Modelling Context with User Embeddings for Sarcasm Detection in Social Media

A Breakdown

Introduction

This paper introduces a novel deep neural network, CUE-CNN, for automated sarcasm detection. Through this system , they aim to improvise over previous methods that either required considerable feature engineering , or did not focus on contextual dependencies. They focus of this neural model is to represent and exploit 'content' and 'context' through convolutional layers and user embeddings

Main Contributions

1. Proposing a novel convolutional neural network based model that explicitly learns and exploits user embeddings in conjunction with features derived from utterances
2. Showing that this model outperforms the strong baseline recently proposed by Bamman and Smith (2015) by more than 2% in absolute accuracy, while obviating the need to manually engineer features
3. Showing that the learned user embeddings can capture relevant user attributes.

The Crux of the paper. Let's dive in!

Learning User Embedding:

The focus here is on understanding the relation between the user and the content they produce. This is done by optimizing the conditional probability of texts, given their authors as follows

$$P(S|\text{user}_j) = \sum_{w_i \in S} \log P(w_i|\mathbf{u}_j) + \sum_{w_i \in S} \sum_{w_k \in C(w_i)} \log P(w_i|\mathbf{e}_k) \quad (1)$$

This objective function now takes care of two contextual dependencies; the author and neighbouring words to a certain word 'wi'.This is done through 2 ways:

1. S represents a vector of words {w1,w2…,wn}. For each word in a sentence , the log probability of each word occurring given the specific user is provided.
2. C(wi) refers to the 'context' around the word wi, representing the neighbouring words.

Basically, for every word, they take the a certain window of neighbouring words and summarise their probability of occurring around word wi.

(NOTE: ek is the embeddings of the words k, and uj is embeddings of the user j.)

To estimate the conditional probabilities which are being worked with above, they are calculated as follows:

$$P(w_i|\mathbf{x}) = \frac{\exp(\mathbf{W}_i \cdot \mathbf{x} + \mathbf{b}_i)}{\sum_{k=1}^{Y} \exp(\mathbf{W}_k \cdot \mathbf{x} + \mathbf{b}_k)}$$

Where x represents a feature vector, W being weights and b representing bias.

An issue with this equation is that the denominator would, over a large text corpus, incur too much computational cost . This is solved by the introducing a hierarchal softmax function, which replaces the flat softmax function with a hierarchal layer represented as a binary tree, and approximates the conditional probabilities by choosing only the most efficient paths down the tree.

To approximate the P(wi | uj) probability, they attempt to minimise the following hinge loss objective function:

$$\mathcal{L}(w_i, \text{user}_j) = \sum_{w_l \in V, w_l \notin S} \max(0, 1 - \mathbf{e}_i \cdot \mathbf{u_j} + \mathbf{e}_l \cdot \mathbf{u_j}) \tag{3}$$

What's very interesting here, is that 'wl' is a word that is not present in the current sentence. This way, it acts as a 'negative example' and over the aggregation , the function learns which words the user is more likely to use than others. This is the basic concept of negative sampling , where you approximate the objective function in a binary classification task by learning to discriminate be- tween observed positive examples (sampled from the true distribution) and pseudo-negative examples (sampled from a large space of predominantly negative instances).

The way this paper proposes choosing negative samples is through picking the most commonly used words in a unigram(1 word considered at a time) model. This forces the model . This forcing the representations to capture the differences between the words a given individual employs and the words that everyone commonly uses.

Proposed Model:

If you remember from the starting of this breakdown, I had mentioned that the focus of this paper's model is on 'content' and 'context'. Let's focus back on that to understand their model.

For a given sentence, they use pre-trained word embeddings as an input to a convolutional layer that extracts high level features. This provides the columns corresponding to each word, and combined represents a sentence matrix as follows:

$$\mathbf{S} = \begin{bmatrix} — & \mathbf{e}_1 & — \\ & \vdots & \\ — & \mathbf{e}_m & — \end{bmatrix}$$

This is passed to a convolutional layer. Now, a convolutional layer is composed of a set of filters , and each of these filters have a certain height h. This filter slides across the sentence matrix, multiplying with a certain sub-matrix of the original sentence matrix, and procuring h-sized features for each word. This constitutes a feature map , which consititues of all the features of the sub-matrices of S. On adding the bias at each time, we pass the obtained multidimensional results through a non-linear activation(ReLU)(alpha in the formula).

$$\mathbf{m}_i = \alpha(\mathbf{F} \cdot \mathbf{S}_{[i:i-h+1]} + b)$$

Here:

1. F representing the filter weights
2. S[i:j] represents a sub-matrix
3. B is the bias

This is then passed through a max pooling layer to convert it to scalar. A max pooling layer picks the largest feature from the feature map for each sentence, and this is then concatenated into a single scalar vector. This is then passed to the hidden layer of the model.

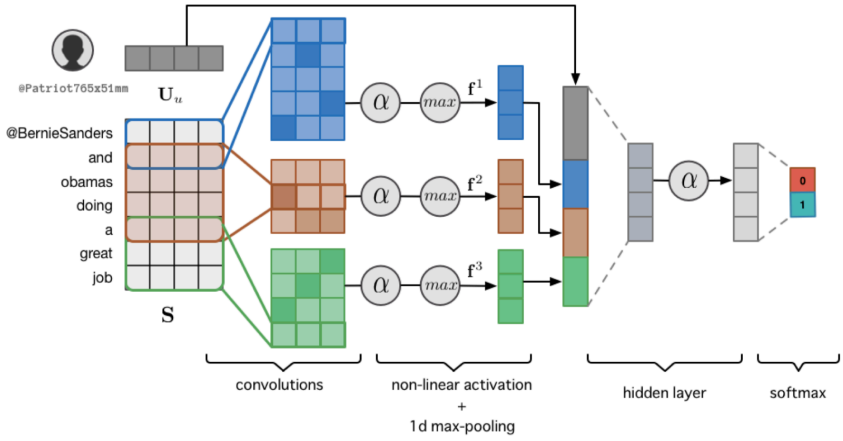
For the context, we use the User Embeddings matrix , which has 'd' dimensional rows for each of the 'N' users(N x d size). The final formulation of the model:

$$P(y = k|s, u; \theta) \propto \mathbf{Y}_k \cdot g(\mathbf{c}_S \oplus \mathbf{U}_u) + \mathbf{b}_k$$
$$g(\mathbf{x}) = \alpha(\mathbf{H} \cdot \mathbf{x} + \mathbf{h})$$

Here:

1. Yk is the weight of output layer
2. Bk is the bias of output layer
3. g(.) is the activation function
4. H and h are the weights and biases of the hidden layer

Following is a representation of the model.



Results!

This model outperforms the state of the art models by an absolute accuracy of 2%. Moreover, it manages to accomplish the task of sarcastic classification with minimal feature engineering, allowing their novel model to decide the best features using convolutional networks and user embedding mathematical modelling.