Modelling Context with User Embeddings for Sarcasm Detection in Social Media

Silvio Amir **Byron C. Wallace**[†] **Hao Lyu**[†] **Paula Carvalho** **Mário J. Silva**
INESC-ID Lisboa, Instituto Superior Técnico, Universidade de Lisboa
[†]iSchool, University of Texas at Austin
samir@inesc-id.pt byron.wallace@utexas.edu xalh8083@gmail.com
pcc@inesc-id.pt mjs@inesc-id.pt

Abstract

We introduce a deep neural network for automated sarcasm detection. Recent work has emphasized the need for models to capitalize on *contextual* features, beyond lexical and syntactic cues present in utterances. For example, different speakers will tend to employ sarcasm regarding different subjects and, thus, sarcasm detection models ought to encode such speaker information. Current methods have achieved this by way of laborious feature engineering. By contrast, we propose to automatically learn and then exploit *user embeddings*, to be used in concert with lexical signals to recognize sarcasm. Our approach does not require elaborate feature engineering (and concomitant data scraping); fitting user embeddings requires only the text from their previous posts. The experimental results show that the our model outperforms a state-of-the-art approach leveraging an extensive set of carefully crafted features.

1 Introduction

Existing social media analysis systems are hampered by their inability to accurately detect and interpret figurative language. This is particularly relevant in domains like the social sciences and politics, in which the use of figurative communication devices such as verbal irony (roughly, sarcasm) is common. Sarcasm is often used by individuals to express opinions on complex matters and regarding specific targets (Carvalho et al., 2009).

Early computational models for verbal irony and sarcasm detection tended to rely on shallow methods exploiting conditional token count regularities. But lexical clues alone are insufficient

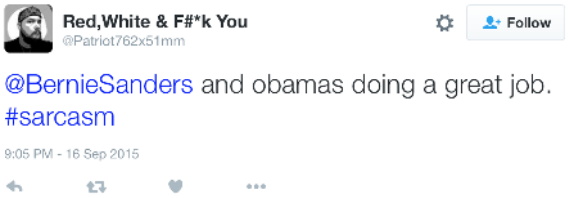


Figure 1: An illustrative tweet.

to discern ironic intent. Appreciating the *context* of utterances is critical for this; even for humans (Wallace et al., 2014). Indeed, the exact same sentence can be interpreted as literal or sarcastic, *depending on the speaker*. Consider the sarcastic tweet in Figure 1 (ignoring for the moment the attached `#sarcasm` hashtag). Without knowing the author's political leanings, it would be difficult to conclude with certainty whether the remark was intended sarcastically or in earnest.

Recent work in sarcasm detection on social media has tried to incorporate contextual information by exploiting the preceding messages of a user, to e.g., detect contrasts in sentiments expressed towards named entities (Khattri et al., 2015), infer behavioural traits (Rajadesingan et al., 2015) and capture the relationship between authors and the audience (Bamman and Smith, 2015). However, all of these approaches require the design and implementation of complex features that explicitly encode the content and (relevant) context of messages to be classified. This feature engineering is labor intensive, and depends on external tools and resources. Therefore, deploying such systems in practice is expensive, time-consuming and unwieldy.

We propose a novel approach to sarcasm detection on social media that does not require extensive manual feature engineering. Instead, we develop a neural model that learns to represent and exploit embeddings of both *content* and *context*. For the former, we induce vector lexical repre-

Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL), pages 167–177,
Berlin, Germany, August 7-12, 2016.

sentations via a convolutional layer; for the latter, our model learns *user embeddings*. Inference concerning whether an utterance (tweet) was intended ironically (or not) is then modelled as a joint function of lexical representations and corresponding author embeddings.

The main contributions of this paper are as follows. (i) We propose a novel convolutional neural network based model that explicitly learns and exploits user embeddings in conjunction with features derived from utterances. (ii) We show 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. (iii) We show that the learned user embeddings can capture relevant user attributes.

2 Related Work

Verbal irony is a rhetorical device in which speakers say something other than, and often opposite to, what they actually mean.[1] Sarcasm may be viewed as a special case of irony, where the positive literal meaning is perceived as an indirect insult (Dews et al., 1995).

Most of the previously proposed computational models to detect irony and sarcasm have used features similar to those used in sentiment analysis. Carvalho et al. (2009) analyzed comments posted by users on a Portuguese online newspaper and found that oral and gestural cues indicate irony. These included: emoticons, onomatopoeic expressions for laughter, heavy punctuation, quotation marks and positive interjections. Others have used text classifiers with features based on word and character n-grams, sentiment lexicons, surface patterns and textual markers (Davidov et al., 2010; González-Ibánez et al., 2011; Reyes et al., 2013; Lukin and Walker, 2013). Elsewhere, Barbieri and Saggion (2014) derived new word-frequency based features to detect irony, e.g., combinations of frequent and rare words, ambiguous words, 'spoken style' words combined with 'written style' words and intensity of adjectives. Riloff et al. (2013) demonstrated that one may exploit the apparent expression of contrasting sentiment in the same utterance as a marker of verbal irony.

