Neural Metaphor Detection with GloVe Embeddings and Bidirectional-LSTMs

Nick Saulnier
Khoury College of Computer Science, MSc Student
Northeastern University
Seattle, Washington

***Abstract*—I present an analysis of current state of the art techniques for detecting metaphorical language usage using GloVe embeddings and bidirectional-LSTM models. A selection of literal and non-literal examples of verb usage, taken from the hand-annotated VU Amsterdam Metaphor Corpus, is used as training data. I demonstrate that test data accuracy scores near 0.70 can be achieved in the literal/non-literal verb classification task with relatively simple model architectures and minimal text preparation. Planned extensions to the baseline model include adding input and embedding layers for ELMo word vectors and POS tags, and experimenting with connecting the LSTM layer to various attention layer configurations. Model evaluation extensions include identification and analysis of hard-to-classify verbs and their metaphor type.**

I. INTRODUCTION

Metaphorical language pervades all modes of discourse, giving humans a means of expressing complex, abstract topics in forms that are accessible and easy to understand. As such, the quality of many natural language processing systems—such as machine translation and question and answering systems—is directly related to the ability of machines to detect and interpret figurative language usage. The goal of this paper is to demonstrate how current state-of-the-art deep learning models perform metaphorical language detection, and to explore the implementation and effectiveness of various techniques using the Keras API.

The Vrije University Amsterdam Metaphor corpus (VUAMC) [1] is the most widely used dataset for training metaphor detection systems. The corpus has annotations for three types of metaphor: indirect, direct, and implicit. Most of the metaphorical usages in the corpus are indirect metaphors, which are identifiable by contrasting a contextual and more basic meaning of the target word. The following example is cited on the VUAMC website as an example indirect metaphorical usage of the word 'valuable': "professional religious education teachers like Marjorie B Clark… are doing *valuable* work in many secondary schools" [2]. In this context, the word 'valuable' is referring not to its basic, literal meaning—a lot of money—but to the concept of value as it relates to education. With direct metaphors, no contrast is made between a contextual and basic meaning; metaphorical usage is determined by how the word is used, and is often signaled by a word such as 'like'. The following is an example of a direct metaphor: "… he's like a ferret" [2]. Implicit metaphors are indicated by semantic and/or grammatical linkage between non-literal usage and a referent that is not immediately identifiable as a direct or indirect metaphor. In the following example, the word 'step' forms an implicit metaphor when referred to by the word 'it': "Naturally, to embark on such as step is not necessarily to succeed in realizing it" [2]. The VUAMC corpus contains text from academic works, newspapers, conversation, and fiction. Annotations were made using the MIPVU metaphor identification protocol [2], with each selected sample having an inter-annotator reliability score over 0.8.

Automatic metaphor identification is often divided into two task formulations: sequence labeling and classification [3]. For the sequence labeling task, each sample sentence is accompanied with a sequence of binary labels indicating the metaphoricity of each word. For the classification task, each sample sentence has a target verb and a binary label indicating the verb's non-literal or literal usage. I chose to work on the

classification task, using a subset of the VUAMC corpus consisting of 23,112 literal and non-literal usages of 2,298 unique verbs. Each sample consists of a single sentence, a target verb, the index of the target verb in the sentence, and a binary label. Using a simple bidirectional-LSTM model with a single hidden layer and an 11-gram text preprocessing technique described by Brooks and Youseef [4], I was able to achieve an accuracy score of 0.69 on the test set.

