Literature Survey

Student Name: Molly Hayward
Supervisor Name: Dr Noura Al-Moubayed
23/10/2019

Project title: Applying machine learning and deep learning techniques to sarcasm detection

I INTRODUCTION

A Problem Background

A form of irony, sarcasm is defined as the use of remarks that clearly mean the opposite of what they say, made in order to hurt someone's feelings or to criticize something in a humorous way [8]. This presents a fatal problem for classic sentiment analyzers, as text that appears to be positive at surface-level can instead convey an alternate negative meaning. Furthermore, failure to recognize sarcasm can be a human issue too, owing to the fact that its presence (or lack thereof) is considered rather subjective. Abercrombie et al. (2016) [1] showed that different humans do not classify statements as either sarcastic or non-sarcastic in the same way - truly, the existence of sarcasm can only be conclusively affirmed by the author. Additionally, developmental differences such as autism, as well as cultural and societal nuances cause variations in the way different people define and perceive sarcasm.

There exists an extensive body of literature covering machine learning and deep learning approaches to natural language processing problems, however in the application domain of sarcasm detection, they mostly display low accuracy. This could be due, in part, to the absence of a concrete, all-encompassing definition of sarcasm. N.B. this may also contribute to poor human accuracy in detecting a fundamentally human construct.

Despite the challenges in this domain, the ultimate aim of this project is to produce a tool that can detect sarcasm with a *high degree* of accuracy. In my endeavour to realise this aim, I will implement state-of-the-art word embedding and classification techniques, using machine learning and deep learning.

B Terms

NLP LSTM

II THEMES

A Vectorization of textual data

Extracting meaningful data from large corpora is undoubtedly a complex task, however in order to do this successfully, we must first construct word vectors that capture semantic and syntactic relationships in a process known as vectorization. In this section, we examine this process with the aim of answering the question: how can we *best* vectorize text so as to encode the important features of language?

Traditional Approaches – Perhaps the simplest vectorization approach is *Bag of Words (BOW)*, which encodes text as a single vector containing a count of the number of occurrences of each word in the snippet. *Term Frequency-Inverse document frequency (TF-IDF)* [9] extends the BOW model by providing each word with a score that reflects its level of importance based upon its frequency within the document. Predictably, vectors produced in this way do not preserve the context within which words are found, complicating the task of discerning their meaning. Hence, these approaches are unlikely to be suitable in the application domain of detecting sarcasm - a highly contextual phenomenon. The *Bag of N-grams* approach is a generalisation of Bag of Words, whereby instead of counting the number of occurrences of each unigram, we count the number of occurrences of each N-gram. This allows us to capture a small window of context around each word, however this results in a much larger and sparser feature set.

Word Embedding – First introduced in Mikolov et al. [5], *Word2Vec* describes a group of related models that can be used to produce high-quality vector representations of words - two such models are *Continuous Bag-of-Words* and *Continuous Skip-Gram*. It consists of a shallow (two-layer) neural network that takes a large corpus and produces a high-dimensional vector space. Unlike previous techniques, words that share a similar context tend to have collinear vectors, that is to say they are clustered together in the feature space. Consequently, Word2Vec is able to preserve the semantic relationships between words, even constructing analogies by composing vectors e.g. king - man + woman $\approx$ queen. Likewise, it captures syntactic regularities such as the singular to plural relationship e.g. cars - car $\approx$ apples - apple. In Word2Vec, intrinsic statistical properties of the corpus, which were key to earlier techniques, are neglected, therefore global patterns may be overlooked. To mitigate this, the *GloVe* [6] approach generates word embeddings by constructing an explicit word co-occurrence matrix for the entire corpus, such that the dot product of two word embeddings is equal to log of the number of times these words co-occur (within a defined window).

Despite their ability to preserve semantic relationships, Word2Vec and GloVe do not accommodate polysemy, which describes the co-existence of alternative meanings for the same word. In addition, they cannot generalise to words that were not specifically included in the training set. *Bojanowski et al. (2017)* [2] describes a different vectorization technique used in Facebook's open-source fastText library. In the fastText approach, each word is represented as a bag of character n-grams which enables it to capture meaning for substrings such as prefixes and suffixes. In order to produce word embeddings for out-of-vocabulary tokens i.e. words not included in the training set, fastText composes vectors for the substrings that form the word. However, like Word2Vec and GloVe, FastText produces a single embedding for each word - hence, it cannot disambiguate polysemous words.