The aforementioned approaches rely predominantly on features *intrinsic* to texts, but these will often be insufficient to infer figurative meaning: context is needed. There have been some recent attempts to exploit contextual information, e.g. Khattri et al. (2015) extended the notion of *contrasting sentiments* beyond the textual content at hand. In particular, they analyzed previous posts to estimate the author's prior sentiment towards specific *targets* (i.e., entities). A tweet is then predicted to be sarcastic if it expresses a sentiment about an entity that contradicts the author's (estimated) prior sentiment regarding the same.

Rajadesingan et al. (2015) built a system based on theories of sarcasm expression from psychology and behavioral sciences. To operationalize such theories, they used several linguistic tools and resources (e.g. lexicons, sentiment classifiers and a PoS tagger), in addition to user profile information and previous posts, to model a range of behavioural aspects (e.g., mood, writing style). Wallace et al. (2015) developed an approach for classifying posts on *reddit*[2] as sarcastic or literal, based in part on the interaction between the specific sub-reddit to which a post was made, the entities mentioned, and the (apparent) sentiment expressed. For example, if a post in the (politically) conservative sub-reddit mentions Obama, it is more likely to have been intended ironically than posts mentioning Obama in the progressive sub-reddit. But this approach is limited because it relies on the unique sub-reddit structure. Bamman and Smith (2015) proposed an approach that relied on an extensive, rich set of features capturing various contextual information about authors of tweets and the audience (in addition to lexical cues). We review these at length in Section 5.1.

A major downside of these and related approaches, however, is the amount of manual effort required to derive these feature sets. A primary goal of this work is to explore whether neural models can effectively learn these rich contextualizing features, thus obviating the need to manually craft them. In particular, the model we propose similarly aims to combine lexical clues with extra-linguistic information. Unlike prior work, however, our model attempts to automatically induce representations for the content and the author of a message that are predictive of sarcasm.

[1] Like other forms of subjective expression, irony and sarcasm are difficult to define precisely.

[2] `http://reddit.com` is a social news aggregation site comprising specific topical *sub-reddits*.

3 Learning User Embeddings

Our goal is to learn representations (vectors) that encode latent aspects of users and capture homophily, by projecting similar users into nearby regions of the embedding space. We hypothesize that such representations will naturally capture some of the signals that have been described in the literature as important indicators of sarcasm, such as contrasts between what someone believes and what they have ostensibly expressed (Campbell and Katz, 2012) or Kreuz (1996) principle of *inferability*, stating that sarcasm requires a common ground between parties to be understood.

To induce the user embeddings, we adopt an approach similar to that described in the preliminary work of Li et al. (2015). In particular, we capture relations between users and the content they produce by optimizing the conditional probability of texts, given their authors (or, more precisely, given the vector representations of their authors). This method is akin to Le and Mikolov (2014)'s *Paragraph Vector* model, which jointly estimates embeddings for words and paragraphs by learning to predict the occurrence of a word w within a paragraph p conditioned on the (learned) representation for p.

Given a sentence $S = \{w_1, \ldots, w_N\}$ where w_i denotes a word drawn from a vocabulary $\mathcal{V}$, we aim to maximize the following probability:

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

Where $C(w_i)$ denotes the set of words in a prespecified window around word w_i, $\mathbf{e}_k \in \mathbb{R}^d$ and $\mathbf{u}_j \in \mathbb{R}^d$ denote the embeddings of word k and user j, respectively. This objective function encodes the notion that the occurrence of a word w, depends both on the author of S and it's neighbouring words.

The conditional probabilities in Equation 1 can be estimated with log-linear models of the form:

$$P(w_i|\mathbf{x}) = \frac{\exp(\mathbf{W}_i \cdot \mathbf{x} + \mathbf{b}_i)}{\sum_{k=1}^{Y} \exp(\mathbf{W}_k \cdot \mathbf{x} + \mathbf{b}_k)} \quad (2)$$

Where $\mathbf{x}$ denotes a feature vector, $\mathbf{W}_k$ and $\mathbf{b}_k$ are the weight vectors and bias for class k. In our case, we treat words as classes to be predicted. Calculating the denominator thus requires summing over all of the words in the (large) vocabulary, an expensive operation. To avoid this computational bottleneck, we approximate the term $P(w_i|\mathbf{e}_k)$ with Morin and Bengio (2005) Hierarchical Softmax.[3]

To learn meaningful user embeddings, we seek representations that are predictive of individual word-usage patterns. In light of this motivation, we approximate $P(w_i|\mathbf{u}_j)$ via the following hinge-loss objective which we aim to minimize:

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

Here, each w_l (and corresponding embedding, $\mathbf{e}_l$) is a *negative example*, i.e., a word not in the sentence under consideration, which was authored by user j. The intuition is that in the aggregate, such words are less likely to be employed by user j than words observed in sentences she has authored. Thus minimizing this objective attempts to induce a representation that is discriminative with respect to word usage.

