Detecting the Target of Sarcasm is Hard: Really??

Pradeesh Parameswaran[1], Andrew Trotman, Veronica Liesaputra, David Eyers
Department of Computer Science, University of Otago, New Zealand

Abstract

Sarcasm target detection (identifying the target of mockery in a sarcastic sentence) is an emerging field in computational linguistics. Although there has been some research in this field, accurately identifying the target still remains problematic especially when the target of mockery is not presented in the text. In this paper, we propose a combination of a machine learning classifier and a deep learning model to extract the target of sarcasm from the text. First, we classify sarcastic sentences using machine learning, to determine whether a sarcastic sentence contains a target. Then we use a deep learning model from Aspect-Based Sentiment Analysis to extract the target. Our proposed system is evaluated on three publicly available data sets: sarcastic book snippets, sarcastic tweets, and sarcastic Reddit comments. Our evaluation results show that our approach achieves equal or better performance compared to the current state-of-the-art system, with an 18% improvement on the Reddit data set and similar scores on the Books and Tweets data sets. This is because our method is able to accurately identify when the target of sarcasm is not present. The primary challenge we identify, that is hindering the creation of a high accuracy classifier, is the lack of consistency among human annotators in identifying the target of sarcasm within standard ground-truth data sets.

Keywords: sarcasm detection, target sarcasm detection, deep learning

[1]Corresponding author. E-Mail Address: mailto:pradeesh@cs.otago.ac.nz

1. Introduction

We use language as a tool to convey our emotions (Sabbagh, 1999) and our thoughts (Cheang & Pell, 2008) to another person. An example of how we convey our emotions is through the usage of verbal irony (Sperber, 1984), where the intended meaning differs from the supposed meaning. Another is through the use of sarcasm, an utterance that on the surface appears to be in context, but that is intended to mock or ridicule a target (Kreuz & Glucksberg, 1989). In our daily lives, we convey sarcasm through our voice, tone and body language.

The use of sarcasm is much more prevalent on the Internet than in face-to-face interactions (Chandrasekharan et al., 2017) as people are able to mask their identity and post sarcastic comments with almost no consequences. Sarcasm poses a challenge for sentiment analysis models which can incorrectly classify the sentiment of sarcastic sentences as positive (Maynard & Greenwood, 2014). Past works have focused on various ways of identifying sarcastic and non-sarcastic text (Bamman & Smith, 2015; Rajadesingan et al., 2015; Amir et al., 2016; Hazarika et al., 2018). However, little work has been done on identifying the *target* of sarcasm (Joshi et al., 2016a; Patro et al., 2019; Parameswaran et al., 2019) in a sarcastic text.

Joshi et al. (2016a) define sarcasm target identification as *identifying the target of ridicule of a given sarcastic text.* The two main challenges of this task are: determining if the target of sarcasm is present in the text, and identifying one or more targets of sarcasm from multiple candidate phrases.

Example text	Target(s) of sarcasm
"This phone heats up so much that I recommend Gordon Ramsay to use it as a cook-top"	"phone"
"He is good at cooking as Taylor Swift is at relationship"	"He", "Taylor Swift"
"Oh I suppose durian ate the kiwi fruit?"	OUTSIDE

Table 1: Sarcasm Target Examples

Table 1 presents some examples of sarcasm from the data sets we use. In the first example, "*This phone heats up so much that I recommend Gordon Ramsay to use it as a cook-top*", there are multiple potential targets: "*Gordon Ramsay*", "*phone*", and "*cook-top*" but the *subject* that is being made fun of is the "*phone*". Some sarcastic sentences contain multiple sarcasm targets. In the second example, "*He is good at cooking as Taylor Swift is at relationship*", the targets that are being made fun of are "*He*" and "*Taylor Swift*" as Taylor Swift is known for having short-lived relationships. There are also instances where the target of the sarcasm is not in the sentence (so-called OUTSIDE cases) such as the phrase "*Oh I suppose durian ate the kiwi fruit?*". Further domain knowledge is needed to know that a fruit cannot eat other fruit.

There are many reasons that it is important to automatically detect the presence and target of sarcasm. Upsetting to us, trolls and cyberbullies use sarcasm to bully their victims through social media, and this has a lasting psychological impact on the victims (Sanfilippo et al., 2018). The ability to automatically identify and block such messages in a timely manner could help reduce the prevalence and the effects of this negative behaviour.

The first step to reducing the amount of person-targeted sarcasm is to identify the presence of sarcasm. The second step is to identify the target of that sarcasm. The third step is determining whether the target is an individual (such as Gordon Ramsay), or whether it is an inanimate object (such as a phone or a cook top). It is the second step that we examine herein. The first step (presence of sarcasm) has already been extensively examined in the literature (Eke et al., 2020; Joshi et al., 2017). We leave the third step for future work.

The pioneering research in identifying the target of sarcasm was done by Joshi et al. (2016a). They used a simple rule-based approach in order to detect and extract the target. Their rules identify and extract probable target pronouns, named entities and gerunds. Patro et al. (2019) extended this work by using Long Short Term Memory networks (LSTM) together with Linguistic Inquiry and Word Count (LIWC) (Chung & Pennebaker, 2013) which helped to increase the overall performance. However, identifying the absence of a target,

and multiple targets still remains difficult (Parameswaran et al., 2019). This can be attributed to the fact that often, when humans express sarcasm, the target is not used explicitly and there are no given fixed set of linguistic rules pointing at the target.

Towards our goal of identifying the target of sarcasm, we investigate two research questions:

- Can we accurately identify the target of sarcasm in a sarcastic text?
- What factors affect the performance of a system that detects the target of sarcasm?

Unlike prior work where the entire problem of identifying the presence of, and then extracting the target of, sarcasm is treated as a single classification task, we treat the problem as two smaller tasks. Our first task is to identify OUTSIDE sentences, where the target of sarcasm is not mentioned in the text, from INSIDE sentences, where the target of sarcasm is in the text. This is done through the use of various embedding techniques (Radford et al., 2019) applied at the sentence level and using an ensemble of classifiers. We theorise that by first filtering for the presence of the target, we can improve the quality of the identification of the target. In our second task, we utilise various deep learning models that are used in Aspect-Based Sentiment Analysis (ABSA) in order to identify the target, but only in sentences containing an INSIDE target.

We conduct extensive experiments and validation with data sets used in previous studies (Joshi et al., 2016a; Patro et al., 2019; Molla & Joshi, 2019). Our results show that the method we use performs at least as well as the state-of-the-art, with an 18% improvement on the Reddit data set, but similar scores on two other data sets. Our findings also highlight some of the key challenges that hinder the creation of a classifier with high accuracy.

We summarise the contributions of this work as follows:

- We propose a model for detecting the presence of the target of sarcasm. That model uses a combination of a machine learning classifier and deep

learning. To the best of our knowledge, this is a unique approach.

- We conduct extensive experiments that evaluate different combinations of classifiers and deep learning models on publicly available data sets. We compare the performance of our system with the current state-of-the-art for this task (Patro et al., 2019).

- We analyse some of the factors that lead to the observed performance in sarcasm detection.

- We suggest both specific and general ways to improve performance in this task.

The remainder of this paper is organised as follows. Section 2 discusses prior work in identifying the target of sarcasm. We present our model in Section 3, and share the details of our experimental setup in Section 4. The results are presented in Section 5. Finally, Section 6 summarises our work and discusses future research.

2. Related Work

In this section, we first review studies in sarcasm detection that explore how to detect sarcasm (a prerequisite for our sarcasm target identification system). Then, we review the current work on our topic of identifying the target. We also review the notion of intended versus perceived sarcasm, as well as aspect-based sentiment analysis and transfer learning which provides us with insights on how to create a novel system that could accurately detect the target of sarcasm.

2.1. Sarcasm Detection in Text

In the past, sarcasm detection was formulated as a classification task (Eke et al., 2020; Joshi et al., 2017), whereby given a piece of text, the goal is to determine if it is sarcastic or not. For instance, the phrase "*love being ignored*" is sarcastic and "*today was a sunny day*" is not sarcastic. In broad terms, the

classification process is done either using feature engineering or through deep learning.

With feature engineering, oftentimes it requires various features such as bag of words (Fersini et al., 2015), irony markers (Ghosh & Muresan, 2018) and Part of Speech (PoS) tagging (Bharti et al., 2015) which are then fed into a machine learning classification system such as a Support Vector Machine (SVM) or Random Forest (RF) to perform the classification (Ling & Klinger, 2016). Some studies incorporate aspects other than text, such as user profiles (Bamman & Smith, 2015; Ghosh & Muresan, 2018).

Some researchers instead leverage the advancement of deep learning to tackle this classification problem through the use of Long-Term Short Term Memory, Convolutional Neural Networks (CNNs) or Deep Neural Networks (DNNs) (Ghosh & Veale, 2016; Agrawal & An, 2018; Hazarika et al., 2018; Martini et al., 2018; Liu et al., 2019). Recently, Liu et al. (2019) used a deep learning model and obtained an F_1 score of 0.993 on the sarcastic Tweets data set. Their method of using auxiliary features such as emoji, and combining several neural networks substantially improves upon earlier work, which only managed to achieve an F_1 score of 0.75 (Buschmeier et al., 2015). We consider this to be a separate problem from sarcasm target detection, and with F_1 scores above 0.99, it is, perhaps, a nearly solved problem. In this work, we assume that the text is classified as sarcastic before we identify the target.

2.2. Identifying Target of Sarcasm in Text

The problem of identifying the target of sarcasm in text was first introduced by Joshi et al. (2016a). The goal of this task is to accurately identify the target of sarcasm that is known to exist in the given text. As stated in Section 1, this is rather a complex problem to tackle due to the fact that there can be multiple sarcasm targets or there can be an absence of sarcasm targets.

Joshi et al. (2016a) introduces two separate data sets (Book Snippets and Tweets) which they tagged as sarcastic themselves. They then asked three qualified judges with a background in linguistics to identify the target of the sarcasm

in the text. They also proposed a way to automatically identify the target of sarcasm using a rule-based approach which uses features such as capitalisation, PoS tags and punctuation. Their models achieved a Dice Score of 0.369 and 0.326 for the books and the Tweets data set respectively. Our informal experimentation with their rule-based approach suggests that the low Dice score could be attributed to the rigidness of rule-based systems: for example, a rule specifying that a word ending with an exclamation mark is a potential target may be right most of the time, but certainly is not right all of the time.

This prompted Patro et al. (2019), to address the gaps of rule-based system by applying deep learning. They used Temporal Dependence-Based Long Short-Term Memory with linguistic features from the Linguistic Inquiry and Word Count (LIWC) library (Chung & Pennebaker, 2013). This resulted in a Dice Score of 0.8404 and 0.8766, a significant improvement on the work of Joshi et al. (2016a). Despite the improvements, the main weakness in their study is that their model is still not able to predict accurately when the target of sarcasm is not present, or when there are multiple targets of sarcasm. This could be attributed to Joshi et al. (2016a)'s data set being relatively small: it contains around 500 sarcastic example texts.

In order to further encourage research in this area, the Australasian Language Technology Association (ALTA)[2] organised a shared challenge with a new data set for the problem (Molla & Joshi, 2019). This new data set is roughly twice the size of that used in previous studies, and includes a variety of discussions from politics to gaming. The details of the data set are discussed in-depth in Section 3. There were no winners declared in ALTA's shared challenge as none of the teams beat the baseline score set by the organisers. This motivated us to investigate the problem further.

[2]http://www.alta.asn.au/

2.3. Intended versus Perceived Sarcasm

What constitutes sarcasm varies from one person to another and is based on their background, educational level and even cultural influences. For example, the sentences in Table 1 can be interpreted differently. If you do not know that Taylor Swift has a reputation for being in short-term relationships, or if you are a fan of Taylor Swift, the text might not be considered sarcastic. Consequently, judging sarcasm can be highly subjective (Rockwell, 2000). Consider the following sentence: "*Oh it is such a lovely day*". A Malaysian person (e.g., an author of this work) may not find it sarcastic whereas a British person (e.g., a different author of this work) may find it sarcastic. This obviously results in low annotator agreement levels across curated data sets for sarcasm detection (Joshi et al., 2016b; González-Ibáñez et al., 2011).

