Word Pair Convolutional Model for Happy Moment Classification

Michael Saxon*[1,2], Samarth Bhandari*[1,3], Lewis Ruskin[1,3], and Gabrielle Honda[1,4]

[1]The Luminosity Lab, Office of Knowledge Enterprise Development
[2]School of Electrical, Computer, and Energy Engineering
[3]School of Computing, Informatics, and Decision Systems Engineering
[4]College of Integrative Sciences and Arts
Arizona State University, Tempe, AZ 85281

Abstract. We propose the Word Pair Convolutional Model (WoPCoM) for the CL-Aff 19 shared task at the AAAI-19 Workshop on Affective Content Analysis. The challenge is the classification of speaker-described happy moments as social events and/or activities in which they have agency. WoPCoM leverages the regular structure of language on an architectural level in a way that recurrent models cannot easily answer, by learning convolutional word pair features that capture the important intra- and inter-phrasal relationships in a sentence. It performs with an average accuracy of 91.45% predicting the social label and 86.49% predicting the agency label assessed on a 10-fold cross validation. This represents a performance improvement of 0.92% for predicting the social label, but only 0.04% on predicting the agency label, over a simpler deep LSTM baseline. In spite of similar performance on these metrics, WoPCoM demonstrates desirable results when other metrics such as training time, model-intermediate class separation, and overfitting propensity are considered, warranting further study.

Keywords: natural language processing · dilated convolutional neural network · social interaction · word vectors · word embeddings · semantic-syntactic features · sentiment analysis

1 Introduction

1.1 CL-Aff Shared Task

This paper will address the challenge put forth in the CL-Aff shared task [6] at the AAAI-19 Workshop On Affective Content Analysis. The task involves rating a happy moment for speaker agency and social interaction given only a sentence describing the moment.

HappyDB is a corpus of 100,000 crowd-sourced happy moments. Workers on Amazon Mechanical Turk were tasked with recalling and writing sentences

* Equal contribution

describing happy moments from three "recollection periods" since they have occurred, the past 24 hours, the past week, and the past month. [1] From this corpus a training set of 10,000 sentences, each labelled with speaker agency and social attributes, was produced. The social label is assessed as "yes" if the moment in question directly involved people other than the speaker, and "no" otherwise. The agency label is rated "yes" if the speaker had direct control over the action described in the moment, and "no" otherwise. For example, the moment "My boyfriend bought me flowers" would be rated as "yes" for social and "no" for agency, whereas "I took my dog for a walk" would be "no" for social and "yes" for agency.

1.2 Related Work

As this is a new shared task, previous approaches to this specific problem do not exist. Because we have selected a deep learning-based approach to the task, we consider neural networks for sentence-level semantic classification, and the word embedding approaches that underpin most methods in that domain related work.

Word embeddings are a powerful tool for NLP tasks due to their capacity to capture both semantic and syntactic data without a need for feature engineering [9]. Despite shortcomings like susceptibility to common unigrams [4] and no mechanism for representing syntactic relationships, a bag of words approach to creating word vectors establishes a good theoretical baseline for identifying and addressing various weaknesses in word encoding models.

Embeddings from Language Models (ELMo), 1024-dimensional word vectors assessed using a bidirectional LSTM applied across the characters in a sentence, are the current state of the art in latent word vector representation [13]. ELMo embeddings have been demonstrated to bring peak performance on a variety of downstream tasks against other universal word embedders such as Word2Vec and GloVe [12]. We selected pre-trained ELMo embeddings for our approach because of this past record of performance on downstream tasks, and their two most useful properties: the learning of meaningful sub-word units [2] by virtue of their character-based processing, and their capture of richer semantic context at the word level from their use of the full sentence context in generating constituent word vectors. While Google's Universal Sentence Encoder has been demonstrated to perform better than ELMo on semantic relatedness and textual similarity tasks [12], we require the granularity provided by word-level rather than sentence-level embeddings for our approach to the task.

