensical sentences mentioned in Section II-D. Each model is trained for 15 epochs with a batch size of 128. Following this training phase each model is saved to a file for testing and evaluation with an entirely disjoint test dataset.

We ran 120 experiments in total, of which half were on the base dataset and half were on the enhanced dataset. From there they were split between training three-class output models and two-class output models. For each of these specifications we used all the possible combinations of the 4 word embeddings (i.e., using 1, 2, 3, and 4 word embedding in every possible way). We then ran post-evaluation tests where we took 2000 regular sentences that had no part in the training process during the creation of the models, and 500 of the automatically generated nonsensical sentences, which contain mainly numbers, and were also not part of the training process of any of the models. Using these two test sets we tested all the models and computed metrics for these tests.

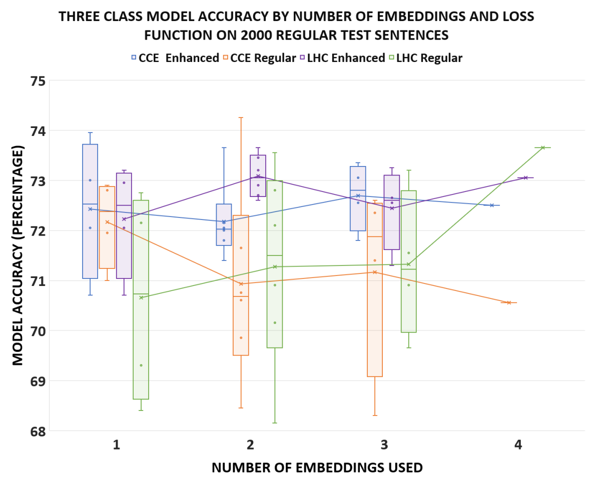
Interestingly, the introduction of more embeddings did not improve the models in any significant manner. Specifically, the plots in Figure 2 do not demonstrate a marked improvement in accuracy overall, as can be seen from the trend-lines in the box-whisker plots that go through the medians. Our intuition was that by introducing more foreign knowledge/context into the training process we would be able to see some improvements. But this was not the case. Moreover, as can be seen in the plots contained in Figure 2, the real discriminator in accuracy is a combination of the inclusion of the nonsensical sentences into the training data as well as whether the model is trained on two or three class labels. On that note, from the results we can also see that the two-class models consistently outperform their three class counterparts, although this might not be as telling since we are consolidating two classes into one to create the two class models.

$$
\begin{aligned}
AVG &= \frac{1}{\sum_{l \epsilon L} |\hat{y_l}|} \sum_{l \epsilon L} |\hat{y_l}| f(y_l, \hat{y_l}) \\
\text{where } L &= \text{the set of labels} \\
f &= \text{the evaluation metric (e.g., recall)} \\
y &= \text{the set of predictions for the data} \\
\hat{y} &= \text{the set of ground truths for the data}
\end{aligned}
\tag{6}
$$

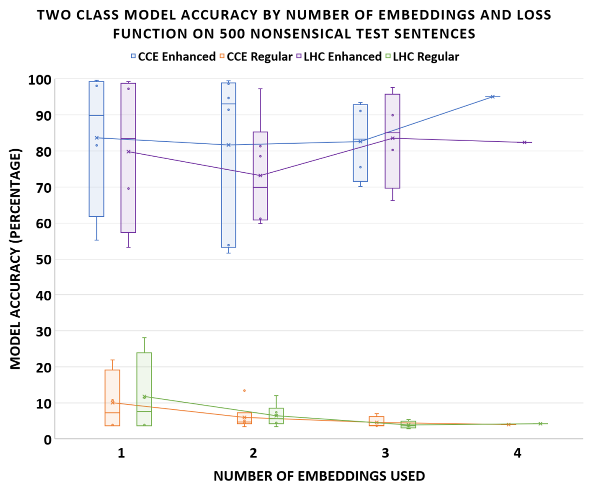
We calculated the average precision, recall, f1-score for two- and three-class models on the enhanced and regular datasets. We also calculated the weighted averages of the previous three measures, using Eq. (6), which took into account class imbalances by multiplying the measure over each class with the number of true instances in that class. In Table II, we only display the Recall since the 500 nonsensical sentences are all mapped to one class, NFS, and therefore measures like Precision and the F1-Score aren't really applicable to that scenario. For the sake of thoroughness we retrained the original SVM algorithm using the same settings as the one proposed in [1] with both the regular and enhanced datasets. Using the regular dataset we attained a model whose performance matched that of what was presented in [1], so in that respect we were able to verify we followed the their model creation pipeline correctly. We proceeded to run the SVM models through the 500 nonsensical test sentence dataset and the 2000 regular test sentence dataset. As can be seen from the last four rows in Table I and the last row in Table II, the SVM classifier trained on the regular dataset outperforms the two and three-class neural network models on the 500 nonsensical test sentence dataset, but this can be attributed to the fact that the metrics for the two and three-class models are averages across many models and do not exclude models that perform particularly poorly. On the 2000 sentences, the SVM algorithm trained on the enhanced dataset seems to classify most things as NFS. Hence its recall in that class is great but it really suffers in every other measure. The model trained on the regular dataset does not show much improvement. Even