II. Related Work

As noted by Stowe et al., while syntactic structure alone can indicate non-literal usage, "most computational approaches to metaphor eschew syntax for more semantic features" [5]. Word vector representations, in particular GloVe (global vectors for word representation) [6] and ELMo (deep contextualized word representations) [7] embeddings, have become standard tools for boosting model performance on the metaphor identification tasks. These embeddings capture not only semantic characteristics of word usage across multiple linguistic contexts, but also model aspects of syntax [7]. In one of the most cited investigations of deep word embeddings and their application to metaphor detection models, Gao et al. were able to improve state-of-the-art performance on the VUAMC sequence labeling task by 7.5 F1, and 2.5 F1 on the verb classification dataset, by concatenating a 50d locally generated index embedding with 1024d ELMo embeddings and 300d GloVe embeddings for each input token [3]. Their model was a "relatively standard BiLSTM model… which operate[s] on complete sentences" [3]. Pramanick, Gupta, and Mitra built a bidirectional-LSTM model augmented with a Conditional Random Field (CRF) layer above the LSTM layer for the VUAMC sequence labeling task [8]. They used word2vec Continuous Bag of Words (CBOW) embeddings with a window size of 8 words in various feature configurations that include: token embeddings, token lemmas, part-of-speech, and lemma-token relational markers. Their results indicate that the models performed best using only token word embeddings as input. Brooks and Youssef built two bidirectional-LSTM models—a many-to-one model where the outputs of the "forward and backward LSTM cells in the attention layer are concatenated only at the output" [4] for the target word, and a many-many model where the weights are updated in relation to the model's performance on both the target word and the surrounding context words—for the VUAMC verb classification task. Similar to Gao et al., they concatenated 300d GloVe and 1024d ELMo embeddings for each input token. They used an 11-gram with the target verb at the center of the 11-gram to represent each sample, padding the 11-gram as necessary. By creating ensembles of these two architectures, Brooks and Youssef were able to achieve F1 scores of over 0.73 for verb classification.

[To Do: mention models that use concreteness scoring, and multi-head models that work on sense disambiguation.]

III. Approach—Data Preprocessing and model architecture Experiments

To construct a baseline model that could be extended with a variety of enhancements, I decided to use the preprocessing techniques outlined by Gao et al. and Brooks and Youssef. I began by downloading 1024d ELMo embeddings and 300d GloVe embeddings, using the 840 billion token, 2.2 million vocab GloVe files. I then created used a spaCy part-of-speech tagger to generate POS tags for each sample sentence. For each sentence, I created an 11-gram window with the target verb at the center, padding the windows with UNK symbols to the left and right if the target verb is located at the beginning or end of the sample sentence. I then replaced each token in the sample windows with a 1341d vector created by concatenating GloVe, ELMo, and one-hot POS vectors. I then created a simple bidirectional-LSTM architecture with a single hidden layer with 300 nodes, and input layer, and an output layer that uses applies a softmax activation. I discovered that inputting the windows in this format using Keras is extremely difficult and fraught with tensor dimension incompatibility errors between the input, embedding, hidden, and output layers. Upon further investigation, I found that the Gao et al. model is publicly available for use on Github; it

uses PyTorch, and I was unable to translate their input scheme to my Keras model. The closest I came was using TensorFlow ragged tensors, but that functionality is not well-supported and repeatedly crashed my notebook.

Taking what I learned about Keras from these experiments, I decided to scale back my input data and create a single GloVe embedding layer for the model. With this setup, outlined by Jason Brownlee in his Machine Learning Mastery blog post, an embedding weight matrix for GloVe vectors is created using the 11-gram window tokens as a vocabulary. The model's embedding layer weights are initialized with the GloVe weight matrix, its vocab size is set to the size of the window samples vocabulary, and the embedding dimension is set to match the GloVe vector dimension [9]. I used what seemed like sensible hyperparameters when creating the model. The bidirectional-LSTM layer has 300 nodes, a dropout of 0.3, and a recurrent dropout of 0.2. I set my learning rate to 0.001, used a batch size of 128, and an epoch count of 5. I used the rmsprop optimizer and a binary cross entropy loss function, optimizing for accuracy. Because the dataset is limited in size, I thought it best to train the model using 5-fold cross validation. I set aside 1/3 of the dataset to use as a test set. The labels are input as one hot vectors of size 2.