González-Ibáñez et al. (2011) found that the agreement level between human annotators identifying sarcastic sentences (and measured using Cohen's Kappa) was only 0.50—which is low compared to other tasks such as image classification where it is unsurprising to see scores as high as 0.80 (Ribeiro et al., 2019). Waseem (2016), studying binary classification on different data, found that expert annotators have a relatively low Kappa score of 0.37 when it comes to classifying sarcastic texts. Even when annotators are from a similar cultural and education background, humans tend to have bias when it comes to what constitutes sarcasm. Oprea & Magdy (2019) measured the agreement level of independent annotators against authors and report a Kappa agreement level of 0.36. These agreement levels might be low because, or even simply show that, it is difficult for a third person to identify and understand sarcasm even if it is deliberate (Gibbs Jr et al., 1994; Giora, 2003; Pexman, 2008). Olkoniemi et al. (2019), found that despite participants requiring more processing effort to comprehend sarcastic texts, they do not always understand the intended meaning. They suggested that this could be because participants failed to consider context. This raises questions about maximum potential machine performance, the quality of the gold-standard test sets, and more fundamentally how to build reliable and reusable data sets (we leave this for future work).

This subjectivity is compounded further when we try to identify the target of sarcasm. One participant in the ALTA challenge highlighted collection-wide inconsistencies in what the annotators identified as the target of sarcasm (Parameswaran et al., 2019). For instance, when there are multiple pronouns in a text, in some cases the annotators only picked one of the pronouns as the target but in other cases, they selected all of the pronouns as the target. Consider the following text, "*America is a free country. It is if you can afford it*". The ground truth selected by annotators is "*America*". However, we can see that the pronoun "*It*" refers to the same target and thus should be identified as the target as well. We discuss this topic further in Section 5.4.

2.4. Aspect-Based Sentiment Analysis

Another relevant study for our field is in the field of Aspect-Based Sentiment Analysis. Aspect-Based Sentiment Analysis focuses on a text with which multiple sentiments are associated (Liu, 2015). For instance, consider the following sentence "*Support was great but UI was confusing*". The phrase *"support was great"* contains a positive sentiment towards the customer support that was received by the person and the negative aspect is towards the User Interface (UI) of the product itself. Traditional single-sentiment analysis software might classify it as negative—but in ABSA the various different aspects are taken into consideration.

There are two approaches to ABSA. One uses lexicon-based strategies (Qiu et al., 2011; Liu, 2015) and the other uses deep learning models such as TD-LSTM and LSTM (Wang et al., 2016; Chen et al., 2017; Tang et al., 2016). Both lexicon and deep learning models have their own advantages and disadvantages. For a lexicon-based strategy, both sentiment and domain-based lexicons are required, which requires significant use of resources to curate. As for deep learning, a fairly large data set is required and a huge amount of training time is often required (Tao & Fang, 2020a).

2.5. Transfer Learning

Transfer learning helps to address the shortcoming of deep learning models that require a long time to train. Transfer learning uses domain-specific data to fine-tune pre-trained models. This is common practice in the field of computer vision (Shin et al., 2016) and this has started to emerge in the field of Natural Language Processing (NLP). One of the successful cases of transfer learning involves fine-tuning a language model such as word embeddings pre-trained on a huge corpora (Pan & Yang, 2010). Examples of such use in NLP include language identification of low-resource languages (Zoph et al., 2016), and bilingual speech separation (Wang & Zheng, 2015).

Tao & Fang (2020b) applied transfer learning to ABSA. They achieved this by fine-tuning BERT, a pre-trained language model, and applying it to the Yelp data set (Zhang et al., 2015). They found that by utilising transfer learning, they could achieve an accuracy score of 0.6415 as opposed to just 0.4112 using an approach that did not employ transfer learning. Zhang et al. (2019b) formulate irony detection as a transfer learning task where supervised learning on irony labelled text is then enriched with knowledge transferred from external sentiment analysis resources. Jia et al. (2019) also used transfer learning in detecting sarcasm in Chinese.

Our task is similar to ABSA: each word in our sarcastic sentence may or may not be the target of sarcasm. The main difference with our task is that unlike ABSA, our target may be completely absent from the given text. But we hypothesise that transferring a pre-trained language model will help to improve the overall performance of identifying the target of sarcasm.

3. Methodology

The architecture of our system is shown in Figure 1, and is described in detail, below. The input to the first phase is a sarcastic text that is assumed to have come from a sarcasm / not sarcasm binary classifier. The text is then fed through a pre-trained embedding system. It is then passed to our classifier,

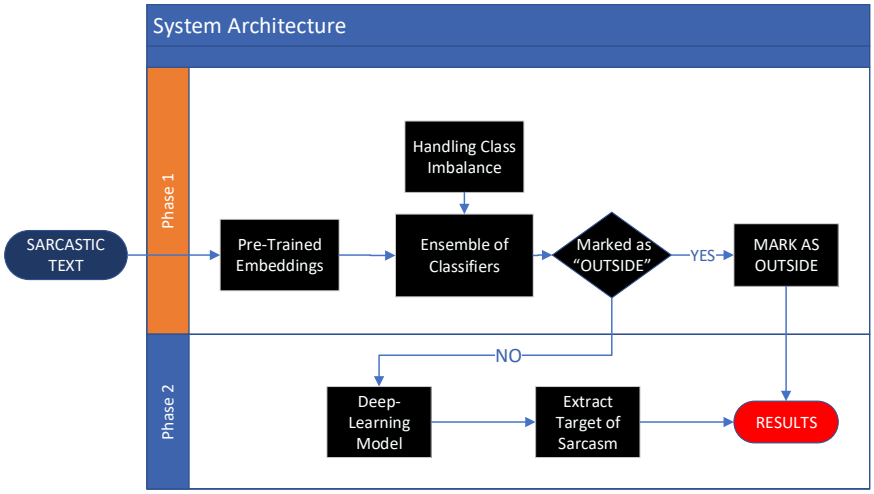


Figure 1: High-level system architecture

which classifies the text as INSIDE or OUTSIDE. If the text is classified as INSIDE, it is passed to the second phase of our system. In the second phase, we use a deep learning model to extract the target of the sarcasm.

3.1. Target Detection (Phase 1)

The purpose of our first phase is to determine whether the target of sarcasm is present in the given sarcastic text or not. We model this behaviour by utilising sentence embeddings to capture the semantics of the sentence which we then use to detect the sarcasm target.

3.1.1. Sentence Embedding

Following the success of pre-trained language models in ABSA (Tao & Fang, 2020b), we generate sentence embeddings using the pre-trained language models summarised in Table 2. These embeddings were selected based on the analysis carried out by Rogers et al. (2018) regarding their performance at various language tasks. They are also freely available in the huggingface[3] library that we

[3]https://github.com/huggingface

Model	Pretrained Model
ALBERT (Lan et al., 2019)	albert_base,albert_large,albert_xlarge
BERT (Devlin et al., 2019)	bert_base_cased,bert_base_uncased,
	bert_base_large_cased,bert_base_large_uncased
ELMO (Peters et al., 2018)	elmo
FASTTEXT (Joulin et al., 2017)	fasttext-gnews
GLOVE (Pennington et al., 2014)	crawl_42b,twitter200d
GPT-2 (Radford et al., 2019)	gpt,gpt2
Transformer-XL (Dai et al., 2019)	XLTransformer
Universal Sentence Encoder (USE) (Cer et al., 2018)	use_elm,use_transformer

Table 2: Summary of pretrained embedding models

use.

1. **GLOVE** (Pennington et al., 2014)—GLOVE is a very well known model of traditional word embedding techniques which aims to learn a global word embedding matrix. There are two types of pre-trained GLOVE embeddings:
   - crawl42b—trained on Common Crawl[4] (1.9M vocab, uncased, 300 dimension vectors)
   - twitter200d—trained on two billion Tweets (2M vocab, uncased, 200 dimension vectors)
2. **FastText** (Joulin et al., 2017)—FastText uses the same traditional word embedding model as GLOVE. However, it also includes character-n-gram which helps with out-of-vocabulary words.
   The only available FastText embedding is:
   - fasttext-gnews—trained on Google News Crawl (1M vocab, uncased, 300 dimension vectors)
3. **GPT2** (Radford et al., 2019)—GPT2, like its predecessor GPT, uses a transformer architecture. It adapts a two-stage learning paradigm. The

[4]https://commoncrawl.org/

first stage is unsupervised pre-training using a language modelling objective and in the second stage, a supervised fine-tuning is done. The key difference between GPT1 and GPT2 is that GPT2 is trained on a larger and more diverse data set from various corpora including books, encyclopaedias, news articles and message boards (Radford et al., 2019). There is only one pre-training of GPT and GPT2 respectively:

- gpt—trained on diverse corpora (110M vocab, uncased, 768 dimension vectors)
- gpt2—trained on diverse corpora and Internet message boards (335M vocab, uncased, 768 dimension vectors)

4. **ELMO** (Peters et al., 2018)—ELMO extends the traditional word embedding model by utilising context-dependent representation. A forward LSTM and a backward LSTM layer are applied to encode the left and right context of the sentence. Unlike other models, ELMO does not use a transformer, instead it uses LSTM. ELMO has only one pre-trained embedding model:

   - elmo—trained on Wikipedia and Google News crawl (1B vocab, uncased, 768 dimension vectors)

5. **BERT** (Devlin et al., 2019)—BERT is a transformer-based model which uses a masked language modelling (MLM) whereby some of the tokens are randomly masked. The objective of BERT is to predict the randomly masked sequence. Given two input sentences, BERT uses a next-sentence-prediction (NSP) to predict if the second sentence is the same as the first. This makes it interesting for our use case as people often utter sarcastic remarks after a certain set of sentences (Jorgensen, 1996).
   The four pre-trained BERT embedding models that we use are :

   - bert_base—trained on Wikipedia (1.9M vocab, uncased, 768 dimension vectors)
   - bert_base_cased—trained on Wikipedia (10M vocab, case sensitive, 768 dimension vectors)

- bert_base_large_cased—trained on Wikipedia along with BookCorpus[5] (5.0M vocab, case sensitive, 1024 dimension vectors)
- bert_base_large_uncased—trained on Wikipedia along with BookCorpus (5.0M vocab, uncased, 1024 dimension vectors)

The key difference between them is the the size of the corpora that was used to build the embedding and the case sensitivity.

6. **ALBERT** (Lan et al., 2019)—ALBERT is essentially an optimised version of BERT which uses parameter-reduction techniques (cross-layer parameter sharing and factorised embedding parameterisation) to lower memory consumption and speed up training. Also, ALBERT does not utilise NSP objectives which makes it more adaptable to other tasks.

   There are three types of pre-trained ALBERT embeddings :

   - albert_base—trained on Wikipedia (1.9M vocab, uncased, 768 dimension vectors)
   - albert_large—trained on Wikipedia (1.9M vocab, uncased, 1024 dimension vectors)
   - albert_xlarge—trained on Wikipedia (1.9M vocab, uncased, 2048 dimension vectors)

   The main key differences between these embeddings is the vector dimensionality.

7. **Transformer-XL** Dai et al. (2019)—Transformer-XL addresses one of the challenges faced by language modelling, which is fixed-length contexts. Transformers have the potential of learning longer-term dependencies but the fixed-length content problem limits the potential of a transformer. Transformer-XL uses a modified transformer model to address the shortcoming of a normal transformer by adding a memory segment between one layer to another layer.

   The only pre-trained Transformer-XL embedding available is:

[5] https://yknzhu.wixsite.com/mbweb

- transfo-xl-wt103—trained on Wikipedia (1M vocab, uncased, 1024 dimension vectors)

8. **Universal Sentence Encoder (USE)** (Cer et al., 2018)—Universal sentence encoder is available both as a transformer-based encoder model and as a Deep Averaging Network (DAN). Unlike the other embeddings we use, which produce word-level embeddings, USE generates embeddings for an entire sentence. USE is trained on various different sources which includes Wikipedia, news sites, question and answer websites, and discussion boards.

   The two versions of pre-trained USE embedding are:

   - use_elm (Version 1/DAN)—trained on various sources (774M vocab, uncased, 512 dimension vectors)
   - use_transformer (Version 2/Transformer)—trained on various sources (335M vocab, uncased, 512 dimension vectors)

With the exception of USE, to provide a holistic representation of the sentence, we followed the method outlined by Arora et al. (2017) of averaging the embedding of each word to get a sentence's embedding. We use the default configuration of huggingface for each of the embedding techniques. We benchmark the performance of each of the embedding techniques described here in detail in section 5.1.1. In addition, we measure the computational cost of each of the embeddings, which is described in detail in section 5.3.1.

3.1.2. Handling Class Imbalances

Oversampling the minority class through the Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002) helps to improve the accuracy and prediction of the classifier when there is a class imbalance in the data (Song et al., 2016; Sarakit et al., 2015). An alternative is to balance the weight of each class (King & Zeng, 2003), which has shown good results in classifying sentiments (Maipradit et al., 2019). The class balancing equation is given by

$$W_y = \frac{\mu}{(\omega \cdot \alpha_y)} \tag{1}$$

where μ is the total number of samples, ω is the number of classes, and α_y is the number of samples in class y. We evaluate the difference between these two class balancing techniques in Section 5.1.2.