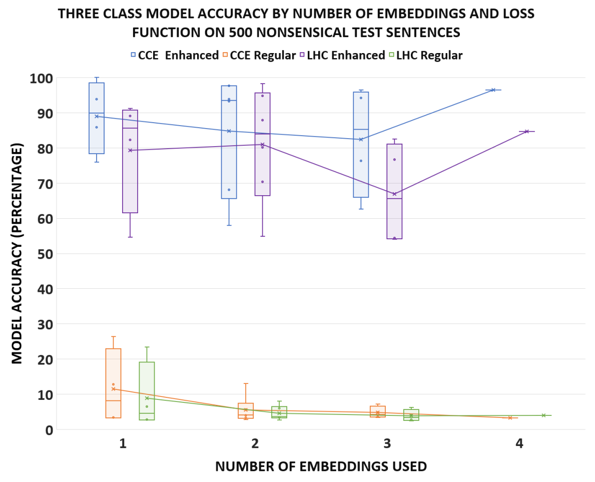
(a)



(b)



(c)



(d)

Fig. 2. Model accuracy is represented by a series of box-whisker plots each pertaining to a particular model that was trained on either the enhanced or regular data, and with CCE or LHC as the loss function.

though the precision for the NFS class is high, the recall is rather low, and even though the recall for the CFS class is high, the precision in that same class is rather low. These results seem to suggest that, when the SVM classifier is trained on the enhanced dataset it over-fits to the NFS class, while when it is trained on the regular dataset it over-fits to the CFS class.

It was surprising for us to see that the SVM model, trained on the regular dataset, performed much better on the 500 nonsensical test sentences than the other neural network models. This however, is a small upside given the rest of the results. Looking at Table II we can also confirm that the neural network models models trained on the extra 2000 nonsensical sentences did learn to avoid the pitfalls that plague their less well-rounded brothers. It should be noted, that with the inclusion of these extra 2000 nonsensical sentences we didn't see an overall drop in performance in the neural network models that were trained on them (when it comes to classifying regular sentences), so we can conclude that while they are successfully discriminating on number loaded sentences they still perform well on sentences with numbers that also happen to be in the CFS category. This is important, as we wouldn't want to over-fit the models to the point where they throw out sentences with numbers haphazardly as we see the SVM models do, if that were the case we'd end up in a similar but opposite scenario to the one we started in.

B. Discussion

In training our models we had also included an extra feature vector that contained information on entities found within a sentence, the sentiment of the entities, and the parts of speech. We found that the inclusion of this vector didn't affect the final model that much, but left it there since it was part of the pipeline that we had already established in creating our models. The paper which presented the original SVM algorithm also sought to leverage entity information and sentiment information, but was also unsuccessful in getting this extra data to have an impact on the final models. Perhaps how this data is being modeled is inadequate to provide

TABLE I
MODEL ACCURACY ON THE 2000 REGULAR SENTENCE TEST DATASET

		Enhanced Dataset			Regular Dataset		
		Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
Two Class Models	**NFS**	0.86	0.87	0.87	0.87	0.87	0.87
	CFS	0.60	0.58	0.58	0.60	0.59	0.59
	AVG	0.80	0.80	0.80	0.80	0.80	0.80
Three Class Models	**NFS**	0.78	0.86	0.83	0.80	0.84	0.82
	UFS	0.60	0.26	0.35	0.57	0.27	0.34
	CFS	0.61	0.60	0.60	0.68	0.65	0.61
	AVG	0.71	0.73	0.71	0.71	0.71	0.70
Original SVM Model	**NFS**	0.62	0.99	0.76	0.78	0.14	0.23
	UFS	0	0	0	0.15	0.13	0.14
	CFS	0.19	0.01	0.02	0.28	0.88	0.43
	AVG	0.42	0.61	0.47	0.57	0.32	0.27

NFS, CFS, and UFS refer to the classes a sentence can be labeled into, and AVG refers to the weighted average of all the three classes for a given metric (e.g., Recall). We use two different data-sets to construct and evaluate three different models (Two Class, Three Class, and the Original SVM Model). The data-sets are differentiated in that one has non-sensical sentences as negative training samples to help alleviate the issue being tackled in this paper.