In practice, V will be very large and hence we approximate the objective via *negative sampling*, a variant of Noise Contrastive Estimation. The idea is to approximate the objective function in a binary classification task by learning to discriminate between observed positive examples (sampled from the true distribution) and *pseudo*-negative examples (sampled from a large space of predominantly negative instances). Intuitively, this shifts probability mass to plausible observations. See Dyer (2014) for notes on Negative Sampling and Noise Contrastive Estimation.

Previous work by Collobert et al. (2011) showed that this approach works well in representation learning tasks when a sufficient amount of training data is available . However, we have access to only a limited amount of text for each user (see Section 5). We hypothesize that this problem can be alleviated by carefully selecting the negative samples that mostly contribute to 'push' the vectors into the appropriate region of the embedding space (i.e., closer to the words commonly employed by a given user and far from other words).

[3] As implemented in `Gensim` (Řehůřek and Sojka, 2010).

This requires designing a strategy for selectively sampling negative examples. One straightforward approach would be to sample from a user-specific unigram model, informing which words are less likely to be utilized by that user. But estimating the parameters of such model with scarce data would be prone to overfitting. Instead, we sample from a unigram distribution estimated from the posts authored by all the users. The goal is to select the most commonly used words as the negative samples, thereby forcing the representations to capture the differences between the words a given individual employs and the words that everyone commonly uses.

4 Proposed Model

We now present the details of our proposed sarcasm detection model. Given a message S authored by user u, we wish to capture both the relevant aspects of the *content* and the relevant *contextual* information about the author. To represent the content, we use pre-trained word embeddings as the input to a convolutional layer that extracts high-level features. More formally, let $\mathbf{E} \in \mathbb{R}^{d \times |\mathcal{V}|}$ be a pre-trained word embedding matrix, where each column represents a word from the vocabulary $\mathcal{V}$ as a d dimensional vector. By selecting the columns of $\mathbf{E}$ corresponding to the words in S, we form the sentence matrix:

$$\mathbf{S} = \begin{bmatrix} — & \mathbf{e}_1 & — \\ & \vdots & \\ — & \mathbf{e}_m & — \end{bmatrix} \tag{4}$$

A convolutional layer is composed of a set of filters $\mathbf{F} \in \mathbb{R}^{d \times h}$ where h is the *height* of the filter. Filters *slide* across the input, extracting h-gram features that constitute a feature map $\mathbf{m} \in \mathbb{R}^{|S|-h+1}$, where each entry is obtained as

$$\mathbf{m}_i = \alpha(\mathbf{F} \cdot \mathbf{S}_{[i:i-h+1]} + b) \tag{5}$$

with $i = 1, \ldots i - h + 1$. Here, $\mathbf{S}_{[i:j]}$ denotes a sub-matrix of $\mathbf{S}$ (from row i to row j), $b \in \mathbb{R}$ is an additive bias and $\alpha(\cdot)$ denotes a non-linear activation function, applied element-wise. We transform the resultant feature map into a scalar using *max-pooling*, i.e., we extract the largest value in the map. We use 3 filters (with varying heights) each of which generates M feature maps that are reduced to a vector $\mathbf{f}^k = [max(\mathbf{m}^1) \oplus max(\mathbf{m}^2) \ldots \oplus max(\mathbf{m}^M)]$, where $\oplus$ denotes concatenation. We set $\alpha(\cdot)$ to be the *Rectified Linear Unit* activation function (Nair and Hinton, 2010). The output of all the filters is then combined to form the final representation $\mathbf{c} = [\mathbf{f}^1 \oplus \mathbf{f}^2 \oplus \mathbf{f}^3]$. We will denote this feature vector of a specific sentence S by $\mathbf{c}_S$.