Contextualized Word Embedding – In pursuit of a more robust approach, we look to deep neural language models. Peters et al. (2018) [7] introduced the *ELMo* model, and showed that the addition of ELMo to existing models can markedly improve the state-of-the-art in a number of NLP tasks. ELMo utilizes a bi-directional LSTM *(long short-term memory)* model, concatenating the left-to-right and right-to-left LSTM in order to produce deep contextualised word representations. Like fastText, they are character based, therefore robust representations can be constructed for out-of-vocabulary tokens. However, rather than 'looking-up' pre-computed embeddings, ELMo generates them dynamically. Hence, it can disambiguate the sense of a word given the context in which it was found.

Transfer learning is a technique whereby instead of training a model from scratch, we use pre-trained models and finetune them for specific NLP tasks. BERT uses a bi-directional transformer BERT representations are

BERT [3] employs masked language modeling 15% of the words were hidden and inferred by their position

B Feature extraction from word embeddings

have seen success in other NLP classification tasks, such as ... In the domain of sarcasm detection,

A growing trend appears to be extraction

Joshi et al. (2016) explores the usefulness of word-embedding based features in sarcasm detection.

Riloff et al. (2013) presents a bootstrapping algorithm for detecting a specific style of sarcasm - contrasting a negative situation with a positive sentiment. This is an oversimplification of sarcasm, hence it is unlikely to capture less explicit formulations.

C Data collection and pre-processing

As sarcasm is used in social, informal conversation. Social networks as a huge database of informal language.

Conversations on social media tend to be informal, however this will introduce noise into the dataset in the form of erroneous spelling,

I expect to encounter some difficuties when it comes to processing informal and erroneous spelling. Additionally, in a dataset of tweets, text is fairly short and could be lacking essential context.

In terms of data quality, it may not be adequate just to analyze the text independently. Some contextual information about the author of the tweet may be useful to add more meaning.
The previous sections have highlighted the ambiguities within sarcasm. This method is not without its flaws. Some people may incorrectly label their tweets as sarcastic, leading to noise in the dataset. Similarly, the non-sarcastic tweets are a subset of all other tweets that are not labelled with sarcasm and it goes without saying that a small percentage of these tweets will in fact be sarcastic - another source of noise. Likewise, numerous studies have shown that humans disagree over labelling statements as sarcastic.

D Model Performance Evaluation

Lastly, evaluating the successes and failures of the trained model is a critical step. A simple approach is to use accuracy which refers to the proportion of data that is correctly labelled as either sarcastic or non-sarcastic. However, sarcasm is a minority class therefore on an unbalanced dataset we could achieve high accuracy by simply labelling every statement as non-sarcastic. In an attempt to mitigate against this, I will instead form a conclusion based on the F_1 score i.e. the harmonic mean of precision and recall, where scores range from 0 (worst) to 1 (best). This metric has faced some criticism for giving equal weight to both precision and recall [4], therefore I will consider both measures separately, as well as in combination.

$$\text{precision} = \frac{\text{true positives}}{\text{true positives + false positives}} \qquad \text{recall} = \frac{\text{true positives}}{\text{true positives + false negatives}}$$

In the domain of sarcasm detection, precision refers to the proportion of the classified-sarcastic data that is *truly sarcastic* i.e. how many of the positives are true positives, and recall describes the proportion of the truly sarcastic data in the corpus that is *classified* as such i.e. how many true positives are labelled as positives.

References

[1] Gavin Abercrombie and Dirk Hovy. Putting sarcasm detection into context: The effects of class imbalance and manual labelling on supervised machine classification of twitter conversations. In *Proceedings of the ACL 2016 Student Research Workshop*, pages 107–113, 2016.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] David Hand and Peter Christen. A note on using the f-measure for evaluating record linkage algorithms. *Statistics and Computing*, 28(3):539–547, 2018.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[6] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[7] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[8] Cambridge University Press. Sarcasm: meaning in the cambridge english dictionary. https://dictionary.cambridge.org/dictionary/english/sarcasm, 2019. Last accessed: 23 Oct 2019.

[9] Stephen E Robertson and K Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146, 1976.