TABLE II
MODEL ACCURACY ON THE 500 NONSENSICAL SENTENCE TEST DATASET

	Models Trained on the Enhanced Dataset	**Models Trained on the Regular Dataset**
Two Class Models N/UFS Recall	0.81	0.07
Three Class Models NFS Recall	0.81	0.06
SVM Model NFS Recall	1.0	0.45

enough context for it to be relevant to the model during the training phase. Although we found that the embedding source didn't really matter in our case, we only tried four different word embeddings (and their combinations) in trying to capture sentences with salient factual claims. There might be other word embeddings out there that capture different aspects of words that weren't already captured by the ones we tried, so that is something we will be on the lookout for.

Although, the performance of the models didn't really differ in a significant manner when different numbers of embeddings were used, we were interested in seeing if different models could be used to create a voting type scenario where the final outcome would be the aggregated response of two or more models. During this exploration, we noticed that models trained on the Dependency embeddings seemed to perform much better at handling sentences with digits (i.e., $0 - 9$) in them. These models also tended to be a bit to harsh on sentences without them. With this in mind we set up a comparison showing how these models scored 7 hand constructed sentences meant to showcase different features that affect each of the models in different ways. It seems, from these preliminary tests, shown on Table III where the scores represent how likely the given model thinks a sentence contains a salient factual statement, that the usage of two different models that handle text with digits differently might provide the best results overall. On Table III we also list the scores for the sentences that were given to them by the both SVM models (i.e., the one trained on the enhanced dataset and the one trained on the regular dataset). The SVM model trained on the enhanced dataset gives every sentence a low score, while the SVM model trained on the regular dataset is a bit more balanced in its results. However, it still does not out perform, in terms of differentiating sentences, the neural network model trained on the GloVe, Google, and Facebook embeddings let alone the aggregation of both neural network models presented in Table III as explained earlier.

We also have data we can leverage in future studies that gives further insight into the labels that are attached to each sentence. For a particular sentence 3 people might have voted for CFS, 0 for NFS, and 2 for UFS, while for another sentence the scores might be 5 CFS, 0 NFS, and 0 UFS. In the previous scenario the second sentence would have a larger shadow of doubt cast over it's classification than the second, by leveraging this information we might be able to capture more nuanced characteristics that make a sentence more salient, within a factual context, than another. In this scenario it might be more difficult to give a final classification to a sentence as the lines between the 3 classes would most likely become more blurred, but it is a viable branch to explore considering we have that information.

One of the hardest things to address in tackling this problem, is that factual-salience isn't an objective quality, but rather subjective and influenced heavily by the zeitgeist of a particular region. Of course, our efforts are heavily biased towards political factual-salience but there is substantial overlap in what makes a sentence salient in a political context and in a

TABLE III
SCORES ON SENTENCES BY FOUR DIFFERENT SELECT MODELS

Sentence	Model Scores			
	GloVe, Gooogle, Dep Embeddings	**GloVe, Google, Facebook Embeddings**	**SVM Model Enhanced Dataset**	**SVM Model Regular Dataset**
I ate apples.	0.06	0.02	0.10	0.20
I ate 2 apples.	0.06	0.02	0.10	0.20
I ate 500 apples.	0.06	0.02	0.10	0.23
Iraq does not have weapons of mass destruction.	0.05	0.76	0.16	0.33
Millions of illegal immigrants voted last year.	0.014	0.98	0.07	0.65
The U.S. allowed 320 million illegal immigrants to vote in the 2016 elections.	0.91	0.98	0.09	0.82
The 534 apples spread out across 3 tables had been left out for 1 day 9 hours and 23 minutes.	0.09	0.97	0.09	0.80

The scores given represent how likely a given model (discriminated by the word-embeddings or data it was trained on) thought a sentence contained a salient factual statement, thus making it check-worthy. A score of 1 is a highest score and thus means the model thought this was definitely a check-worthy sentence. A score of 0 means the model thought it was an unimportant sentence overall. We assess how each model weighs a sentence that has numbers and how different word-embeddings affect the score. We compare two neural network models and an SVM model trained on different data-sets.

more general context as well. Is it referencing something that affects a large group of people, places, things? Is it talking about well known and prominent entities? Is it trying to assert some interpretation of reality that is being presented as an objective truth? Most of these questions require contextual knowledge to answer which is hard to acquire in the correct context and furthermore parse.