3.1.3. Ensemble Classifier

The sentence embedding is fed in to an ensemble of classifiers consisting of Random Forest (RF), Support Vector Machine (SVM), and Logistic Regression (LF). Ensemble classification was chosen because a single classifier may not be able to generalise well on unseen data (Perikos & Hatzilygeroudis, 2016). Our ensemble classifier uses a majority voting approach to make the final classification based on the outputs of the three classifiers.

One of the challenges in building an ensemble classifier is the task of fine tuning each of the classifiers. We used the work of Feurer et al. (2015), which exploits Bayesian optimisations, to help reduce the time required to find good hyper-parameters for individual classifiers. We discuss the results of our hyper-parameter tuning in Section 4.4.

3.2. Target Extraction (Phase 2)

Once we have separated the OUTSIDE from the INSIDE sentences, our next task is to identify the target of sarcasm in INSIDE sentences. We apply deep learning and compare the performance of the following deep learning models:

1. **Target Dependent Long Term Short Memory (TD-LSTM)** (Tang et al., 2016)—The idea of this model is to use the preceding and the following context surrounding the target sarcastic word as a feature representation. Two LSTM networks are used for this, the left LSTM neural network consists of the preceding sentence along with the potential target and the right LSTM neural network consists of remaining context along with the potential target. The left LSTM network runs from left to right

and the right LSTM network runs from right to left. These LSTM networks are able to learn the semantics of the sentence (Tang et al., 2016).

2. **Target Connection Long Term Short Memory (TC-LSTM)** (Tang et al., 2016)—This is a modification of TD-LSTM. The main difference between TC-LSTM and TD-LSTM is that in TD-LSTM the input at each position includes the embedding of the current word, whereas TC-LSTM contains the concatenation of the set of words preceding and following the target. We expect the concatenation of the words will result in a higher accuracy than TD-LSTM.

3. **Recurrent Attention Network on Memory (RAM)** (Chen et al., 2017)—This uses a bi-directional LSTM in order to produce a memory slice. The memory slice is used to address the shortcoming of the TC-LSTM model (not being able to capture the target word if it is far away from the target). These memory slices are weighted according to the position of the target. The input of RAM is the entire sarcastic sentence and the distance of potential targets of the sarcasm. Then, to classify the target of sarcasm the results are combined non-linearly with a Gated Recurrent Unit (GRU).

TD-LSTM is the current state-of-the-art system for identifying the target of sarcasm (Patro et al., 2019). However, we believe that TC-LSTM and RAM should perform better as the model should be able to understand the nuances of the sentences and identify the target more accurately.

Figure 2 shows the architecture diagram of our deep learning model. Our setup closely follows Patro et al. (2019) whereby we define a sarcastic sentence as a sequence of words $\{w_1, w_2, w_3, \ldots, w_n\}$ and each word, w_k, is a potential target of the sarcasm. We set the maximum length of the sentence to 250 words. Because TD-LSTM and TC-LSTM require a left and right context for each word, we appended a dummy word w_s at the beginning and the end of the sentence. Thus, for each word w_k in a sentence, we have a set of all of the words prior to w_k, i.e., $P_k = \{w_s, w_1, \ldots, w_{k-3}, w_{k-2}, w_{k-1}\}$, and a set of all

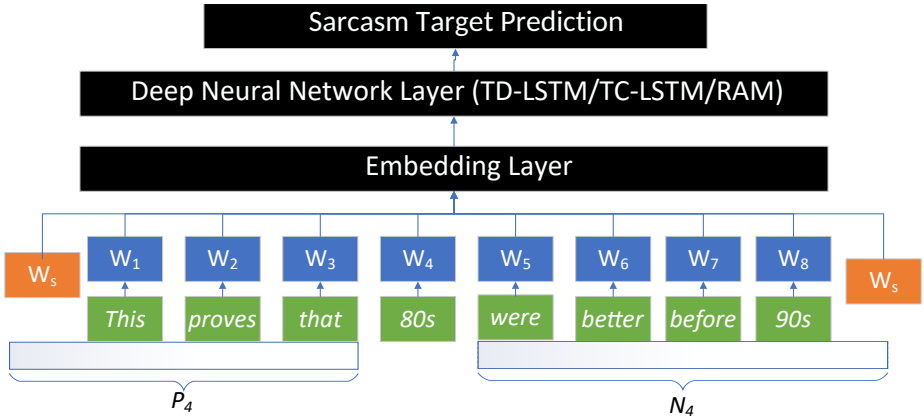


Figure 2: Overall System Architecture of the Deep Learning Model

the words after w_k, i.e., $N_k = \{w_{k+1}, w_{k+2}, w_{k+3}, \ldots, w_n, w_s\}$. k is the position of a word in a sentence, i.e., $k = \{1, 2, 3, \ldots, n\}$, and n is the total number of words in the sentence.

In the example shown in Figure 2, the sentence is "*This proves that 80s were better before 90s*" and the current potential target of the sarcasm in context is "*80s*". In this case, $k = 4$, and so $w_4 =$ "80s", $P_4 = \{w_s$, "This", "proves", "that"$\}$ and $N_4 = \{$"were", "better", "before", "90s", $w_s\}$.

We then proceed to feed each of the possible sequences of words from each of the input sentences to the embedding layer to initialise it. We used the same embeddings as Patro et al. (2019) which is a combination of ELMO and GLOVE, as they already found that this combination yielded the best result in identifying the target of sarcasm. Once the embeddings are initialised, they are passed to the deep neural network layer. The input to the neural network varies depending on the type of neural network:

- **TD-LSTM**—We pass in the set of left context and right context. For the right context, we pass in the current word and the set of words after the current word, i.e., $\{w_k\} + N_k$. While for the left context, we pass in the

current word and the words prior to the current word, i.e., $P_k + \{w_k\}$.

- **TC-LSTM**—We concatenate the words preceding and following the current word together, and append the current word at the end before we pass it to TC-LSTM, i.e., $P_k + N_k + \{w_k\}$.

- **RAM**—We pass in the entire sentence, i.e., $P_k + \{w_k\} + N_k$, and append it with the distance of the first word w_1 to the current word w_k, which is $[k]$.

For all three deep learning models, we used a softmax activation function to classify the current word as the target of sarcasm or not.

We did not remove stop words from our sentences, as stop words were not removed during evaluation and assessment of the ALTA challenge (Molla & Joshi, 2019). For example, in the data set sentence, "*i found it in the bin*", the target of sarcasm identified by the annotators is "*the bin*". If we were to remove stop words from this sentence then our model might only detect "*bin*" and thus would have its performance limited.

4. Experiments and Results

4.1. Data sets

We use three public data sets: two released by Joshi et al. (2016a) and one released by Molla & Joshi (2019). These represent three distinct types of data: sarcastic book snippets (*Books*), sarcastic tweets (*Tweets*), and sarcastic Reddit comments (*Reddit*). Each data set contains two fields: the sarcastic text, and the target of the sarcasm. If there are multiple targets, each target is separated by a space. If the text does not have any target, it is marked OUTSIDE.

Table 3 presents some statistics of the data sets including the number of sentences, the average sentence length, and the average length of a target. The percentage of sarcasm targets labelled OUTSIDE varies across the data sets. Both *Books* and *Tweets* have very low percentages of OUTSIDE cases compared to *Reddit*.

	Tweets	*Books*	*Reddit*
Count	224	506	950
Average sentence length	13.06	28.47	25.30
Average sarcasm target length	2.08	1.6	2.8
% of OUTSIDE	10%	5%	35%

Table 3: Statistics of data sets

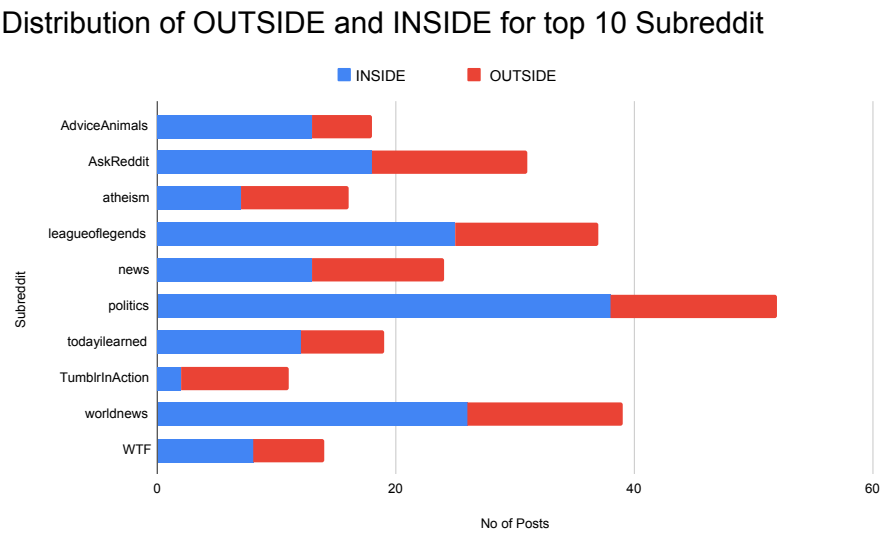


Figure 3: Distribution of OUTSIDE and INSIDE of top-10 popular Subreddits

To investigate the reason behind the high percentage of OUTSIDE in *Reddit*, we used the Khodak et al. (2018) Reddit Corpus to obtain information about each Subreddit[6] in the data set. There are 128 Subreddits represented in the data set. We observe a higher proportion of OUTSIDE cases to INSIDE cases in niche Subreddits, such as *tumblrinaction* and *atheism*, than in general purpose Subreddits such as *worldnews* and *adviceanimals*. Figure 3 shows the 10 Subreddits with the largest number of posts along with the distribution of OUTSIDE

[6]A forum dedicated to a specific topic on the website Reddit (`https://www.reddit.com`)

and INSIDE sentences. Although all Subreddits assume readers have the domain knowledge necessary to understand the forum content, the knowledge required in niche Subreddits is specific and not necessarily common knowledge. Thus without the proper domain knowledge and context, assessors may not be able to identify the target of sarcasm very easily or accurately (Justo et al., 2014).

4.2. Baselines

We use state-of-the-art TD-LSTM (Patro et al., 2019), TC-LSTM (Tang et al., 2016) and RAM (Chen et al., 2017) as our deep learning baselines. We also include a Naive Bayes (NB) Classifier as an additional baseline. We believe that, despite Naive Bayes being a simple classifier, the ability to classify words based on their prior probabilities should be competitive. Naive Bayes is simple to implement and quick to run, and so also offers quick insights into whether this is a simple or a difficult problem.

4.3. Metrics

To measure the effectiveness of our system, we use three different metrics: Dice Score, F_1 Score and Balanced Accuracy. We use Dice Score in order to measure the accuracy of identifying the target of sarcasm as it was used in the prior works of Joshi et al. (2016a); Patro et al. (2019); Molla & Joshi (2019).

The Dice Score, $D(A, B)$ is given by

$$D(A, B) = 2 \times \frac{A \cap B}{|A| + |B|} \tag{2}$$

where A are predicted words generated by the system and B are words identified by the assessors. If the target of sarcasm is not in the text the assessments are marked OUTSIDE and so the task is to produce this as the system output—this is standard practice in this field of research and not a decision made by us.

In addition to the Dice Score, we use the Balanced Accuracy Score and macro F_1 Score to measure the performance of our target detection classifers. Balanced Accuracy was selected as it provides a holistic view of the accuracy of the classification by looking at the accuracy of both of the classes. If we were to

report the standard accuracy metric, it may give a false impression that model is performing well due to the class imbalances (Buda et al., 2018), especially due to the low occurrence count of OUTSIDE in the collections.

The equation for balanced accuracy score is given in Equation 3 and F_1 is defined in Equation 4

$$BalAcc = \frac{\frac{TP}{TP+FP} + \frac{TN}{TN+FN}}{2} \tag{3}$$

$$F_1 = \frac{2TP}{2TP + FP + FN} \tag{4}$$

where TP is True Positive (OUTSIDE), TN is True Negative (INSIDE) and FN and FP are False Negative and False Positive respectively.

4.4. Experimental Setup

In order to train and evaluate the model, we took a random 60/10/30 split of the data with 60% for training, 10% for validation (including hyper-parameter tuning) and 30% for testing. Since a single such split is prone to bias, we performed this operation 10 times (using 10 different random seeds), and report the mean performance. We did not perform 10-fold cross validation because some of our data sets are small and doing so would result in evaluation on tiny sets of sentences.