To represent the context, we assume there is a user embedding matrix $\mathbf{U} \in \mathbb{R}^{d \times N}$, where each column represents one of N users with a d dimensional vector. The parameters of this embedding matrix can be initialized randomly or using the approach described in Section 3. Then, we simply map the author of the message into the user embedding space by selecting the corresponding column of $\mathbf{U}$. Letting $\mathbf{U}_u$ denote the user embedding of author u, we formulate our sarcasm detection model as follows:

$$\begin{gathered} P(y = k|s, u; \theta) \propto \mathbf{Y}_k \cdot g(\mathbf{c}_S \oplus \mathbf{U}_u) + \mathbf{b}_k \\ g(\mathbf{x}) = \alpha(\mathbf{H} \cdot \mathbf{x} + \mathbf{h}) \end{gathered} \tag{6}$$

where $g(\cdot)$ denote the activations of a hidden layer, capturing the relations between the content and context representations, and $\theta = \{\mathbf{Y}, \mathbf{b}, \mathbf{H}, \mathbf{h}, \mathbf{F}^1, \mathbf{F}^2, \mathbf{F}^3, \mathbf{E}, \mathbf{U}\}$ are parameters to be estimated during training. Here, $\mathbf{Y} \in \mathbb{R}^{2 \times z}$ and $\mathbf{b} \in \mathbb{R}^2$ are the weights and bias of the output layer; $\mathbf{H} \in \mathbb{R}^{z \times 3M+d}$ and $\mathbf{h} \in \mathbb{R}^z$ are the weights and bias of the hidden layer; and $\mathbf{F}^i$ are the weights of the convolutional filters. Henceforth, we will refer to this approach as *Content and User Embedding Convolutional Neural Network* (CUE-CNN). Figure 2 provides an illustrative schematic depicting this model.

5 Experimental Setup

We replicated Bamman and Smith (2015) experimental setup using (a subset of) the same Twitter corpus. The labels were inferred from self-declarations of sarcasm, i.e., a tweet is considered sarcastic if it contains the hashtag `#sarcasm` or `#sarcastic` and deemed non-sarcastic otherwise.[4] To comply with Twitter terms of service, we were given only the tweet ids along with the corresponding labels and had to retrieve the messages ourselves. By the time we tried to retrieve the messages, some of them were not available. We also did not have access to the historical user tweets used by Bamman and Smith, hence, for

[4]Note that this is a form of noisy supervision, as of course all sarcastic tweets will not be explicitly flagged as such.

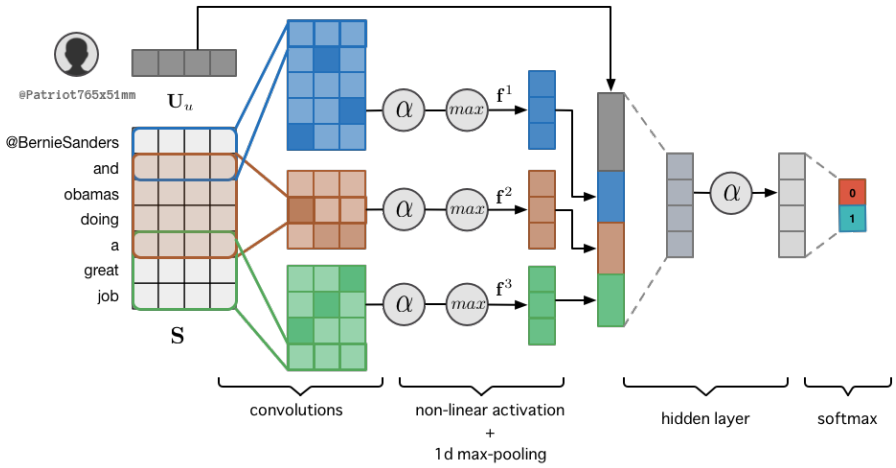


Figure 2: Illustration of the CUE-CNN model for sarcasm detection. The model learns to represent and exploit embeddings of both *content* and *users* in social media.

each author and mentioned user, we scraped additional tweets from their Twitter feed. Due to restrictions in the Twitter API, we were only able to crawl at most 1000 *historical tweets* per user.[5] Furthermore, we were unable to collect historical tweets for a significant proportion of the users, thus, we discarded messages for which no contextual information was available, resulting in a corpus of 11,541 tweets involving 12,500 unique users (authors and mentioned users). It should also be noted that our historical tweets were posted *after* the ones in the corpus used for the experiments.

5.1 Baselines

We reimplemented Bamman and Smith (2015)'s sarcasm detection model. This a simple, logistic-regression based classifier that exploits rich feature sets to achieve strong performance. These are detailed at length in the original paper, but we briefly summarize them here:

- **tweet-features**, encoding attributes of the target tweet text, including: uni- and bi-gram bag of words (BoW) features; Brown et al. (1992) word clusters indicators; unlabeled dependency bigrams (both BoW and with Brown cluster representations); part-of-speech, spelling and abbreviation features; inferred sentiment, at both the tweet and word level; and 'intensifier' indicators.
- **author-features**, aimed at encoding attributes of the author, including: historically 'salient' terms used by the author; the inferred distribution over topics[6] historically tweeted about by the user; inferred sentiment historically expressed by the user; and author profile information (e.g., profile BoW features).
- **audience-features**, capturing properties of the *addressee* of tweets, in those cases that a tweet is directed at someone (via the @ symbol). A subset of these, duplicate the aforementioned author features for the addressee. Additionally, author/audience interaction features are introduced, which capture similarity between the author and addressee, w.r.t. inferred topic distributions. Finally, this set includes a feature capturing the frequency of past communication between the author and addressee.
- **response-features**, for tweets written in response to another tweet. This set of features captures information relating the two, with BoW features of the original tweet and pairwise cluster indicator features, which the encode Brown clusters observed in both the original and response tweet.

