Published in DAIR.AI · Follow

elvis · Follow
Apr 30, 2018 · 4 min read

Detecting Sarcasm with Deep Convolutional Neural Networks

Overview
This paper addresses a key NLP problem known as *sarcasm detection* using a combination of models based on convolutional neural networks (CNNs). Detection of sarcasm is important in other areas such as affective computing and sentiment analysis because such expressions can flip the polarity of a sentence.

Example
Sarcasm can be considered as expressing a *bitter gibe* or *taunt*. Examples include statements such as "Is it time for your medication or mine?" and "I work 40 hours a week to be this poor". (Find more fun examples here)

Challenges
To understand and detect sarcasm it is important to understand the *facts* related to an event. This allows for *detection of contradiction* between the objective polarity (usually negative) and the *sarcastic characteristics* conveyed by the author (usually positive).

Consider the example, "I love the pain of breakup", it is difficult to extract the knowledge needed to detect if there is sarcasm in this statement. In the example, "I love the pain" provides knowledge of the sentiment expressed by the author (in this case positive), and "breakup" describes a contradicting sentiment (that of negative).

Other challenges that exist in understanding sarcastic statements is the reference to multiple events and the need to extract a large amount of facts, commonsense knowledge, anaphora resolution, and logical reasoning. The authors avoid automatic feature extraction and rely on CNNs to automatically learn features from a sarcasm dataset.

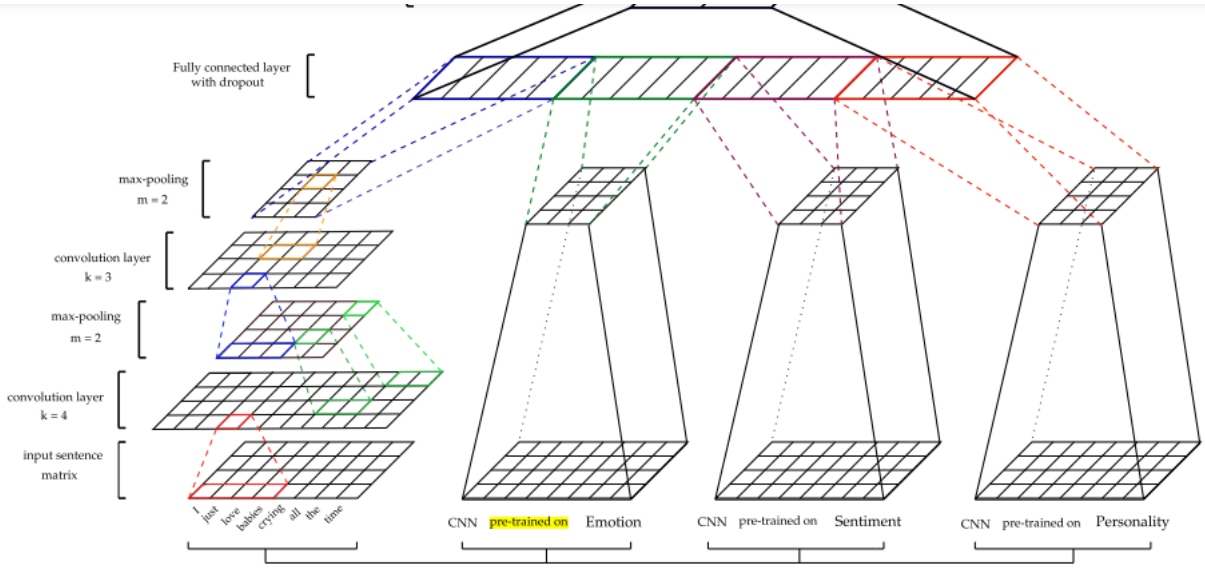
Contributions

- Apply deep learning to sarcasm detection
- Leverage user profiling, emotion, and sentiment features for sarcasm detection
- Apply pre-trained models for automatic feature extraction

Model
Sentiment shifting is prevalent in sarcasm-related communication; thus, the authors propose to first train a sentiment model (based on a CNN) for learning sentiment-specific feature extraction. The model learns local features in lower layers which are then converted into global features in the higher layers. The authors observe that sarcastic expressions are user-specific — some users post more sarcasm than others.

In the proposed framework, personality-based features, sentiment features, and emotion-based features are incorporated into the sarcasm detection framework. Each set of features are learned by separate models, becoming pre-trained models used to extract sarcasm-related features from a dataset.

CNN Framework
CNNs are effective at modeling hierarchy of local features to learn more global features, which is essential to *learn context*. Sentences are represented using word vectors (embeddings) and provided as input. Google's word2vec vectors are employed as input. Non-static representations are used, therefore, parameters for these word vectors are learned during the training phase. Max pooling is then applied to the feature maps to generate features. A fully connected layer is applied followed by a softmax layer for outputting the final prediction. (See diagram of the CNN-based architecture below)



To obtain the other features — sentiment (S), emotion (E), and personality (P) — CNN models are pre-trained and used to extract features from the sarcasm datasets. Different training datasets were used to train each model. (Refer to paper for more details)

Two classifiers are tested — a pure CNN classifier (*CNN*) and CNN-extracted features fed to an SVM classifier (*CNN-SVM*).