In order to get the right parameters for our ensemble classifier, we set the Sequential-Model Based Bayesian Optimisation (SMBO) meta-learning initialisation (outlined by Feurer et al. (2015)) to 0. This allowed us to fine-tune our hyper-parameters based on optimisations already performed by Fersini et al. (2015) on data sets similar to ours. We limit the time to optimise the hyper-parameters to 7200 seconds (2 hours). This is to ensure that we have sufficient time to find a relatively good model whilst acknowledging the time limit that we have with our computing resources. We share the details on our GitHub page,[7] which lists the best performing hyper-parameters for each of our classifiers.

[7] https://github.com/prasys/textanalyzer

For our deep learning models, we set the dropout to 0.2 to avoid overfitting. The hidden units in the bi-direction LSTM were set to 300 (150 in each of the two layers). As for our LSTM Dense Layer, we set it to 128. We used the Adam Optimizer with a learning rate of 10^{-5} for 30 epochs. We used a batch size of 32. We also used our validation F_1 score as an early stopping criterion. Training stopped if the score used as the stopping criterion does not increase for 15 consecutive epochs or the maximum number of epochs was reached. The model achieving the best result in the validation set was chosen for the final evaluation.

We follow the recommendation outlined by Crane (2018) in listing the version of software and hardware used to conduct our experiment to aid in reproducing our results. We used TensorFlow[8] 1.14, Keras[9] 2.2, sklearn[10] 0.22, huggingface 2.7.0, and for the hyper-parameter tuning we used auto-sklearn 0.69. We ran our experiments on an Intel Core i7 6700K @ 4.00GHz CPU with an NVIDIA GeForce GTX TITAN (CUDA Version 10.1) running on Red Hat 4.8.5-36.

To validate our TD-LSTM deep model implementation, we performed a comparison with the current state-of-the-art model (Patro et al., 2019). We report our scores in Table 4 where it can be seen that our system is performing comparably. We performed a one-way ANOVA, which showed no statistically significant difference at the $p < 0.05$ level thus providing confidence in our implementation.

Data set	Patro et al. (2019) (TD-LSTM)	Our Impl. (TD-LSTM)
Books	0.8277	0.8315
Tweets	0.8771	0.8579

Table 4: Dice Score yielded by our implementations and on the current the state-of-the-art on the data sets used in their experiments to published results.

[8] https://www.tensorflow.org/
[9] https://keras.io/
[10] https://scikit-learn.org/

5. Results

To examine the effectiveness of our system, we performed two types of experiments. First, we evaluated the performance of the target detection (i.e., the classifier), the results of which are presented in Section 5.1. Then we measure the performance of extracting the target of sarcasm (i.e., the deep learning), presented in Section 5.2. To determine the efficiency of our system we measured training time and run-time, the results of which are presented in Section 5.3. Section 5.4 includes a failure analysis.

5.1. Performance of Target Detection

To show the effectiveness of using a classifier to distinguish between OUTSIDE and INSIDE, we evaluated the performance of different embedding techniques with each classifier. Then we looked at improving the performance by utilising two different class imbalance techniques. Finally, we measured the performance of the classifier against deep learning models used in distinguishing OUTSIDE and INSIDE.

5.1.1. Performance of Individual Classifiers

We ran Random Forest (RF), Support Vector Machine (SVM) and Logistic Regression (LF) individually on the various sentence embedding models to identify if there is a presence of sarcasm in the text (INSIDE). We evaluated their balanced accuracy score on our validation set for all three data sets. Table 5 shows the performance of each of the classifiers and confirms that each of the classifiers individually does perform well on the validation data.

5.1.2. Performance of Class Imbalances Techniques

We found a large class imbalance between the two classes of OUTSIDE and INSIDE. This could have a detrimental effect on classifier performance if ignored (Gosain & Sardana, 2017).

To determine the best class balancing technique for this task, we measured the number of times that balancing the weights of each of the classes outperformed SMOTE in balanced accuracy. We used sklearn for the implementation

Embedding	*Tweets*			*Books*			*Reddit*		
	SVM	LR	RF	SVM	LR	RF	SVM	LR	RF
albert_base	0.4125 ± 0.0792	0.4185 ± 0.0751	0.4326 ± 0.0443	0.3782 ± 0.0443	0.3691 ± 0.0471	0.3855 ± 0.0790	0.3834 ± 0.0365	0.3691 ± 0.0365	0.4055 ± 0.0365
albert_large	**0.4516 ± 0.0536**	0.4432 ± 0.0538	0.4644 ± 0.0574	0.3857 ± 0.0574	0.3754 ± 0.0565	0.3796 ± 0.0537	0.3123 ± 0.0361	0.3191 ± 0.0361	0.4151 ± 0.0361
albert_xlarge	0.3868 ± 0.0641	0.3780 ± 0.0637	0.4241 ± 0.0181	0.3729 ± 0.0181	0.3755 ± 0.0161	0.3721 ± 0.0635	0.3319 ± 0.0223	0.3412 ± 0.0223	0.3992 ± 0.0223
bert_base_cased	0.4092 ± 0.0659	0.4145 ± 0.0657	0.4525 ± 0.0493	0.3753 ± 0.0493	0.3653 ± 0.0523	0.3951 ± 0.0656	0.3882 ± 0.0408	0.3779 ± 0.0408	0.4223 ± 0.0408
bert_base_uncased	0.4457 ± 0.0251	**0.4687 ± 0.0253**	0.4595 ± 0.0245	0.4134 ± 0.0245	0.4145 ± 0.0221	0.4153 ± 0.0251	0.3666 ± 0.0536	0.3711 ± 0.0536	0.3921 ± 0.0536
bert_base_large_cased	0.3940 ± 0.0326	0.4049 ± 0.0347	0.4224 ± 0.0185	0.3695 ± 0.0185	0.3543 ± 0.0185	0.3591 ± 0.0327	0.3656 ± 0.0696	0.3614 ± 0.0696	0.3810 ± 0.0696
bert_base_large_uncased	0.4232 ± 0.0305	0.429 ± 0.0312	**0.4956 ± 0.0636**	**0.4221 ± 0.0636**	**0.4225 ± 0.0596**	**0.4321 ± 0.0300**	0.4259 ± 0.1000	0.4151 ± 0.1000	0.4357 ± 0.1000
use_elm	0.4064 ± 0.0418	0.4041 ± 0.0406	0.4337 ± 0.0197	0.3991 ± 0.0197	0.3859 ± 0.0153	0.3993 ± 0.0419	0.4327 ± 0.0990	0.4353 ± 0.0990	0.4491 ± 0.0990
use_transformer	0.4335 ± 0.0285	0.4311 ± 0.0266	0.4618 ± 0.0204	0.3582 ± 0.0204	0.3661 ± 0.0222	0.3643 ± 0.0286	0.4333 ± 0.0811	0.4412 ± 0.0811	0.4415 ± 0.0811
fasttext	0.4157 ± 0.0323	0.4142 ± 0.0354	0.4433 ± 0.0191	0.3598 ± 0.0191	0.3604 ± 0.0205	0.3674 ± 0.0323	0.4321 ± 0.0193	0.4285 ± 0.0193	0.4082 ± 0.0193
twitter_200d	0.4149 ± 0.0339	0.4099 ± 0.0370	0.4391 ± 0.0363	0.3450 ± 0.0363	0.3411 ± 0.0339	0.3488 ± 0.0340	0.4899 ± 0.1017	**0.4714 ± 0.1017**	0.4431 ± 0.1017
glove_commoncrawl	0.4164 ± 0.0504	0.4153 ± 0.0515	0.4681 ± 0.0413	0.3781 ± 0.0413	0.3541 ± 0.0437	0.3815 ± 0.0505	0.3426 ± 0.0397	0.3331 ± 0.0397	0.3813 ± 0.0397
elmo	0.4333 ± 0.0429	0.4295 ± 0.0429	0.4240 ± 0.0464	0.3582 ± 0.0464	0.3586 ± 0.0462	0.3819 ± 0.0429	0.4322 ± 0.0270	0.4251 ± 0.0270	0.4513 ± 0.0270
gpt	0.3986 ± 0.0482	0.4098 ± 0.0482	0.4621 ± 0.0196	0.3864 ± 0.0196	0.3860 ± 0.0196	0.3998 ± 0.0483	0.4481 ± 0.0294	0.4414 ± 0.0294	0.4388 ± 0.0294
gpt2	0.4251 ± 0.0524	0.4316 ± 0.0555	0.4925 ± 0.0106	0.3833 ± 0.0106	0.3835 ± 0.0106	0.3912 ± 0.0525	0.4664 ± 0.0499	0.4711 ± 0.0499	0.4841 ± 0.0499
XLTransformer	0.4211 ± 0.0323	0.4223 ± 0.0324	0.4543 ± 0.0095	0.3572 ± 0.0195	0.3572 ± 0.0167	0.3515 ± 0.0324	**0.4719 ± 0.0432**	0.4688 ± 0.0200	**0.4915 ± 0.0300**

Table 5: Mean balanced accuracy scores (with standard deviation) of individual classifiers with different embedding techniques. Best performing combinations are highlighted in bold

Data set	P(sklearn > SMOTE)	SMOTE	sklearn	No Class Balancing
Tweets	0.497	0.5221 ± 0.0143	**0.5154 ± 0.0197**	0.4925 ± 0.0719
Books	0.502	0.4517 ± 0.0483	**0.4581 ± 0.0204**	0.4145 ± 0.0657
Reddit	0.513	0.5096 ± 0.1221	**0.5122 ± 0.0184**	0.4915 ± 0.0300

Table 6: Mean balanced accuracy scores (with standard deviation) of using SMOTE and sklearn. Best performing combinations are highlighted in bold.

of class weights and imbalanced-learn for the implementation of SMOTE[11]. We conducted this experiment using 1000 different seed values across our three data sets by using the best performing embedding technique and individual classifier which was a combination of RF and *bert_base_large_uncased* for *Tweets* and *Books*. For *Reddit*, the best performing combination was *XLTransformer* and RF. We report our results in Table 6. In the table, we present the probability that sklearn outperformed SMOTE P(sklearn > SMOTE) and as well as the mean Balanced Accuracy Score obtained for SMOTE, sklearn, and without using any class balancing techniques.

From our results, we can see that balanced sklearn weighting and SMOTE helped to improve our performance. Balanced sklearn outperformed SMOTE across all the three data sets. In order to determine if there are any statistically significant differences between them, we performed a modified paired t-test (Nadeau & Bengio, 2003) on the results across the three data sets. In the original paired t-test, the assumption of independence is violated as there is an overlap between training and the validation data set. Although class weighting seemed to perform better than SMOTE, our t-tests showed that the results are not statistically significant at the $p < 0.05$ level. Additionally, we performed a two-way ANOVA across our results which gave a p-value of 0.0952 (not significant at $p < 0.05$). Since there is no clear advantage of using a sklearn over SMOTE, we decided to use SMOTE to handle our class imbalance. Past studies have found SMOTE to be better at handling class imbalance (Douzas et al.,

[11] https://pypi.org/project/imbalanced-learn/

2018; Fernández et al., 2018; Maipradit et al., 2019; Guo et al., 2019).

5.1.3. Performance of Ensemble Classifiers

To examine whether or not we can further improve the performance of our first phase classifier, we used an ensemble of classifiers to combine the three classifiers we used earlier. We experimented with two approaches: majority voting (hard voting) and soft voting.

In the majority voting approach, each of the individual classifiers votes for a class and the majority class is chosen. For the soft voting approach, every individual classifier provides a probability score to a particular target class. These probabilities are then summed and the class with the largest score is chosen. Table 7 shows a summary of the performance of the ensemble classifier across the three data sets.

The best performance is obtained by using an ensemble classifier with a hard voting approach. This is because there is a high degree of similarity in SVM and LF classifiers, so the final ensemble decision relied strongly on Random Forest (RF) in soft voting, and thus the results did not as fare as well as a hard voting approach.

5.1.4. Performance of Best Performing Classifiers versus Baselines

To evaluate our classifier's performance, we compare the performance of our hard voting ensemble classifier with each of the embedding methods we used, and against our baselines (NB, TC-LSTM, TD-LSTM and RAM) on our *evaluation* portion of our three data sets. Table 8 shows the performance of the classifiers in identifying the presence of sarcasm targets in the text.