We emphasize that implementing this rich set of features took considerable time and effort. This motivates our approach, which aims to effectively induce and exploit contextually-aware representations without manual feature engineering.

[5]The original study (Bamman and Smith, 2015) was done with at most 3,200 historical tweets.

[6]The topics were extracted from Latent Dirichlet Allocation (Blei et al., 2003).

To assess the importance of modelling contextual information in sarcasm detection, we considered two groups of models as baselines: the first only takes into account the content of a target tweet. The second, combines lexical cues with contextual information. The first group includes the following models:

- UNIGRAMS: ℓ_2-regularized logistic regression classifier with binary unigrams as features.
- TWEET ONLY: ℓ_2-regularized logistic regression classifier with binary unigrams and **tweet-features**.
- NBOW: Logistic regression with neural word embeddings as features. Given a sentence matrix $\mathbf{S}$ (Eq. 4) as input, a d-dimensional feature vector is computed by summing the individual word embeddings.
- NLSE: The Non-linear subspace embedding model due to Astudillo et al. (2015). The NLSE model adapts pre-trained word embeddings for specific tasks by learning a projection into a small subspace. The idea is that this subspace captures the most discriminative latent aspects encoded in the word embeddings. Given a sentence matrix $\mathbf{S}$, each word vector is first projected into the subspace and then transformed through an element-wise sigmoid function. The final sentence representation is obtained by summing the (adapted) word embeddings and passed into a softmax layer that outputs the predictions.
- CNN: The CNN model for text classification proposed by Kim (2014), using only features extracted from the convolutional layer acting on the lexical content. The input layer was initialized with pre-trained word embeddings.

The second group of baselines consists of the following models:

- TWEET+*: ℓ_2-regularized logistic regression classifier with a combination of **tweet-features** and each of the aforementioned Bamman and Smith (2015) feature sets.
- SHALLOW CUE-CNN: A simplified version of our neural model for sarcasm detection, without the hidden layer. We evaluated two variants: initializing the user embeddings at random, and initializing the user embeddings with the approach described in Section 3 (SHALLOW CUE-CNN+USER2VEC). In both cases, the (word and user) embeddings weights were updated during training.
- CUE-CNN+*: Our proposed neural network for sarcasm detection. We also evaluated the two aforementioned variants: randomly initialized user embeddings and pre-trained user embeddings. But here we compared two different approaches for the negative sampling procedure, namely, sampling from a unigram distribution (CUE-CNN+USER2VEC) and sampling uniformly at random from the vocabulary (CUE-CNN+USER2VEC-UNIFRAND).

5.2 Pre-Training Word and User Embeddings

We first trained Mikolov et al. (2013)'s *skip-gram* model variant to induce word embeddings using the union of: Owoputi et al. (2013)'s dataset of 52 Million unlabeled tweets, Bamman and Smith (2015) sarcasm dataset and 5 Million historical tweets collected from users.

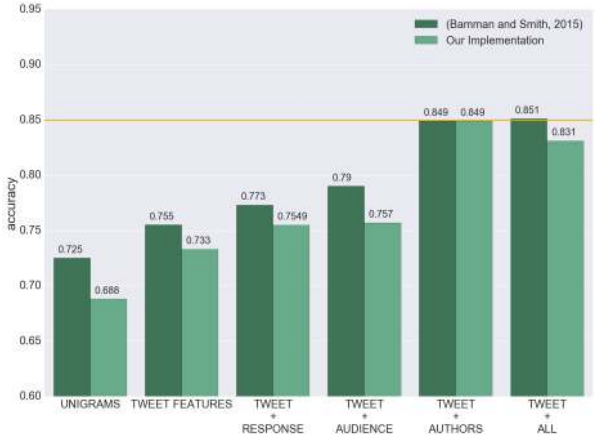
To induce user embeddings, we estimated a unigram distribution with maximum likelihood estimation. Then, for each word in a tweet, we extracted 15 negative samples (for the first term in Eq.1) and used the skip-gram model to precompute the conditional probabilities of words occurring in a window of size 5 (for the second term in Eq.1). Finally, Equation 1 was minimized via Stochastic Gradient Descent on 90% of the historical data, holding out the remainder for validation and using the $P(\text{tweet text}|\text{user})$ as early stopping criteria.