1.3 Design Rationale

Our neural network design arose by first considering how we would implement a feature engineering approach, and then determining how a neural network could learn similar features to the ones we would have hand-engineered.

The "agency-ness" and "social-ness" of the vast majority of sentences we considered at this stage hinged on a few critical features. For example, the "I

walked" in "I walked my dog to the park" is critical for understanding the agency of the speaker. Perhaps this pair of words could be captured by a two-word filter evaluating personal pronouns to the left of active verbs. Through the consideration of these toy examples our hypothetical feature engineering approach began to take shape.

By approximating distributions of certain semantic-syntactic concepts such as personal nouns, social verbs, and action verbs in the embedding space the probability a word belongs to such classes can be estimated. The word count distance between two words in a sentence can be roughly correlated with their syntactic relationship. Coupled with semantic-syntactic information about the individual words themselves features correlated with sentence meaning would arise.

Rather than hand engineer these word probability distributions we would design the initial layers of the neural network to learn them. Rather than hand engineer the two-word filters we would use convolutional layers to learn them. Inspired by the dilated convolution approach employed by WaveNet [10] to generate audio samples by capturing various levels of sample stride, we decided to create a filterbank of varying dilation factor size-two convolutional filters to capture various inter- and intra-phrasal relationships. We were confident in this approach because it would allow us to constrain the solution space and build in insights from the structure of language that are not present in the far more general sequence learning recurrent methodologies such as LSTM-based [5] models.

English is a strongly head-initial language with a subject-verb-object word order. The word determining the syntactic category of an English phrase generally precedes its complements, leading to the formation of right-branching grammatical structures. Nouns generally form constituent noun phrases of the verbs they follow [14]. Taken together, this information about the regular structure of English sentences offers an opportunity to design network architectures that capture the important interplay between key pairs of words in a sentence. In other words, because we can rely on the direction between pairs of words to matter in many of these examples, a filter-based conceptualization of how to process sentences for semantics makes sense.

These observations justified the design of the Word Pair Convolutional Model (WoPCoM), hinging on the importance of pairs of words and the flexibility of neural networks to form a prediction.

2 Approach

Almost all of the classes we considered boiled down to a prototypical feature $f(x(n), k) = P(x(n) \in S_1)P(x(n+k) \in S_2)$ for input sentence x of words $x(n)$, word count separation k and word classes S_1 and S_2. For example, one possible feature for testing speaker agency in a sentence might consider S_1 to be the set of personal nouns and S_2 to be the set of active verbs for $k = 1$, which would capture subclauses common to active sentences such as "I took" in the example sentence above, or "we went," "I made," etc. Similar features for

labelling social interactions can be envisioned. Through the convolution of these features across the sentences, the labels could be ascertained. While modelling the probability distributions of the various word sets and considering all the important features by hand is not feasible, we hypothesized that a convolutional neural network (CNN) could solve both of these problems at once, implicitly learning the distributions of word classes and choosing which classes to select for simultaneously during the learning process. What follows are the insights from our model design process through the models we considered.

We chose to use ELMo vectors as the input feature for all models. All models were implemented using PyTorch [11] and AllenNLP [3], all of our code is available at `https://github.com/luminositylab/CL-AFF-ST`.

2.1 Baseline Recurrent Model

We first considered a naïve model employing a long short-term memory (LSTM) layer with variable hidden size taking the sentence set of ELMo embeddings as input. The final hidden state of the LSTM was processed through a single fully-connected layer with output size 2, and then a sigmoid function to map all possible outputs to the $(0, 1)$ probability range, representing the probability of membership to the two classes.

2.2 Word Pair Convolutional Model

In our prototypical feature engineering formulation of the task we considered a feature extractor convolving across a vector of class membership probabilities for each word in the sentence to extract various intermediate word pair class attributes that could be used in higher-level classification. However, due to the feasibility issues discussed above and the lack of a ground truth objective to train word class probabilities with, we opted for a looser type of pre-CNN processing, a set of linear layers separated by ReLU operations with no strict class probability constraint at the output. We do this by taking the ELMo-evaluated sentence set of word vectors through a $1024 \times N$ linear layer, with N corresponding to the number of word attributes we want the WoPCoM to learn.