The best performing classifier for *Tweets* is using *bert_base_uncased* embedding which achieved a mean balanced accuracy of 0.5492 and a mean F_1 of 0.2165. As for *Reddit*, the best performing classifier is *albert_base* which to obtained a mean balanced accuracy of 0.6027 and a mean F_1 of 0.4619. Lastly for *Books*, the best performing classifier is *gpt2* with a mean balanced accuracy of 0.5080 and a mean F_1 of 0.1855.

Embedding	*Tweets*		*Books*		*Reddit*	
	Soft Voting	Hard Voting	Soft Voting	Hard Voting	Soft Voting	Hard Voting
albert_base	0.4666 ± 0.0427	0.4716 ± 0.0795	**0.5328 ± 0.0748**	**0.5414 ± 0.0570**	**0.5576 ± 0.0601**	**0.5691 ± 0.0584**
albert_large	0.4859 ± 0.0772	0.5238 ± 0.0430	0.5145 ± 0.0849	0.5224 ± 0.0508	0.5413 ± 0.0627	0.5555 ± 0.0492
albert_xlarge	0.4916 ± 0.0697	0.5111 ± 0.0749	0.5049 ± 0.0415	0.5142 ± 0.0536	0.5094 ± 0.0450	0.5104 ± 0.0652
bert_base_cased	0.4810 ± 0.0578	0.4869 ± 0.0801	0.4913 ± 0.0506	0.5195 ± 0.0671	0.5413 ± 0.0793	0.5495 ± 0.0790
bert_base_uncased	0.4675 ± 0.0540	0.4991 ± 0.0723	0.4895 ± 0.0622	0.4923 ± 0.0894	0.5049 ± 0.0759	0.5131 ± 0.0643
bert_base_large_cased	0.4412 ± 0.0567	0.4815 ± 0.0535	0.5111 ± 0.0680	0.5102 ± 0.0878	0.5243 ± 0.0634	0.5333 ± 0.0888
bert_base_large_uncased	0.4656 ± 0.0868	0.5115 ± 0.0872	0.5221 ± 0.0517	0.5204 ± 0.0646	0.5396 ± 0.0829	0.5413 ± 0.0635
use_elm	0.4537 ± 0.0411	0.4825 ± 0.0406	0.5009 ± 0.0505	0.5041 ± 0.0629	0.5094 ± 0.0654	0.5219 ± 0.0828
use_transformer	0.4718 ± 0.0067	0.4910 ± 0.0406	0.4918 ± 0.0439	0.5013 ± 0.0793	0.5103 ± 0.0739	0.5221 ± 0.0569
fasttext	0.4833 ± 0.0748	0.5012 ± 0.0430	0.4681 ± 0.0842	0.4958 ± 0.0597	0.5335 ± 0.0516	0.5443 ± 0.0801
twitter_200d	0.4591 ± 0.0743	0.4996 ± 0.0525	0.5004 ± 0.0553	0.5009 ± 0.0540	0.5139 ± 0.0559	0.5121 ± 0.0525
glove_commoncrawl	0.4681 ± 0.0618	0.5220 ± 0.0886	0.4918 ± 0.0808	0.5144 ± 0.0865	0.5231 ± 0.0761	0.5315 ± 0.0833
elmo	0.4240 ± 0.0606	0.5014 ± 0.0894	0.5104 ± 0.0505	0.5354 ± 0.0886	0.5402 ± 0.0683	0.5498 ± 0.0493
gpt	0.4621 ± 0.0540	0.5328 ± 0.0782	0.5006 ± 0.0808	0.5137 ± 0.0577	0.5219 ± 0.0567	0.5314 ± 0.0585
gpt2	**0.4925 ± 0.0719**	**0.5342 ± 0.0795**	0.4684 ± 0.0855	0.4841 ± 0.0797	0.5523 ± 0.0468	0.5672 ± 0.0611
XLTransformer	0.4543 ± 0.0448	0.5254 ± 0.0641	0.4985 ± 0.0479	0.4974 ± 0.0523	0.5413 ± 0.0491	0.5485 ± 0.0682

Table 7: Mean balanced accuracy score with standard deviation of soft and hard voting techniques used in classifiers. Best performing combinations are highlighted in bold

	Tweets		*Books*		*Reddit*	
Method	Balanced Accuracy	F_1 Score	Balanced Accuracy	F_1 Score	Balanced Accuracy	F_1 Score
albert_base	0.5292 ± 0.0790	0.1460 ± 0.1175	0.4902 ± 0.0359	0.1007 ± 0.1397	**0.6027 ± 0.0201**	**0.4619 ± 0.0263**
albert_large	0.5367 ± 0.0537	0.1826 ± 0.1027	0.4857 ± 0.0201	0.0801 ± 0.1573	0.5602 ± 0.0412	0.3684 ± 0.0779
albert_xlarge	0.4991 ± 0.0635	0.0482 ± 0.0542	0.4799 ± 0.0175	0.0125 ± 0.0395	0.5379 ± 0.0288	0.3014 ± 0.0770
bert_base_cased	0.5169 ± 0.0656	0.1273 ± 0.1288	0.4826 ± 0.0317	0.0284 ± 0.0647	0.5802 ± 0.0353	0.4320 ± 0.0968
bert_base_uncased	**0.5492 ± 0.0251**	**0.2165 ± 0.0581**	0.4734 ± 0.0154	0.0000 ± 0.0000	0.5714 ± 0.0283	0.4172 ± 0.0671
bert_base_large_cased	0.4909 ± 0.0327	0.0789 ± 0.0756	0.5066 ± 0.0521	0.0794 ± 0.1203	0.5872 ± 0.0545	0.4206 ± 0.0729
bert_base_large_uncased	0.5131 ± 0.0300	0.1240 ± 0.0441	0.4850 ± 0.0213	0.0000 ± 0.0000	0.5680 ± 0.0385	0.3805 ± 0.0740
use_elm	0.5096 ± 0.0419	0.1368 ± 0.1000	0.4781 ± 0.0203	0.0100 ± 0.0316	0.5664 ± 0.0301	0.4092 ± 0.0811
use_transformer	0.5353 ± 0.0286	0.1767 ± 0.0621	0.4879 ± 0.0243	0.0160 ± 0.0506	0.5723 ± 0.0289	0.4298 ± 0.0544
fasttext-gnews	0.5153 ± 0.0323	0.1432 ± 0.0629	0.4832 ± 0.0297	0.0375 ± 0.0509	0.5683 ± 0.0483	0.3946 ± 0.1009
twitter_200d	0.5073 ± 0.0340	0.1288 ± 0.0814	0.4911 ± 0.0293	0.0478 ± 0.0799	0.5374 ± 0.0328	0.3618 ± 0.0765
crawl_42b	0.5121 ± 0.0505	0.1273 ± 0.0785	0.4977 ± 0.0381	0.0758 ± 0.1230	0.5603 ± 0.0279	0.4041 ± 0.0599
elmo	0.5335 ± 0.0429	0.1456 ± 0.0756	0.4890 ± 0.0205	0.0384 ± 0.0546	0.5730 ± 0.0363	0.4117 ± 0.0744
gpt	0.4972 ± 0.0483	0.0642 ± 0.0827	0.5061 ± 0.0785	0.0567 ± 0.1315	0.5787 ± 0.0259	0.4015 ± 0.0638
gpt2	0.5228 ± 0.0525	0.1376 ± 0.1007	**0.5080 ± 0.0696**	**0.1855 ± 0.2277**	0.5641 ± 0.0310	0.3557 ± 0.0638
XLTransformer	0.4760 ± 0.0324	0.0422 ± 0.0642	0.5024 ± 0.0479	0.0788 ± 0.1181	0.5773 ± 0.0372	0.3899 ± 0.0892
NB (Baseline)	0.4513 ± 0.0086	0.0000 ± 0.0000	0.4728 ± 0.0085	0.0000 ± 0.0000	0.5118 ± 0.0516	0.0555 ± 0.0280
TD-LSTM (Baseline)	0.4880 ± 0.0317	0.0881 ± 0.0642	0.4808 ± 0.0157	0.0205 ± 0.0445	0.4895 ± 0.0322	0.3192 ± 0.0544
RAM	0.4967 ± 0.0229	0.1193 ± 0.0488	0.4815 ± 0.0247	0.0329 ± 0.0538	0.4933 ± 0.0342	0.2939 ± 0.0668
TC-LSTM	0.4890 ± 0.0171	0.0802 ± 0.0496	0.4910 ± 0.0302	0.0535 ± 0.0772	0.5107 ± 0.0484	0.3173 ± 0.0788

Table 8: Mean balanced accuracy score and F_1 score (with standard deviation) of the best performing classifier with various embeddings. Best performing combinations are highlighted in bold

For the *Books* data set *bert_base_uncased* and *bert_base_large_uncased* embedding techniques achieved a F_1 score of 0. This we attribute to the very low number of OUTSIDE cases in the data set, thus making it difficult for the classifier to correctly identify OUTSIDE cases. Apart from low number of OUTSIDE cases, case sensitivity had an impact on the overall performance of our classifiers. Embedding techniques such as *bert_base_cased* which is case sensitive performed poorly on the *Tweets* data set. However, it did result in an increase of about 4% for both *Reddit* and *Books*. From visual inspection, the poor performance for *Tweets* can be attributed to the observation that the majority of tweets are typed all in lower case. This is in line with Rao et al. (2010)'s findings. The reason why different embeddings perform differently on each of the data sets is due to the nature of the data set itself. For instance, *gpt2*'s good performance on the *Books* data set can be attributed to the vast amount of books data it was trained on.

For *Tweets* and *Books*, we also obtained a F_1 score of 0 when NB is used. This is mainly due to the way the probabilities are calculated. As some of the words are not seen in the training data, the NB classifier cannot predict OUTSIDE accurately. Additionally we have observe that our other baselines (such as TC-LSTM, TD-LSTM, and RAM) do not yield substantially higher results compared to our hard-voting classifier.

As for the *Reddit* data, TC-LSTM achieved a balanced accuracy of 0.5107. TD-LSTM and RAM only obtained a balanced accuracy of 0.4895 and 0.4933 respectively. The balanced accuracy scores obtained by TC-LSTM, TD-LSTM and RAM are lower than NB, which is 0.5118 whereas our best performing classifier with *albert_base* obtained the higher score of 0.6027.

One of the reasons why TC-LSTM, TD-LSTM and RAM did not perform well in distinguishing OUTSIDE from INSIDE is that these models tend to predict pronouns and pronominal adjectives in the cases of OUTSIDE, which reduces the overall balanced accuracy score and F_1 score. Another observation we have on the poor performance of these models on the *Reddit* data set is the approximately 30% out-of-vocabulary (OOV) words in the *Reddit* data set compared

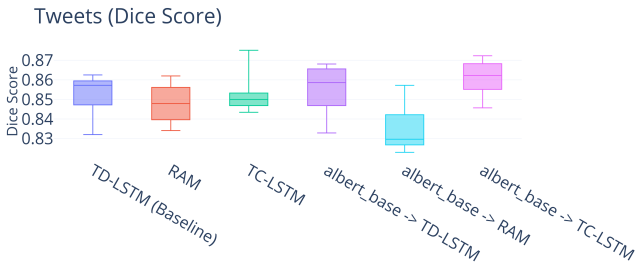
to 5% in *Tweets* and 2% in *Books*. OOV terms are likely to negatively impact the performance of the model.

5.2. Performance of Target Extraction

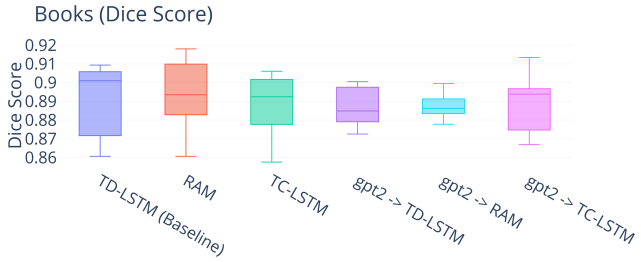
Our next phase of evaluation was to measure the performance in extracting the target of sarcasm. For this, we selected the best embedding combination from each of the data sets which are: *bert_base_uncased* (*Tweets*), *gpt2* (*Books*) and *albert_base* (*Reddit*); and compared to TC-LSTM, TD-LSTM and RAM. We additionally tested TC-LSTM, TD-LSTM and RAM without a classifier as we wanted to measure the effectiveness of our classifier as a pre-process for identifying the target of sarcasm.