Note that the parameters for each user only depend on their own tweets; this allowed us to perform these computations in parallel to speed-up the training.

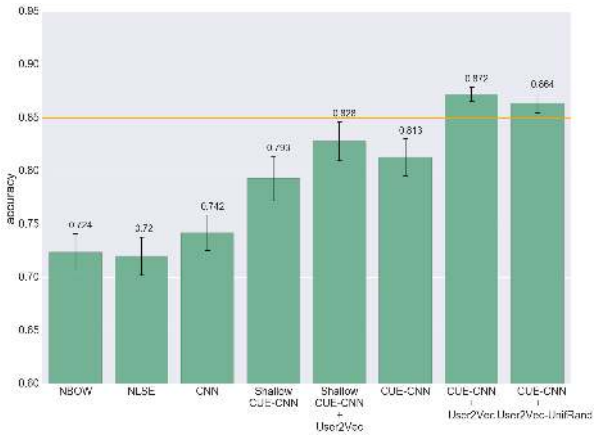
m

5.3 Model Training and Evaluation

Evaluation was performed via 10-fold cross-validation. For each split, we fit models to 80% of the data, tuned them on 10% and tested on the remaining, held-out 10%. These data splits were kept fixed in all the experiments. For the linear classifiers, in each split, the regularization constant was selected with a linear search over the



(a) Performance of the linear classifier baselines. We include the results reported by Bamman and Smith (2015) as a reference. Discrepancies between their reported results and those we achieved with our re-implementation reflect the fact that their experiments were performed using a significantly larger training set and more historical tweets than we had access to.



(b) Performance of the proposed neural models. We compare simple neural models that only consider the lexical content of a message (first 3 bars) with architectures that explicitly model the context. Bars 4 and 5 show the gains obtained by pre-training the user embeddings. The last 2 bars compare different negative sampling procedures for the user embedding pre-training.

Figure 3: Comparison of different models. The left sub-plot shows linear model performance; the right shows the performance of neural variants. The horizontal line corresponds to the best performance achieved via linear models with rich feature sets. Performance was measured in terms of average accuracy over 10-fold cross-validation; error bars depict the variance.

range $C = [1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}, 1, 10]$ using the training set to fit the model and evaluating on the tuning set. After selecting the best regularization constant, the model was re-trained on the union of the train and tune sets, and evaluated on the test set.

To train our neural model, we first had to choose a suitable architecture and hyperparameter set. However, selecting the optimal network parametrization would require an extensive search over a large configuration space. Therefore, in these experiments, we followed the recommendations in Zhang and Wallace (2015), focusing our search over combinations of dropout rates $D = [0.0, 0.1, 0.3, 0.5]$, filter heights $H = [(1, 3, 5), (2, 4, 6), (3, 5, 7), (4, 6, 8), (5, 7, 9)]$, number of feature maps $M = [100, 200, 400, 600]$ and size of the hidden layer $Z = [25, 50, 75, 100]$.

We performed random search by sampling without replacement over half of the possible configurations. For each data split, 20% of the training set was reserved for early stopping. We compared the sampled configurations by fitting the model on the remaining training data and testing on the tune set. After choosing the best configuration, we retrained the model on the union of the train and tune set (again reserving 20% of the data for early stopping) and evaluated on the test set.

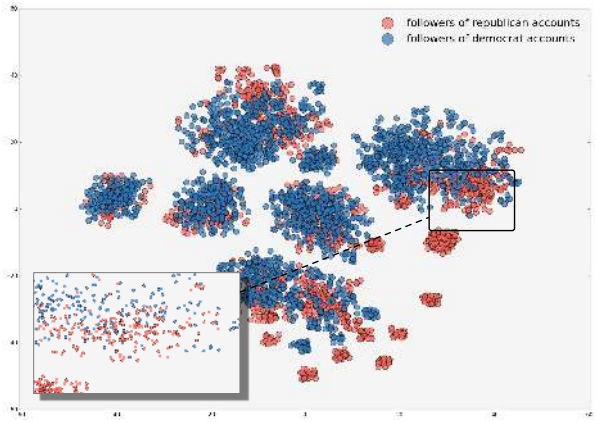
The model was trained by minimizing the cross-entropy error between the predictions and true labels, the gradients w.r.t to the network parameters were computed with backpropagation (Rumelhart et al., 1988) and the model weights were updated with the AdaDelta rule (Zeiler, 2012).