IV. Conclusion

Although, the results we presented are not what we expected when we began our research there are some promising aspects to these. We saw that trying to differentiate our data into more categories than just two provided little to no benefit and that word-embeddings weren't that important in the sense that having them was important but how many wasn't, although we did find that some embeddings tend to produce models that have an affinity for sentences with digits in them. We also found that the inclusion of nonsensical data that was used to mitigate some issues with the base models did not negatively impact the accuracy of these enhanced models on regular sentences, which was key to the success of this study. We conclude that perhaps the best approach might be to use multiple models working together to effectively and accurately score sentences.

A. Future Work

In the near future we will integrate our best performing model into the platform that currently makes use of all work based on [1][11][12]. We also plan to leverage the voting information regarding the labels attached to each sentence, explained in the Discussion, to try and train better models. We have yet to exlpore how creating a voting system, wherein more than two models are used to score a sentence and then a label is given based on the majority score (i.e., given a particular threshold). There are many different paths we can explore from here, but those are the ones we are currently considering and assessing. The code used to develop these models and write this paper is available at https://github.com/damianj/ijcnn2018_factualsalience.

[11]idir-server2.uta.edu/claimbuster/
[12]idir-server2.uta.edu/factchecker/apidocs.html#!/apidocs/get_0_1_2_3

References

[1] N. Hassan, C. Li, and M. Tremayne, "Detecting check-worthy factual claims in presidential debates," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '15, 2015, pp. 1835–1838.

[2] N. Hassan, G. Zhang, F. Arslan, J. Caraballo, D. Jimenez, S. Gawsane, S. Hasan, M. Joseph, A. Kulkarni, A. K. Nayak, V. Sable, C. Li, and M. Tremayne, "Claimbuster: The first-ever end-to-end fact-checking system," *Proc. VLDB Endow.*, pp. 1945–1948, 2017.

[3] D. Jimenez, "Towards building an automated fact-checking system," in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD SRC '17, 2017, pp. 7–9.

[4] J. Cheng and D. Kartsaklis, "Syntax-aware multi-sense word embeddings for deep compositional models of meaning," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, 2015, pp. 1531–1542.

[5] M. T. Pilehvar, J. Camacho-Collados, R. Navigli, and N. Collier, "Towards a seamless integration of word senses into downstream nlp applications," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2017, pp. 1857–1869.

[6] X. Yu and N. T. Vu, "Character composition model with convolutional neural networks for dependency parsing on morphologically rich languages," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, 2017, pp. 672–678.

[7] A. Herbelot and M. Baroni, "High-risk learning: acquiring new word vectors from tiny data," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2017, pp. 304–309.

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

[9] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, 2015, pp. 2342–2350.

[10] B. Plank, A. Søgaard, and Y. Goldberg, "Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, 2016, pp. 412–418.

[11] FullFact.org, "The State of Automated Factchecking," *Full Fact*, August, 2016. https://fullfact.org/blog/2016/aug/automated-factchecking/.

[12] F. Arslan, “Detecting real-time check-worthy factual claims in tweets related to U.S. politics,” Master’s thesis, University of Texas at Arlington, 2015.

[13] C. Shao, G. L. Ciampaglia, A. Flammini, and F. Menczer, “Hoaxy: A Platform for Tracking Online Misinformation,” in *Proceedings of the 25th International Conference Companion on World Wide Web*, 2016.

[14] B. Walenz, J. Gao, E. Sonmez, Y. Tian, Y. Wen, C. Xu, B. Adair, and J. Yang, “Fact checking congressional voting claims,” in *Proceedings of the 2016 Computation+Journalism Symposium*, 2016.

[15] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *CoRR*, vol. abs/1301.3781, 2013.

[17] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS’13, 2013, pp. 3111–3119.

[18] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *EMNLP*, 2014.

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

[20] O. Levy and Y. Goldberg, “Dependency-based word embeddings,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, 2014, pp. 302–308.

[21] M. Baroni, G. Dinu, and G. Kruszewski, “Don’t count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2014, pp. 238–247.