Two-vector convolutional filters of varying stride are then applied across the sentence set of size N word attribute vectors to evaluate word pair semantic-syntactic features. Through the varying dilation of the filters different types of syntactic relationships can be captured, allowing for intra- and inter-phrasal relationships to be assessed. We implemented our feature extractors as a set of five 1-dimensional $2 \times N \times NC$ convolutional layers with dilation factors varying from 1 to 5. The output dimension NC (number of classes) is configurable to allow for evaluating different quantities of word pair relationships, and can be considered analogous to the hidden state size in the LSTM implementation.

To "pool" the five sentence-length convolution outputs we feed them through an LSTM with hidden size NC and take the final $NC \times 1$ hidden state as the "pooled" output of the five filter sets. Those vectors are then concatenated into one vector, and fed through a linear $5NC \times 2$ layer to map these feature outputs

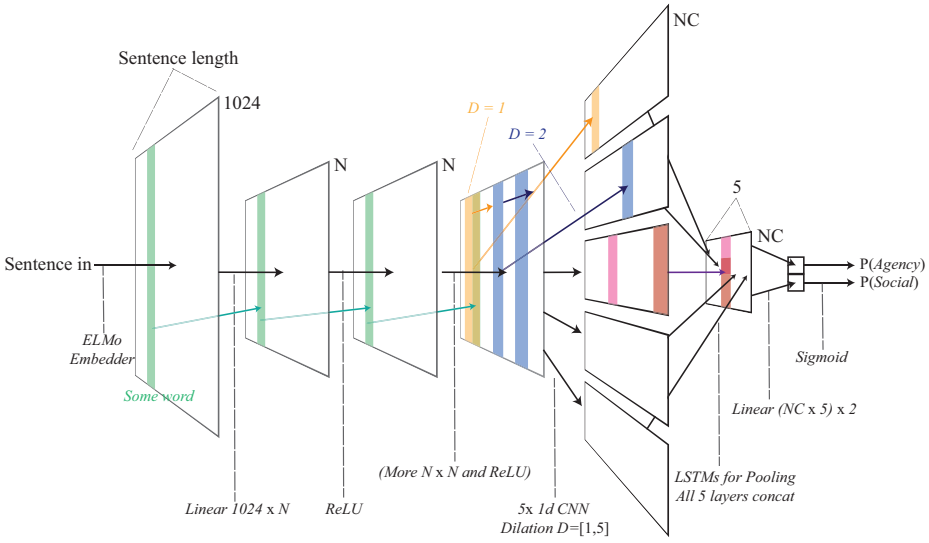

Fig. 1. A diagram of the layer outputs of the WoPCoM model for a single sentence.

to the two classes. Finally, these feature outputs are evaluated as probabilities through the sigmoid function. Figure 1 depicts the WoPCoM model by showing selected layer outputs in the processing of a single sentence batch.

2.3 Hyperparameters

Baseline For the baseline recurrent model a hidden representation size of 25 was used.

WoPCoM For the final implementation the word class count N was set to 100, and the convolutional filter set output size NC was set to 50.

2.4 Training

The Adam optimizer [7] was used with a learning rate of 0.0001.

The models were trained with random batches of 50 same-length sentences to minimize necessary input padding. A patience factor of 10 was used to allow variable-length training, once 10 epochs pass without a drop in validation loss, training is halted. Mean-squared error was used as the loss metric.

Class labels were tokenized as 0 for "yes" and 1 for "no." This means that the model is really learning negative probabilities (probability that the sentence is not social/agency) but this has no bearing on final accuracy. To compute accuracy, F1, and AUC the output probabilities are rounded to 0 or 1.