6 Results

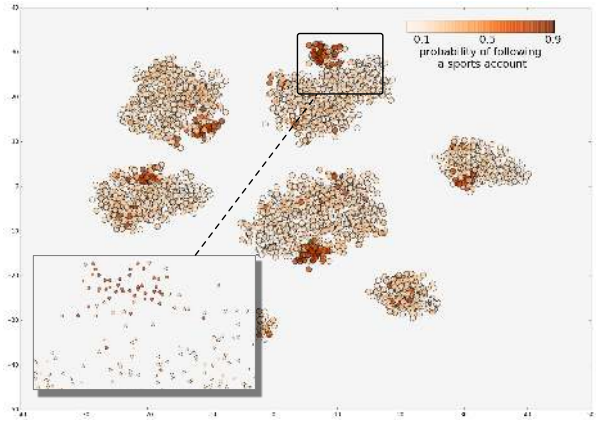
6.1 Classification Results

Figure 3 presents the main experimental results. In Figure 3a, we show the performance of linear classifiers with the manually engineered feature sets proposed by Bamman and Smith (2015). Our results differ slightly from those originally reported. Nonetheless, we observe the same general trends: namely, that including contextual features significantly improves the performance, and that the biggest gains are attributable to features encoding information about the authors of tweets.

The results of neural model variants are shown in Figure 3b. Once again, we find that modelling the context (i.e., the author) of a tweet yields significant gains in accuracy. The difference is that here the network jointly *learns* appropriate user representations, lexical feature extractors and, finally, the classification model. Further improvements are realized by pre-training the user embed-



(a) Users colored according to the politicians they follow on Twitter: the blue circles represent users that follow at least one of the (democrats) accounts: *@BarackObama*, *@HillaryClinton* and *@BernieSanders*; the red circles represent users that follow at least one of the (republicans) accounts: *@marcorubio*, *@tedcruz* and *@realDonaldTrump*. Users that follow accounts from both groups were excluded. We can see that users with a similar political leaning tend to have similar vectors.



(b) Users colored with respect to the likelihood of following a sports account. The 500 most popular accounts (according to the authors in our training data) were manually inspected and 100 sports related accounts were selected, e.g., *@SkySports*, *@NBA* and *@cristiano*. We should note that users for which the probabilities lied in the range between $0.3 - 0.7$ were discarded to emphasize the extremes.

Figure 4: T-SNE projection of the user embeddings into 2-dimensions. The users are color coded according to their political preferences and interest in sports. The visualization suggests that the learned embeddings capture some notion of homophily.

dings (we elaborate on this in the following section). We see additional gains when we introduce a hidden layer capturing the *interactions* between the context (i.e., user vectors) and the content (lexical vectors). This is intuitively agreeable: the recognition of sarcasm is possible when we jointly consider the speaker and the utterance at hand. Interestingly, we observed that our proposed model not only outperforms all the other baselines, but also shows less variance over the cross-validation experiments.

Finally, we compared the effect of obtaining negative samples uniformly at random with sampling from a unigram distribution. The experimental results show that the latter improves the accuracy of the model by 0.8%. We believe the reason is that considering the most likely words (under the model) as negative samples, helps by pushing the user vectors away from non-informative words and simultaneously closer to the most discriminative words for that user.

6.2 User Embedding Analysis

We now investigate the user embeddings in more detail. In particular, we are interested in two questions: first, what aspects are being captured in these representations; and second, how they contribute to the improved performance of our model.

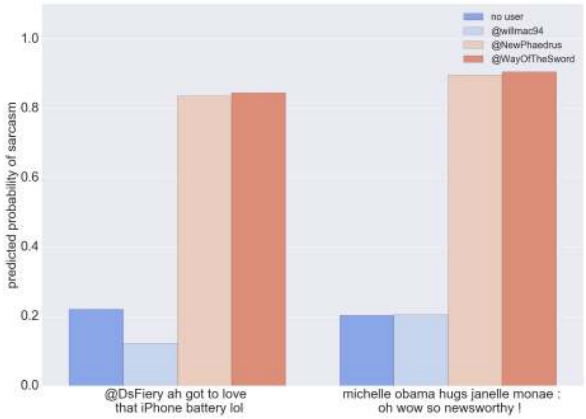


Figure 5: Two sarcastic examples that were misclassified by a simple CNN (`no user`). Using the CUE-CNN with contextual information drastically changes the model's predictions on the same examples.

To investigate the first question, we plotted a T-SNE projection (Maaten and Hinton, 2008) of the high-dimensional vector space where the users are represented into two-dimensions. We then colored each point (representing a user) according to their apparent political leaning (Figure 4a), and according to their interest in sports (Figure 4b). These attributes were inferred using the Twitter accounts that a user follows, as a proxy. The plots suggest that the user vectors are indeed able to cap-

ture latent aspects, such as political preferences and personal interests. Moreover, the embeddings seem to uncover a notion of homophily, i.e. similar users tend to occupy neighbouring regions of the embedding space. Regarding the second question, we examined the influence of the contextual information on the model's predictions. To this end, we measured the response of our model to the same textual content with different hypothetical contexts (authors). We selected two examples that were misclassified by a simple CNN and ran them trough the CUE-CNN model with three different user embeddings. In Figure 5, we show these examples along with the predicted probabilities of being a sarcastic post, when no user information is considered and when the author is taken into account. We can see that the predictions drastically change when contextual information is available and that two of the authors trigger similar responses on both examples. This example provides evidence that our model captures the intuition that the same utterance can be interpreted as sarcastic or not, depending on the speaker.