Figure 4 presents the performance of our novel sarcasm detection system using a combination of our best performing classifiers and TC-LSTM, TD-LSTM and RAM against solely using TC-LSTM, TD-LSTM and RAM. Overall, across all three data sets, we observe the combination of using TC-LSTM with our classifier yields the best mean score (0.860 for *Tweets*, 0.715 for *Reddit* and 0.890 for *Books*). By comparison, the current state-of-the-art model (TD-LSTM) achieves scores of 0.851 for *Tweets*, 0.587 for *Reddit*, and 0.891 for *Books*. Comparing the two results, t-tests show no statistical significance at $p < 0.05$ for *Books* and *Tweets*.

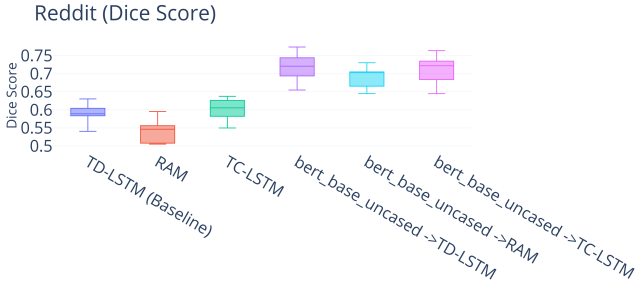
For *Reddit*, we performed a one-way ANOVA and obtained a p value of 0.0354, a statistically significant difference at the $p < 0.05$ level. Then we used paired t-tests to identify any statistically significantly different pairs ($p < 0.05$ level). From our paired t-test, we found statistically significant differences between two groups: TC-LSTM and combination of *albert_base* as well as TD-LSTM and combination of *albert_base*. We observe that when we identify OUTSIDE accurately, it yields a higher score compared to obtaining the target accurately, which is a lot more difficult (Parameswaran et al., 2019). Consider the following the example from the *Reddit* training data set. The target of sarcasm given by the annotators are highlighted in bold.



(a) Tweets (Dice score)



(b) Books (Dice score)



(c) Reddit (Dice score)

Figure 4: Performance of classifier in identifying the presence of a target of sarcasm

Predicted Words	Dice Score
cinematic	0.5000
splicing artifacts for the full	0.2857
Entire Sentence	0.1935

Table 9: Impact of the length of subsequence of predicted words on Dice scores

"*if the console is only rendering 30 fps then the game will look times better than it would at 60 fps because it has more time to render it better..*" (**Outside**)

"*you have to add the noise from a old style reel to reel movie projector as well as simulating film and splicing artifacts for the* ***full cinematic experience****.*"

In the first example the target of sarcasm is OUTSIDE. We get a perfect Dice score of 1 if predicted properly. In the second example, it is very hard to get a perfect Dice score. In Table 9, we show three examples of possible targets and how the score varies depending on the number of words predicted correctly, and length of the predicted words. To achieve a perfect score the system must produce the perfect answer *after* first identifying that the target is INSIDE.

We are also interested in the reasons for the lower scores on the other two data sets. Investigation into why this is the case suggests that for these data sets our classifier correctly identifies the target as INSIDE, but the deep learning disagrees and labelled them as OUTSIDE. These sentences tend not to contain pronoun and pronominal adjectives. We analyse this further as part of our failure analysis in Section 5.4.

5.3. Efficiency

In this section we measure the efficiency of our systems by measuring both training time and run-time of our target detector and target extractor.

We report the training time to demonstrate that the transfer learning aspect of our work does not require an unreasonably large amount of hardware or

period of time. We report the run-time to demonstrate that the overhead of our approach, over others', is not large.

For all of our experiments, we use Python's time library (using *time.time()* method). Clearly, our implementation is in Python and this, itself, introduces a run-time overhead. We have not spent time optimising our implementation of any of the systems we report, so our results should be considered indicative of the expected relative performance. Better results can be expected from a hand-optimised implementation written in a compiled language, but we leave this for future work.

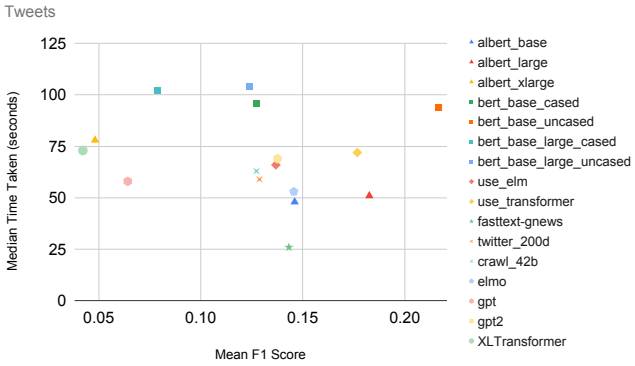
We report the median time across five runs, but mean F_1 and mean Dice scores for quality.

5.3.1. Target Detection Training Time

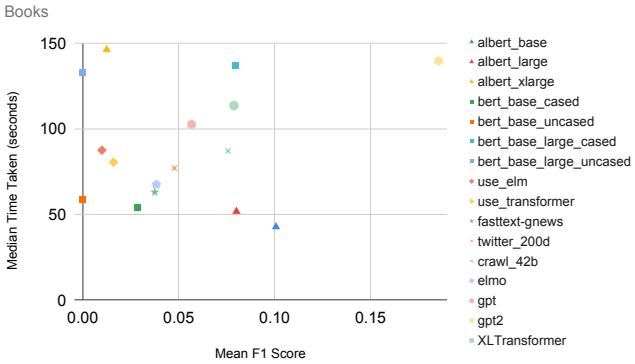
We examine and compare cost by measuring the time it took to train our best classifier, and plotting that against the corresponding mean F_1 score from Table 8. We measured the time taken from the transformation of texts into an embedding until the completion time of sklearn's *fit* function (training function). That is, we do not measure the time to compute the embedding from the words in the text—because that is a one-off process that would ordinarily happen before any training happens. We present our findings in Figure 5. From the figure, it can be seen that training time can reasonable be described as "a few minutes" and that it is not generally true that a longer training time results in better quality of results. Some embeddings are better than others regardless of training we perform on our classifer. Our experimental results suggests to us that some embeddings are better suited for the tasks that we are performing compared to others.

5.3.2. Target Extraction Training Time

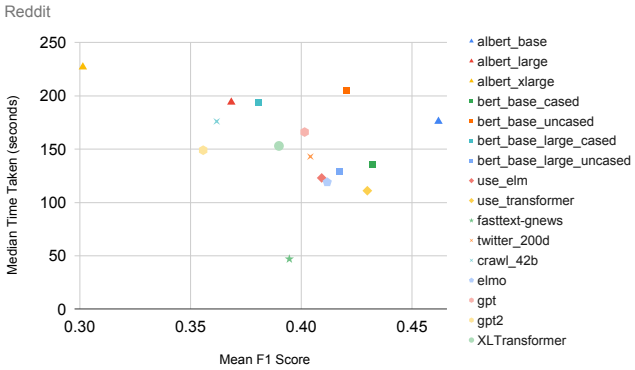
We measured the time taken by the deep learning classifier from the time to transform the text into the embedding until the completion of TensorFlow's *train* function. We compared the median time taken with the mean Dice score



(a) Tweets



(b) Books



(c) Reddit

Figure 5: Median time taken to train the best performing classifier with various embeddings and the corresponding mean F_1 scores obtained in evaluation set

we obtained from Figure 4. We report our findings in Figure 6.

From our experimental results on *Books* and *Tweets*, our method of using a combination of a classifier and a deep learning model takes 5% longer without gaining any statistically significant advantage (or disadvantage) measured with the Dice Score. However, for *Reddit* our approach is justifiable. For example, after spending 5.03 hours training TD-LSTM, an additional 4.33% training time to add the classifier resulted in an 18% improvement in the quality of the results.

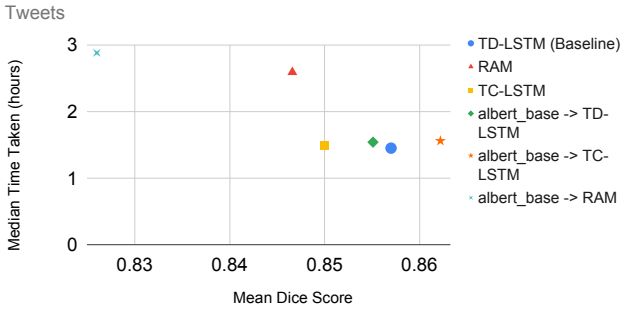
RAM was the slowest to train among the deep neural network models, and we investigated further. We believe that the primary reason for this is the high memory requirements (Zhang et al., 2019a) which resulted in operating system requiring to swap to disk. As part of future work, this might be optimised by fine-tuning the implementation of the model or making the model more memory efficient.

5.3.3. Target Detection Run-Time

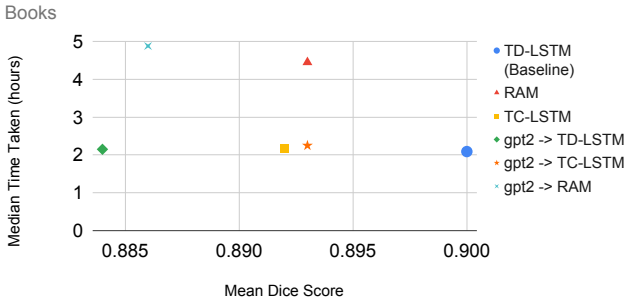
We examine the median time taken for our best performing classifiers to completely classify our evaluation data and plot this against the F_1 score from Table 8.

We measured the run-time as the time taken including transforming the text into embeddings and the completion of sklearn's *predict* function (classification). We include the cost of transforming into embeddings as that happens only once for each text and is, therefore, part of the total cost of finding the target of the sarcasm. We report our findings in Figure 7.

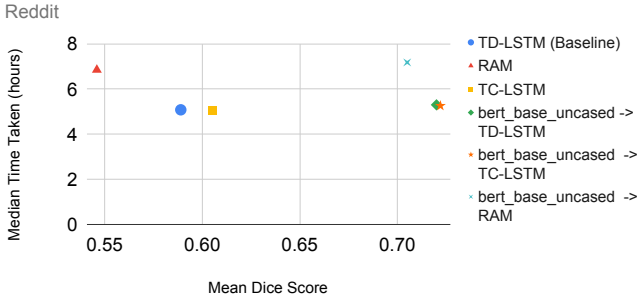
Overall, for *Tweets*, the best performing embedding (*bert_base_uncased*) takes 6.8 seconds to classify all 67 tweets. *gpt2* (the best performing embedding for *Books*) taking 18.0 seconds to classify 152 book snippets. For *Reddit*, our best performing embedding, *albert_base* takes 61.6 seconds to classify 285 posts. The difference is timing is due to the embedding dimensions and also how the embedding model was constructed.



(a) Tweets

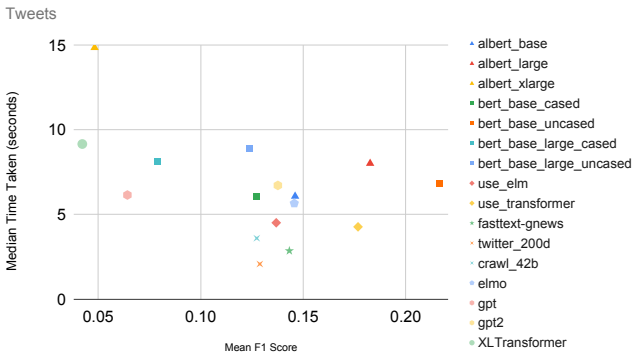


(b) Books

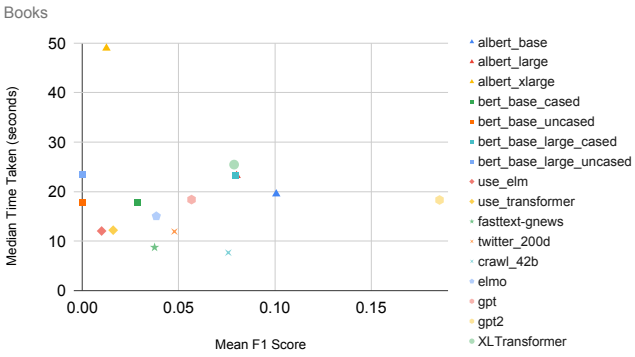


(c) Reddit

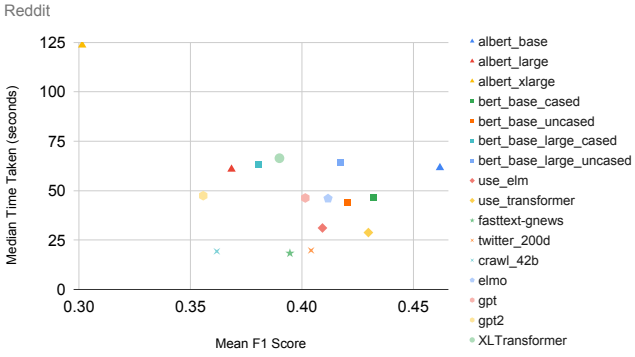
Figure 6: Median time taken to train deep learning classifiers and the corresponding mean Dice scores obtained in evaluation set



(a) Tweets



(b) Books



(c) Reddit

Figure 7: Median time taken to classify texts in evaluation data set with various embeddings and the corresponding mean F_1 scores

5.3.4. Target Extraction Run-Time

Next, we examine the total time taken for each system to identify the target. We plot the median time taken to identify the target against the mean Dice Score from Figure 4. We present our findings in Figure 8. We measured the run-time as the time taken including the transformation of the text into an embedding until the completion of the TensorFlow *run* function.