3 Results and Discussion

We evaluated WoPCoM and our baseline model using 10-fold cross validation, achieving the results in Table 1. Results that have over a 0.5% improvement from the baseline are in **bold**.

	Social			Agency		
Model	Accuracy	F1	AUC	Accuracy	F1	AUC
LSTM Baseline	90.58%	91.31%	95.72%	86.22%	90.68%	91.81%
WoPCoM	**91.45**%	**91.90**%	96.08%	86.49%	90.82%	91.93%

Table 1. A diagram of the layer outputs of the WoPCoM architecture for a single sentence.

To illustrate the general separation of classes performed by WoPCoM t-SNE projection [8] was employed to generate Figure 2. Note that the "Social Only" and "Agency Only" classes both overlap significantly with the most heavily "Social & Agency" region, while overlapping significantly less with each other's regions.

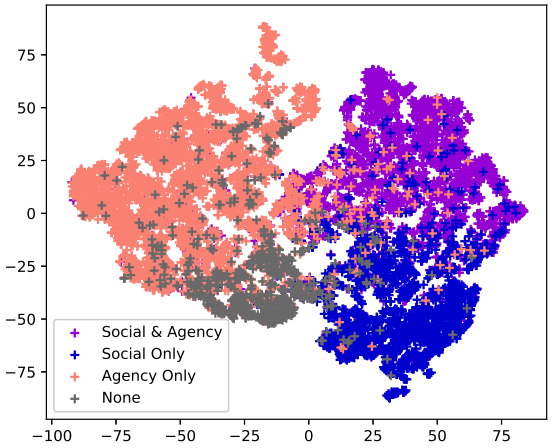

Fig. 2. t-SNE projection of the $5 \times NC$ post-pooling filter set outputs for each sentence.

WoPCoM shows a modest improvement over the LSTM baseline on the social classification task, but the difference between its performance and the LSTM performance on the agency classification task is meager. This disparity could have meaningful implications about the underlying data, warranting further study.

Figures 3 and 4 depict the progression of validation and test accuracy across epochs for WoPCoM and the LSTM baseline. They demonstrate that despite having similar numbers of internal parameters, WoPCoM has a tendency to resist the kind of dramatic overfitting that takes place with the LSTM baseline. It is important to note that these accuracy figures only assess a given sentence as accurate if both social and agency are properly labelled, meaning the individual accuracy numbers being produced by these models are actually higher when considering each individual task.

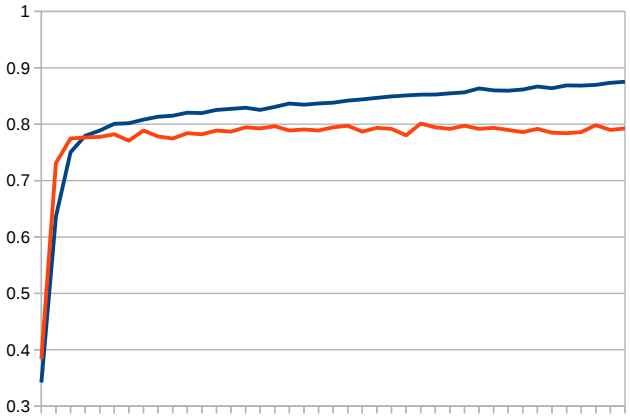

Fig. 3. Validation (red) and training (blue) accuracy for WoPCoM across epochs.

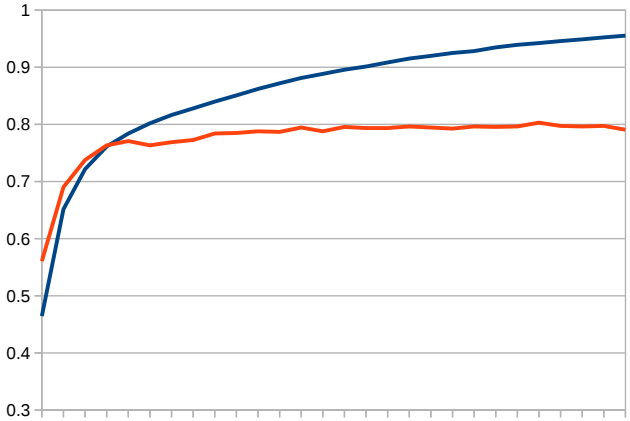

Fig. 4. Validation (red) and training (blue) accuracy for the LSTM baseline across epochs.