7 Conclusions

We have introduced CUE-CNN, a novel, deep neural network for automatically recognizing sarcastic utterances on social media. Our model jointly learns and exploits embeddings for the content and users, thus integrating information about the speaker and what he or she has said. This is accomplished without manual feature engineering. Nonetheless, our model *outperforms* (by over 2% in absolute accuracy) a recently proposed state-of-the-art model that exploits an extensive, handcrafted set of features encoding user attributes and other contextual information. Unlike other approaches that explicitly exploit the structure of particular social media services, such as the forum where a message was posted or metadata about the users, learning user embeddings only requires their preceding messages. Yet, the obtained vectors are able to capture relevant user attributes and a soft notion of homophily. This, we believe, makes our model easier to deploy over different social media environments.

Our implementation of the proposed method and the datasets used in this paper have been made publicly available[7]. As future work, we intended to further explore the user embeddings for context representation, namely by also incorporating the interaction between the author and the audience into the model.

[7] https://github.com/samiroid/CUE-CNN

Acknowledgments

This work was supported in part by the Army Research Office (grant W911NF-14-1-0442) and by The Foundation for Science and Technology, Portugal (FCT), through contracts UID/CEC/50021/2013, EXCL/EEI-ESS/0257/2012 (DataStorm), grant UTAP-EXPL/EEIESS/0031/2014 and Ph.D. scholarship SFRH/BD/89020/2012. This work was also made possible by the support of the Texas Advanced Computer Center (TACC) at UT Austin.

References

Ramón Astudillo, Silvio Amir, Wang Ling, Mario Silva, and Isabel Trancoso. 2015. Learning word representations from scarce and noisy data with embedding subspaces. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1074–1084, Beijing, China, July.

David Bamman and Noah A Smith. 2015. Contextualized sarcasm detection on twitter. In *Proceedings of the 9th International Conference on Web and Social Media*, pages 574–77. AAAI Menlo Park, CA.

Francesco Barbieri and Horacio Saggion. 2014. Modelling irony in twitter. In *EACL*, pages 56–64.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

John D Campbell and Albert N Katz. 2012. Are there necessary conditions for inducing a sense of sarcastic irony? *Discourse Processes*, 49(6):459–480.

Paula Carvalho, Luís Sarmento, Mário J Silva, and Eugénio De Oliveira. 2009. Clues for detecting irony in user-generated contents: oh...!! it's so easy;-). In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 53–56. ACM.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 107–116. Association for Computational Linguistics.

Shelly Dews, Joan Kaplan, and Ellen Winner. 1995. Why not say it directly? the social functions of irony. *Discourse processes*, 19(3):347–367.

Chris Dyer. 2014. Notes on noise contrastive estimation and negative sampling. *arXiv preprint arXiv:1410.8251*.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 581–586. Association for Computational Linguistics.

Anupam Khattri, Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman. 2015. Your sentiment precedes you: Using an author's historical tweets to predict sarcasm. In *6th Workshop on Computational Approaches To Subjectivity, Sentiment And Social Media Analysis WASSA 2015*, page 25.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

Roger J Kreuz. 1996. The use of verbal irony: Cues and constraints. *Metaphor: Implications and applications*, pages 23–38.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Jiwei Li, Alan Ritter, and Dan Jurafsky. 2015. Learning multi-faceted representations of individuals from heterogeneous evidence using neural networks. *arXiv preprint arXiv:1510.05198*.

Stephanie Lukin and Marilyn Walker. 2013. Really? well. apparently bootstrapping improves the performance of sarcasm and nastiness classifiers for online dialogue. In *Proceedings of the Workshop on Language Analysis in Social Media*, pages 30–40.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics.

Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 97–106. ACM.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

Antonio Reyes, Paolo Rosso, and Tony Veale. 2013. A multidimensional approach for detecting irony in twitter. *Language resources and evaluation*, 47(1):239–268.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *EMNLP*, pages 704–714.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Byron C. Wallace, Do Kook Choe, Laura Kertz, and Eugene Charniak. 2014. Humans require context to infer ironic intent (so computers probably do, too). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 512–516, Baltimore, Maryland, June. Association for Computational Linguistics.

Byron C. Wallace, Do Kook Choe, and Eugene Charniak. 2015. Sparse, contextually informed models for irony detection: Exploiting user communities, entities and sentiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1035–1044, Beijing, China, July. Association for Computational Linguistics.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.