We observe from our experiments that there is a measurable overhead to using a classifier.

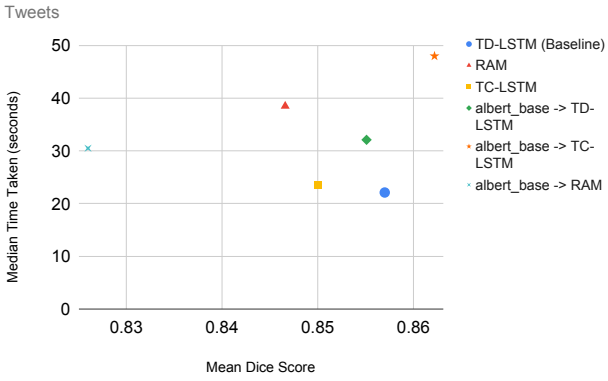
Our method of using a classifier and deep learning model works well when it comes to *Reddit*. Although the combination of the *bert_base_uncased* classifier and TD-LSTM takes 34% longer than TD-LSTM, it gives an 18% improvement in the Dice Score. We believe that the efficiency of our method can be improved by caching embedding (that are currently computed twice for reason of simplicity in implementation), but we leave this to future work.

5.4. Failure Analysis

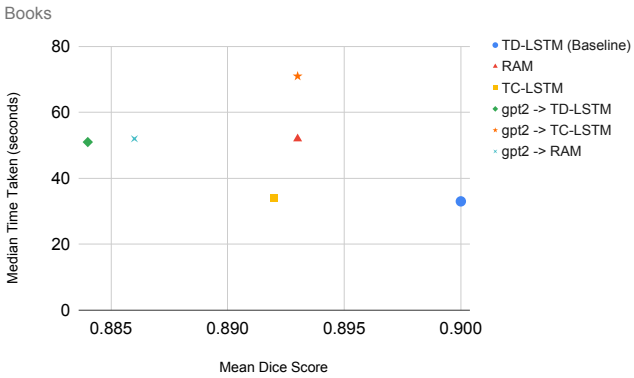
In Section 5.2 we observed cases of the deep-learning models re-classifying sentences as OUTSIDE despite the classifier having already classified them as INSIDE. We investigated this further with respect to the *Reddit* data as there were more cases of OUTSIDE compared to the other two data sets. On *Reddit*, novel words are coined often on various different Subreddits (Cole et al., 2017) and these words are not in any of the pretrained language models, thus making it challenging to learn their semantics. As an example from our training data, *"gamecribs"* is League of Legends (a computer game) terminology. However, we could not locate this word in any of the pretrained models. Such words are marked OOV. We believe that the presence of OOV terms reduces the overall performance of our system.

There are several ways OOV terms might be managed. One is to use a crowd-sourced dictionary such as Urban Dictionary.[12] This would allow neol-

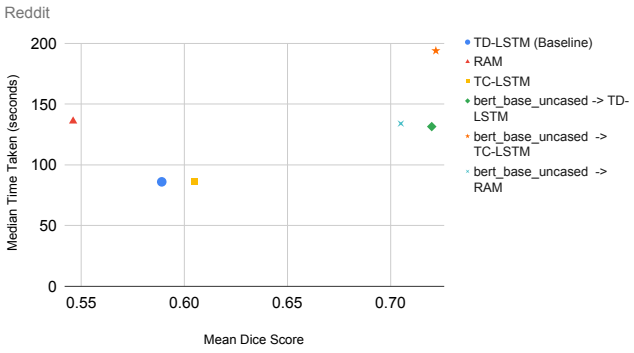
[12] https://www.urbandictionary.com/



(a) Tweets



(b) Books



(c) Reddit

Figure 8: Median time taken to extract the target in evaluation data set with the corresponding mean Dice scores

Example Text	Predicted Words	Ground Truth	Data set
"America: It's a free country; that is if you can afford it"	"America", "It"	"America"	*Books*
"I don't see why OP needed to point out that the seal was black in his title"	"OP", "his"	"OP"	*Reddit*
"#sleep lots last night...good thing is my computer is down and hurray for Monopoly"	"computer", "Monopoly"	"my computer"	*Tweets*

Table 10: Inconsistency in Ground Truth

ogisms (that are commonly not found in traditional lexicons) to be captured and replaced with more common synonyms (Nguyen et al., 2018). Another approach might be to try to understand the context in the sentence. Several approaches might be taken, for example: using subword embeddings (Bojanowski et al., 2017) or learning different representations for words that occur less frequently (Sergienya & Schütze, 2015). However, recent work suggests that deriving context from word embedding is challenging (Li et al., 2017). We leave the exploration of OOV terms to future work.

Our model can identify additional words which refer to the target but which are not present in the assessments. We show an example from each of our data sets in Table 10. From Table 10, we can see that for the *Books* example that annotators selected "*America*" as the target of the sarcasm. However our model identified "*America*" and "*It*". This is because they both refer to the same entity. Unfortunately, there are many similar instances in the data where the annotators do include the pronoun of entity. This pattern can be seen across all data sets and it is not clear to us what the rule for inclusion might be.

Our experiments also confirm the work of others which finds a low agreement level between annotators for sarcasm tasks and as well inconsistency among the annotators (González-Ibáñez et al., 2011; Parameswaran et al., 2019). In order to address this gap, we support the suggestion of Amidei et al. (2018), that more attention be given to internal consistency of the responses from each annotator instead of focusing on inter-agreement between annotators. One way

of addressing this would be through the use of co-reference resolution (Peng et al., 2015), whereby all the potential references of the targets can be included if one of the targets is marked.

6. Conclusion and Future Work

In this paper we investigated the identification of the target of a sarcastic text. The task is challenging as there may be a single target, multiple targets, or the target may not be in the text being processed. We present a system that uses a machine learning classifier to detect the presence or absence of a sarcastic target, then uses a deep learning model to accurately determined the target or targets from within the text.

The effectiveness of this approach is demonstrated through extensive experimentation on three public data sets. We show that our approach performs similarly to the current state-of-the-art of Patro et al. (2019) on two of the three data sets. For the other, *Reddit*, our approach achieved a statistically significant 18% improvement.

Our first research question was: *Can we accurately identify the target of sarcasm in a sarcastic text?* We find that this is difficult, but in many cases we can. Our approach performs no worse than previous approaches, and better in one of the three data sets.

Our second research question was: *What factors affect the performance of a system that detects the target of sarcasm?* Our failure analysis suggests that an effective method for dealing with out of vocabulary terms will result in improvements. Our anaysis also suggests that it will be difficult to make substantial improvements because of inconsistencies in the currently openly available data sets. That is, a new data set that is consistent (particularly with multiple targets, pronouns, and so on) is needed, and that data set should (at least initially) stick to commonly used words.

We believe that our work builds a strong foundation for future approaches to sarcasm target identification, and we offer suggestions on how to build on

it. Firstly, one could explore the use of external sources such as a dictionary to further improve the classifiers especially when it comes to jargon and rare (out of vocabulary) words. Secondly, we suggest asking the authors of sarcastic texts to identify the targets (rather than a third-party annotator). Clearly, alternative embeddings could be used, or combinations of embeddings could be used. Other state-of-the-art ABSA models such as auxiliary memory (Xue & Li, 2018) in a deep learning model might help. We also believe that using the linguistic features, such as those used by Patro et al. (2019) could strengthen the approach. Importantly, in an effort to improve the quality and consistency of the training data, we suggest a set of conventions for constructing a new data set.

7. Acknowledgements

We are grateful to the anonymous reviewers whose detailed feedback helped improve and clarify this manuscript. Additionally, we would like to thank all the unsung heroes during COVID-19 in keeping us safe.

References

Agrawal, A., & An, A. (2018). Affective representations for sarcasm detection. In *41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2018* (pp. 1029–1032). doi:10.1145/3209978.3210148.

Amidei, J., Piwek, P., & Willis, A. (2018). Rethinking the Agreement in Human Evaluation Tasks. In *Proceedings of the 27th International Conference on Computational Linguistics* (pp. 3318–3329). URL: https://www.aclweb.org/anthology/C18-1281.

Amir, S., Wallace, B. C., Lyu, H., Carvalho, P., & Silva, M. J. (2016). Modelling context with user embeddings for sarcasm detection in social media. In

CoNLL 2016 - 20th SIGNLL Conference on Computational Natural Language Learning, Proceedings (pp. 167–177). doi:10.18653/v1/k16-1017.

Arora, S., Liang, Y., & Ma, T. (2017). A simple but tough-to-beat baseline for sentence embeddings. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*.

Bamman, D., & Smith, N. A. (2015). Contextualized sarcasm detection on twitter. In *Proceedings of the 9th International Conference on Web and Social Media, ICWSM 2015* (pp. 574–577).

Bharti, S. K., Babu, K. S., & Jena, S. K. (2015). Parsing-based sarcasm sentiment recognition in Twitter data. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2015* (pp. 1373–1380). doi:10.1145/2808797.2808910.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146.

Buda, M., Maki, A., & Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, *106*, 249–259. doi:10.1016/j.neunet.2018.07.011.

Buschmeier, K., Cimiano, P., & Klinger, R. (2015). An Impact Analysis of Features in a Classification Approach to Irony Detection in Product Reviews. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis.* (pp. 42–49). doi:10.3115/v1/w14-2608.

Cer, D., Yang, Y., Kong, S. y., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Céspedes, M., Yuan, S., Tar, C., Sung, Y. H., Strope, B., & Kurzweil, R. (2018). Universal sentence encoder for English. In *EMNLP 2018 - Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Proceedings* (pp. 169–174). doi:10.18653/v1/d18-2029.

Chandrasekharan, E., Samory, M., Srinivasan, A., & Gilbert, E. (2017). The bag of communities: Identifying abusive behavior online with preexisting internet data. In *Conference on Human Factors in Computing Systems - Proceedings* (pp. 3175–3187). volume 2017-May. doi:10.1145/3025453.3026018.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. doi:10.1613/jair.953.

Cheang, H. S., & Pell, M. D. (2008). The sound of sarcasm. *Speech Communication*, *50*, 366–381. doi:10.1016/j.specom.2007.11.003.

Chen, P., Sun, Z., Bing, L., & Yang, W. (2017). Recurrent attention network on memory for aspect sentiment analysis. In *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings* (pp. 452–461). doi:10.18653/v1/d17-1047.

Chung, C. K., & Pennebaker, J. W. (2013). Linguistic Inquiry and Word Count (LIWC). In *Applied Natural Language Processing*. doi:10.4018/978-1-60960-741-8.ch012.

Cole, J. R., Ghafurian, M., & Reitter, D. (2017). Is word adoption a grassroots process? An analysis of Reddit communities. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (pp. 236–241). doi:10.1007/978-3-319-60240-0.

Crane, M. (2018). Questionable Answers in Question Answering Research: Reproducibility and Variability of Published Results. *Transactions of the Association for Computational Linguistics*, *6*, 241–252.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., & Salakhutdinov, R. (2019). Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association* (pp. 2978–2988). doi:10.18653/v1/p19-1285.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* (pp. 4171–4186). volume 1. URL: http://arxiv.org/abs/1810.04805.

Douzas, G., Bacao, F., & Last, F. (2018). Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE. *Information Sciences*, *465*, 1–20. doi:10.1016/j.ins.2018.06.056.

Eke, C. I., Norman, A. A., Liyana Shuib, & Nweke, H. F. (2020). Sarcasm identification in textual data: systematic review, research challenges and open directions. *Artificial Intelligence Review*, *53*, 4215–4258. URL: https://doi.org/10.1007/s10462-019-09791-8. doi:10.1007/s10462-019-09791-8.

Fernández, A., García, S., Herrera, F., & Chawla, N. V. (2018). SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. doi:10.1613/jair.1.11192.