IV. Preliminary Results and Model Evaluation

In my preliminary experiments, the model trains quickly: at the end of the second fold, training accuracy reaches 0.86 and validation accuracy is around 0.8. At the end of the fifth fold, the training accuracy is above 0.97 and validation accuracy is not far behind at just above 0.96. When the model is run on the test set, an accuracy score of 0.69 is achieved. In future runs, I plan to evaluate the model using F1 scoring, precision, and recall so that I can better compare its performance against models cited from the literature. I would also like to more closely examine the results, looking at the types of metaphors—indirect, direct, and implicit—and their misclassification rates. If there's time, I will test my model on the TroFi dataset; detecting novel metaphors is a big challenge that still has quite a bit of work for improvement. It would be interesting to see if I can boost performance on the TroFi dataset with a model trained on the VUAMC dataset with only hyperparameter tuning and architectural tweaks.

V. Future Work

Time allowing, I plan on creating multiple versions of this model enhanced with multiple input layers and attention mechanisms. Research indicates that adding greater amounts of input data—concatenation of multiple deep word embeddings, POS tag vectors, named entity tags, etc.—sometimes leads to increased model performance [3][4], and sometimes does not [8]. I am interested to discover if these observations are dependent to some extent on model architecture. The Gao et al. PyTorch model worked well with embedding concatenation for each input token, but as mentioned above, this technique does not scale well to the Keras API. Adding multiple input layers and connecting them to a larger model using the Keras API may also be an interesting problem.

VI. Conclusion

While my baseline model implementation is fairly straightforward and simple, I learned how to transfer pre-trained embedding weights to Keras embedding layers and map them to a vocabulary. This style of using deep word embeddings contrasts with methods I've read about in the literature, particularly the methods described by Gao et al. and Brooks and Youssef, and is actually a little exciting. Once I created the GloVe embedding layer, I realized that it would be easy to create an ELMo embedding layer and additional input layers for POS and named entity tags. Using pretrained embeddings as separate, multiple inputs opens up new model architecture possibilities and methods of connecting multiple input layers that seem, at least conceptually, more nuanced than the crude concatenation methods described above. Brooks and Youssef mention that their experiments indicate that 11-gram windows resulted in the best performance. I am also interested in testing the 11-gram method against full sample sequences.

References

[1] Steen et al., "A Method for linguistic metaphor identification. From MIP to MIPVU," John Benjamins, Amsterdam, 2010.

[2] Netherland Organization for Scientific Research. [Online] Available: http://www.vismet.org/metcor/documentation/home.html [Accessed Nov. 27, 2021].

[3] Gao et al., "Neural Metaphor Detection in Context," Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Oct.-Nov. 2018. [Online] Available: https://aclanthology.org/D18-1060/ [Accessed Nov. 27, 2021].

[4] Jennifer Brooks and Abdou Youssef, "Metaphor Detection using Ensembles of Bidirectional Recurrent Neural Networks," Proceedings of the Second Workshop on Figurative Language Processing, July 2020. [Online] Available: https://aclanthology.org/2020.figlang-1.33/ [Accessed Nov. 27, 2021].

[5] Stowe et al., "Linguistic Analysis Improves Neural Metaphor Detection," Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL), Nov. 2019. [Online] Available: https://aclanthology.org/K19-1034/ [Accessed Nov. 27, 2021].

[6] Jeffrey Pennington, Richard Socher, and Christopher D. Manning, "GloVe: Global Vectors for Word Representation," Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Oct. 2014. [Online] Available: https://aclanthology.org/D14-1162/ [Accessed Nov. 27, 2021].

[7] Peters et al., "Deep Contextualized Word Representations," Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1, June 2018. [Online] Available: https://aclanthology.org/N18-1202/ [Accessed Nov. 27, 2021].

[8] Malay Pramanick, Ashim Gupta, and Pabitra Mitra, "An LSTM-CRF Based Approach to Token-Level Metaphor Detection," Proceedings of the Workshop on Figurative Language Processing, June 2018. [Online] Available: https://aclanthology.org/W18-0908/ [Accessed Nov. 27, 2021].

[9] Jason Brownlee, "How to Use Word Embeddings Layers for Deep Learning with Keras," Machine Learning Mastery blog, Feb. 2, 2021. [Online] Available: https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/ [Accessed Nov. 26, 2021].

[10] Julia Birke, "TroFi Example Base", July 2005. [Online] Available: TroFi Metaphor Dataset - SFU Natural Language Laboratory [Accessed Nov. 27, 2021].