A separate baseline classifier (B) — consisting of only the CNN model without the incorporation of the other models (e.g., emotion and sentiment) — is trained as well.

Experiments

Data — Balanced and imbalanced sarcastic tweets datasets were obtained from (Ptacek et al., 2014) and The Sarcasm Detector. Usernames, URLs, and hashtags are removed, and the NLTK Twitter Tokenizer was used for tokenization. (See paper for more details)

The performances of both the CNN and CNN-SVM classifier, when applied to all datasets, are shown in the table below. We can observe that when the models (specifically CNN-SVM) combine sarcasm features, emotion features, sentiment features, and personality traits features, it outperforms all the other models with the exception of the baseline model (B).

B	S	E	P	Dataset 1		Dataset 2		Dataset 3	
				CNN	CNN-SVM	CNN	CNN-SVM	CNN	CNN-SVM
+				95.04%	97.60%	89.33%	92.32%	88.00%	92.20%
	+			-	87.00%	-	86.50%	-	73.50%
		+		-	76.30%	-	84.71%	-	72.10%
			+	-	75.00%	-	77.90%	-	74.41%
	+	+	+	-	**90.70%**	-	**90.90%**	-	**84.43%**
+	+			95.21%	97.67%	89.69%	94.60%	88.58%	93.12%
+		+		95.22%	97.65%	89.72%	94.50%	88.56%	92.63%
+			+	95.21%	97.64%	89.62%	93.60%	88.26%	92.50%
+	+	+	+	95.30%	**97.71%**	89.73%	**94.80%**	88.51%	**93.30%**

Table 2: Experimental Results. Legenda: B = Baseline, S = Sentiment, E = Emotion, P = Personality, 5-fold cross-validation is carried out for all the experiments.

The table below shows comparison results of the the state-of-the-art model (method 1), other well-known sarcasm detection research (method 2), and the proposed model (method 3). The proposed model consistently outperforms all the other models.

Method	Dataset 1	Dataset 2	Dataset 3	D3 => D1
(Ptácek et al., 2014)	94.66%	92.37%	63.37%	53.02%
(Joshi et al., 2015)	65.05%	62.37%	60.80%	47.32%
Proposed Method (using all features)	97.71%	94.80%	93.30%	76.78%

Table 3: Performance comparison of the proposed method and the state-of-the-art approaches. Legenda:

Dataset 1 and tested on Dataset 2; the F1-score of the model was 33.05%, significantly dropping in accuracy.

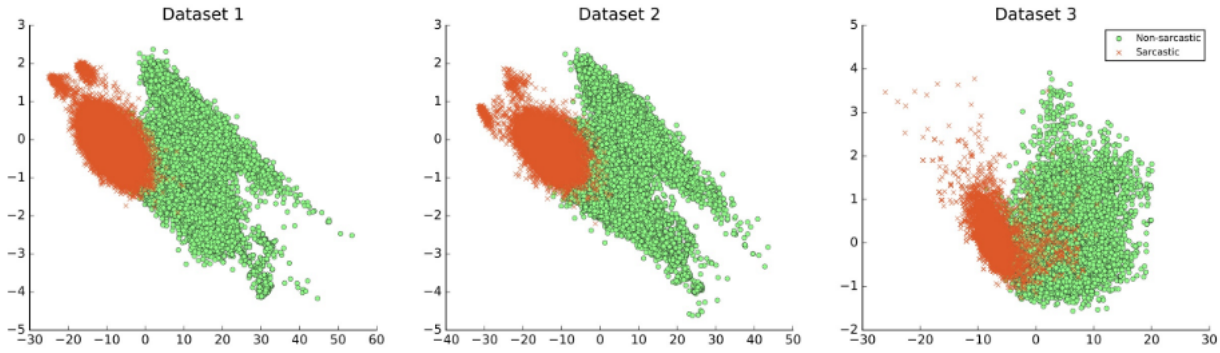


Figure 3: Visualization of the data.

Conclusion and Future Work

- Overall, the authors found that sarcasm is very *topic-dependent* and *highly contextual*, therefore, sentiment and other contextual clues help to detect sarcasm from text. Pre-trained sentiment, emotion, and personality models are used to capture contextualized information from text.
- Hand-crafted features (e.g., n-grams), though somewhat useful for sarcasm detection, will produce very sparse feature vector representations. For those reasons, word embeddings are used as input features.

References

Ref: https://arxiv.org/abs/1610.08815 — "A Deeper Look into Sarcastic Tweets Using Deep Convolutional Neural Networks"

You can find an interesting discussion of this post on Reddit here.

👋*Wait! Here is more if you want to continue reading similar research:* Detecting Emotions with CNN Fusion Models