Fersini, E., Pozzi, F. A., & Messina, E. (2015). Detecting irony and sarcasm in microblogs: The role of expressive signals and ensemble classifiers. In *Proceedings of the 2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015* (pp. 1–8). IEEE. doi:10.1109/DSAA.2015.7344888.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. *Advances in Neural Information Processing Systems*, *2015-Janua*, 2962–2970. doi:10.1007/978-3-030-05318-5.

Ghosh, A., & Veale, D. T. (2016). Fracking Sarcasm using Neural Network. In *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis* (pp. 161–169). doi:10.18653/v1/w16-0425.

Ghosh, D., & Muresan, S. (2018). "With 1 follower I must be AWESOME :P". Exploring the role of irony markers in irony recognition. *Icwsm*, (pp. 588–591). URL: http://arxiv.org/abs/1804.05253.

Gibbs Jr, R. W., Gibbs, R. W., & Gibbs, J. (1994). *The poetics of mind: Figurative thought, language, and understanding*. Cambridge University Press.

Giora, R. (2003). *On our mind: Salience, context, and figurative language*. Oxford University Press.

González-Ibáñez, R., Muresan, S., & Wacholder, N. (2011). Identifying sarcasm in Twitter: A closer look. *ACL-HLT 2011 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, *2*, 581–586.

Gosain, A., & Sardana, S. (2017). Handling class imbalance problem using oversampling techniques: A review. In *2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017* (pp. 79–85). doi:10.1109/ICACCI.2017.8125820.

Guo, S., Chen, R., Li, H., Zhang, T., & Liu, Y. (2019). Identify Severity Bug Report with Distribution Imbalance by CR-SMOTE and ELM. *International Journal of Software Engineering and Knowledge Engineering*, *29*, 139–175. doi:10.1142/S0218194019500074.

Hazarika, D., Poria, S., Gorantla, S., Cambria, E., Zimmermann, R., & Mihalcea, R. (2018). CASCADE: Contextual Sarcasm Detection in Online Discussion Forums. In *Proceedings of the 27th International Conference on Computational Linguistics* (pp. 1837–1848). URL: http://arxiv.org/abs/1805.06413.

Jia, X., Deng, Z., Min, F., & Liu, D. (2019). Three-way decisions based feature fusion for Chinese irony detection. *International Journal of Approximate Reasoning*, *113*, 324–335.

Jorgensen, J. (1996). The functions of sarcastic irony in speech. *Journal of Pragmatics*, *26*, 613–634. doi:10.1016/0378-2166(95)00067-4.

Joshi, A., Bhattacharyya, P., & Carman, M. J. (2017). Automatic sarcasm detection: A survey. *ACM Computing Surveys*, *50*, 1–22. doi:10.1145/3124420.

Joshi, A., Goel, P., Bhattacharyya, P., & Carman, M. (2016a). Automatic Identification of Sarcasm Target: An Introductory Approach. URL: http://arxiv.org/abs/1610.07091.

Joshi, A., Tripathi, V., Bhattacharyya, P., Carman, M., Singh, M., Saraswati, J., & Shukla, R. (2016b). How challenging is sarcasm versus irony classification?: A study with a dataset from English literature. In *Proceedings of the Australasian Language Technology Association Workshop 2016* (pp. 123–127). URL: https://www.aclweb.org/anthology/U16-1013.

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, *2*, 427–431. doi:10.18653/v1/e17-2068.

Justo, R., Corcoran, T., Lukin, S. M., Walker, M., & Torres, M. I. (2014). Extracting relevant knowledge for the detection of sarcasm and nastiness in the social web. *Knowledge-Based Systems*, (pp. 124–133). doi:10.1016/j.knosys.2014.05.021.

Khodak, M., Saunshi, N., & Vodrahalli, K. (2018). A Large Self-Annotated Corpus for Sarcasm. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)* (pp. 641–646). URL: https://www.reddit.com.

King, G., & Zeng, L. (2003). Logistic regression in rare events data. *Journal of Statistical Software*, (pp. 137–163). doi:10.18637/jss.v008.i02.

Kreuz, R. J., & Glucksberg, S. (1989). How to Be Sarcastic: The Echoic Reminder Theory of Verbal Irony. *Journal of Experimental Psychology: General*, *118*, 374. doi:10.1037/0096-3445.118.4.374.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. URL: http://arxiv.org/abs/1909.11942.

Li, M., Lu, Q., Long, Y., & Gui, L. (2017). Inferring affective meanings of words from word embedding. *IEEE Transactions on Affective Computing*, *8*, 443–456.

Ling, J., & Klinger, R. (2016). An empirical, quantitative analysis of the differences between sarcasm and Irony. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9989 LNCS*, 203–216. doi:10.1007/978-3-319-47602-5{_}39.

Liu, B. (2015). *Sentiment analysis: Mining opinions, sentiments, and emotions*. doi:10.1017/CBO9781139084789.

Liu, L., Priestley, J. L., Zhou, Y., Ray, H. E., & Han, M. (2019). A2Text-Net: A Novel Deep Neural Network for Sarcasm Detection. In *2019 IEEE First International Conference on Cognitive Machine Intelligence (CogMI)* (pp. 118–126). doi:10.1109/cogmi48466.2019.00025.

Maipradit, R., Hata, H., & Matsumoto, K. (2019). Sentiment Classification Using N-Gram Inverse Document Frequency and Automated Machine Learning. *IEEE Software*, *36*, 65–70. doi:10.1109/MS.2019.2919573.

Martini, A. T., Farrukh, M., & Ge, H. (2018). Recognition of ironic sentences in Twitter using attention-based LSTM. *International Journal of Advanced Computer Science and Applications*, *9*, 7–11. doi:10.14569/ijacsa.2018.090802.

Maynard, D., & Greenwood, M. A. (2014). Who cares about sarcastic tweets? Investigating the impact of sarcasm on sentiment analysis. In *Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014* (pp. 4238–4243).

Molla, D., & Joshi, A. (2019). Overview of the 2019 ALTA Shared Task : Sarcasm Target Identification. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association* (pp. 192–196).

Nadeau, C., & Bengio, Y. (2003). Inference for the generalization error. *Machine Learning*, (pp. 307–313). doi:10.1023/A:1024068626366.

Nguyen, D., McGillivray, B., & Yasseri, T. (2018). Emo, love and god: making sense of urban dictionary, a crowd-sourced online dictionary. *Royal Society open science*, *5*, 172320.

Olkoniemi, H., Johander, E., & Kaakinen, J. K. (2019). The role of look-backs in the processing of written sarcasm. *Memory & cognition*, *47*, 87–105.

Oprea, S., & Magdy, W. (2019). Exploring Author Context for Detecting Intended vs Perceived Sarcasm. doi:10.18653/v1/p19-1275.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. doi:10.1109/TKDE.2009.191.

Parameswaran, P., Trotman, A., Liesaputra, V., & Eyers, D. (2019). Detecting Target of Sarcasm using Ensemble Methods. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association* (pp. 197–203).

Patro, J., Bansal, S., & Mukherjee, A. (2019). A deep-learning framework to detect sarcasm targets. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 6335–6341). doi:10.18653/v1/d19-1663.

Peng, H., Khashabi, D., & Roth, D. (2015). Solving hard coreference problems. In *NAACL HLT 2015 - 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*. doi:10.3115/v1/n15-1082.

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference* (pp. 1532–1543). doi:10.3115/v1/d14-1162.

Perikos, I., & Hatzilygeroudis, I. (2016). Recognizing emotions in text using ensemble of classifiers. *Engineering Applications of Artificial Intelligence*, *51*, 191–201. doi:10.1016/j.engappai.2016.01.012.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep Contextualized Word Representations. In *Proceedings of NAACL-HLT* (pp. 2227–2237). doi:10.18653/v1/n18-1202.

Pexman, P. M. (2008). It's fascinating research: The cognition of verbal irony. *Current Directions in Psychological Science*, *17*, 286–290.

Qiu, G., Liu, B., Bu, J., & Chen, C. (2011). Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, *37*, 9–27.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners. URL: https://github.com/codelucas/newspaper.

Rajadesingan, A., Zafarani, R., & Liu, H. (2015). Sarcasm detection on Twitter: A behavioral modeling approach. In *WSDM 2015 - Proceedings of the 8th ACM International Conference on Web Search and Data Mining* (pp. 97–106). doi:10.1145/2684822.2685316.

Rao, D., Yarowsky, D., Shreevats, A., & Gupta, M. (2010). Classifying latent user attributes in Twitter. In *International Conference on Information and*

Knowledge Management, Proceedings (pp. 37–44). doi:10.1145/1871985.1871993.

Ribeiro, V., Avila, S., & Valle, E. (2019). Handling Inter-Annotator Agreement for Automated Skin Lesion Segmentation. URL: http://arxiv.org/abs/1906.02415.

Rockwell, P. (2000). Lower, slower, louder: Vocal cues of sarcasm. *Journal of Psycholinguistic Research*, *29*, 483–495. doi:10.1023/A:1005120109296.

Rogers, A., Ananthakrishna, S. H., & Rumshisky, A. (2018). What's in Your Embedding, And How It Predicts Task Performance. In *Proceedings of the 27th International Conference on Computational Linguistics* (pp. 2690–2703). URL: http://ldtoolkit.space.

Sabbagh, M. A. (1999). Communicative intentions and language: Evidence from right-hemisphere damage and autism. *Brain and Language*, *70*, 29–69. doi:10.1006/brln.1999.2139.

Sanfilippo, M. R., Fichman, P., & Yang, S. (2018). Multidimensionality of online trolling behaviors. *Information Society*, *34*, 27–39. doi:10.1080/01972243.2017.1391911.

Sarakit, P., Theeramunkong, T., & Haruechaiyasak, C. (2015). Improving emotion classification in imbalanced YouTube dataset using SMOTE algorithm. In *ICAICTA 2015 - 2015 International Conference on Advanced Informatics: Concepts, Theory and Applications* (pp. 241–252). doi:10.1109/ICAICTA.2015.7335373.

Sergienya, I., & Schütze, H. (2015). Learning better embeddings for rare words using distributional representations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 280–285).

Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., & Summers, R. M. (2016). Deep Convolutional Neural Networks for

Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE Transactions on Medical Imaging*, *35*, 1285–1298. doi:10.1109/TMI.2016.2528162.

Song, J., Huang, X., Qin, S., & Song, Q. (2016). A bi-directional sampling based on K-means method for imbalance text classification. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science, ICIS 2016 - Proceedings* (pp. 1–5). doi:10.1109/ICIS.2016.7550920.

Sperber, D. (1984). Verbal irony: Pretense or echoic mention? *Journal of Experimental Psychology: General*, *113*, 130–136. doi:10.1037/0096-3445.113.1.130.

Tang, D., Qin, B., Feng, X., & Liu, T. (2016). Effective LSTMs for target-dependent sentiment classification. In *COLING 2016 - 26th International Conference on Computational Linguistics, Proceedings of COLING 2016: Technical Papers*.

Tao, J., & Fang, X. (2020a). Toward multi-label sentiment analysis: a transfer learning based approach. *Journal of Big Data*, *7*, 1–26. doi:10.1186/s40537-019-0278-0.

Tao, J., & Fang, X. (2020b). Toward multi-label sentiment analysis: a transfer learning based approach. *Journal of Big Data*, *7*, 1–26. doi:10.1186/s40537-019-0278-0.

Wang, D., & Zheng, T. F. (2015). Transfer Learning for Speech and Language Processing. In *Proceedings of APSIPA Annual Summit and Conference* (pp. 1225–1237).

Wang, Y., Huang, M., Zhao, L., & Zhu, X. (2016). Attention-based LSTM for aspect-level sentiment classification. In *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*. doi:10.18653/v1/d16-1058.

Waseem, Z. (2016). Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter. In *Proceedings of the first workshop on NLP and computational social science* (pp. 138–142). doi:10.18653/v1/w16-5618.

Xue, W., & Li, T. (2018). Aspect based sentiment analysis with gated convolutional networks. In *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*. doi:10.18653/v1/p18-1234.

Zhang, J., Yeung, S. H., Shu, Y., He, B., & Wang, W. (2019a). Efficient memory management for GPU-based deep learning systems. *arXiv preprint arXiv:1903.06631*, .

Zhang, S., Zhang, X., Chan, J., & Rosso, P. (2019b). Irony detection via sentiment-based transfer learning. *Information Processing & Management*, *56*, 1633–1644.

Zhang, X., Zhao, J., & Lecun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems* (pp. 1–9).

Zoph, B., Yuret, D., May, J., & Knight, K. (2016). Transfer learning for low-resource neural machine translation. In *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings* (pp. 1568–1575). doi:10.18653/v1/d16-1163.