4 Conclusions and Future Work

Through the process of training and testing these models many times we have come up with some informal observations in addition to the concrete. Some confirm basic concepts in machine learning, such as the efficacy of deeper architectures at fitting to the complex underlying distributions at the cost of overfitting. We began paying attention to which models quickly fit to high accuracy across relatively few epochs, and which models maintained a relatively narrow gulf between the training and validation loss across many epochs.

One potential advantage we found to WoPCoM as opposed to the LSTM is that it tends to resist overfitting. The training and test loss both saturate around the same epoch, and there does not come a point where the overfitting pattern of simultaneously decreasing training loss and increasing validation loss takes place. This might be a result of the constraints on the solution space structure described briefly above. We are interested in applying a more detailed analysis to this phenomenon.

We plan to complete future work to look more rigorously to validate parameters of our approach that were chosen almost arbitrarily, such as the choice of two-word convolution filters, and dilation factors up to five, against similar approaches leveraging three- or four-word filters and larger dilation factors, as well as pit WoPCoM or models designed with a similar philosophy against more radically different architectures on more established tasks.

Currently we are using pretrained ELMo embeddings, we suspect that for task/dataset-specific problems such as this shared task word embeddings learned from the unlabeled data directly could improve performance. An unsupervised approach employing an autoencoder that shares the sentence-feature compression architecture of WoPCoM could potentially be adapted to improve performance as well.

We would also like to investigate the utility of the WoPCoM architecture we've developed to applications with scarcity of training data and build on this architecture to fit on such problems better.

5 Acknowledgements

We would like to thank Dr. Hemanth Venkateswara for his guidance, particularly in shooting down our worst ideas early, so we never had to discover how bad they were ourselves.

References

1. Asai, A., Evensen, S., Golshan, B., Halevy, A., Li, V., Lopatenko, A., Stepanov, D., Suhara, Y., Tan, W.C., Xu, Y.: HappyDB: A corpus of 100,000 crowdsourced happy moments. In: Proceedings of LREC 2018. European Language Resources Association (ELRA), Miyazaki, Japan (May 2018)

2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606 (2016)
3. Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N.F., Peters, M., Schmitz, M., Zettlemoyer, L.S.: AllenNLP: A deep semantic natural language processing platform. arXiv preprint arXiv:1803.07640 (2017)
4. Hacohen-Kerner, Y., Rosenfeld, A., Sabag, A., Tzidkani, M.: Topic-based classification through unigram unmasking. Procedia Computer Science **126**, 69–76 (2018)
5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735, https://doi.org/10.1162/neco.1997.9.8.1735
6. Jaidka, K., Mumick, S., Chhaya, N., Ungar, L.: The CL-Aff Happiness Shared Task: Results and Key Insights. In: Proceedings of the 2nd Workshop on Affective Content Analysis @ AAAI (AffCon2019). Honolulu, Hawaii (January 2019)
7. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
8. Maaten, L., Hinton, G.: Visualizing high-dimensional data using t-SNE. Journal of Machine Learning Research **9**, 2579–2605 (01 2008)
9. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 26, pp. 3111–3119. Curran Associates, Inc. (2013)
10. van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K.: Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499 (2016)
11. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch (2017)
12. Perone, C.S., Silveira, R., Paula, T.S.: Evaluation of sentence embeddings in downstream and linguistic probing tasks. arXiv preprint arXiv:1806.06259 (2018)
13. Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 2227–2237. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/N18-1202, http://aclweb.org/anthology/N18-1202
14. Radford, A.: English Syntax: An Introduction. Cambridge University Press (2004)