TweetNLP: Cutting-Edge Natural Language Processing for Social Media

Jose Camacho-Collados\textsuperscript{1} Kiamehr Rezaee\textsuperscript{1} Talayeh Riahi\textsuperscript{1} Asahi Ushio\textsuperscript{1} Daniel Loureiro\textsuperscript{1} Dimosthenis Antypas\textsuperscript{1} Joanne Boisson\textsuperscript{1,6} Luis Espinosa-Anke\textsuperscript{1,6} Fangyu Liu\textsuperscript{2} Eugenio Martínez-Cámara\textsuperscript{3} Gonzalo Medina\textsuperscript{3} Thomas Buhrmann\textsuperscript{4} Leonardo Neves\textsuperscript{5} Francesco Barbieri\textsuperscript{5}

\textsuperscript{1}Cardiff NLP, Cardiff University, UK  \textsuperscript{2}LTL, University of Cambridge, UK  \textsuperscript{3}DaSCI, University of Granada, Spain  \textsuperscript{4}Graphext, Spain  \textsuperscript{5}Snap Inc., USA  \textsuperscript{6}AMPLYFI, UK
cardiffnlp.contact@gmail.com

Abstract

In this paper we present TweetNLP, an integrated platform for Natural Language Processing (NLP) in social media. TweetNLP supports a diverse set of NLP tasks, including generic focus areas such as sentiment analysis and named entity recognition, as well as social media-specific tasks such as emoji prediction and offensive language identification. Task-specific systems are powered by reasonably-sized Transformer-based language models specialized on social media text (in particular, Twitter) which can be run without the need for dedicated hardware or cloud services. The main contributions of TweetNLP are: (1) an integrated Python library for a modern toolkit supporting social media analysis using our various task-specific models adapted to the social domain; (2) an interactive online demo for codeless experimentation using our models; and (3) a tutorial covering a wide variety of typical social media applications.

1 Introduction

Today’s society cannot be understood without the role of social media. Online users connect more and more via platforms that enable content sharing, either generic or around specific topics, and do this by means of text-only messages, or augmenting them with multimedia content such as pictures, audio or video. As such, these platforms have been used to understand user, group and organization-wide behaviours (Yang et al., 2021; Hu et al., 2021). In particular, Twitter, which is the main platform studied in this paper, has long been an important resource for understanding society at large (Weller et al., 2013). Twitter is interesting for NLP because it embodies many features that are natural in spontaneous and ever-evolving fast-paced communication. However, the majority of NLP research focuses on optimizing model development against training data and evaluation benchmarks which are, at worst, reasonably clean (e.g., news articles, blog posts or Wikipedia). Consequently, when deployed \textit{in the wild}, features such as noisiness, multilinguality, immediacy, slang, technical jargon, lack of context, platform-specific restrictions on message length, emoji and other modalities, etc. become core communicative variables that need to be factored in. Indeed, even traditional NLP tasks such as normalization (Han and Baldwin, 2011; Baldwin et al., 2015), POS tagging (Derczynski et al., 2013), sentiment analysis (Poria et al., 2020) or named entity recognition (Ritter et al., 2011; Baldwin et al., 2013) have been shown to produce suboptimal results in the context of social media.

Given the above, we put forward TweetNLP (tweetnlp.org), which offers a full-fledged NLP platform specialized in Twitter. The backbone of TweetNLP consists of Transformer-based language models that have been trained on Twitter (Barbieri et al., 2020, 2022; Loureiro et al., 2022). Then, these specialized language models have been further fine-tuned for specific NLP tasks on Twitter data. These models have already proved highly popular, with thousands of downloads from the Hugging Face model hub every month (Wolf et al., 2020).\textsuperscript{1} TweetNLP integrates all these resources into a single platform. With a simple Python API, TweetNLP offers an easy-to-use way to leverage cutting-edge NLP models in a variety of social media tasks. Despite the trend of ever-larger language models (Shoeybi et al., 2019; Brown et al., 2020), TweetNLP is more focused on the general user and applicability, and therefore integrates base models which are easily run in standard computers or on free cloud services. Finally, all models can be accessed from an interactive online demo, offering anyone the possibility to test models and perform

\textsuperscript{1}Most notably, the sentiment analysis model has been the most downloaded model in the Hugging Face model hub in January 2022, with over 15M downloads. Similarly, the TweetEval benchmark, in which most task-specific Twitter models are fine-tuned, has been the second most downloaded dataset in April 2022, with over 150K downloads.
real-time analysis on Twitter.

2 Related Work

General-purpose NLP libraries have been available for many years. Starting from the Java-based CoreNLP (Manning et al., 2014) to the more recent Python-based library Stanza (Qi et al., 2020). More recently, libraries such as spaCy have been ubiquitous in NLP, both in research and industry. Finally, in the language models and Transformers era, the Hugging Face Transformer hub has become indispensable for state-of-the-art NLP (Wolf et al., 2020), which is also leveraged for our library TweetNLP. However, none of these libraries is specialized in social media or Twitter.

As for libraries developed specifically for social media, these are more limited and mostly associated with low-level tasks such as tokenization, part-of-speech (Owoputi et al., 2013) tagging or dependency parsing (Kong et al., 2014), and initially available for Java. The most recent Twitter-specific Python library is TweebankNLP (Jiang et al., 2022) based on Stanza. This library provides state-of-the-art models on tokenization and lemmatization, besides competitive models on NER, part-of-speech tagging and dependency parsing. In contrast, TweetNLP focuses on specialized Twitter-specific language models for downstream tasks such as sentiment analysis and offensive language identification.

3 Models and Functionalities

TweetNLP is versatile in that it covers a wide range of tasks and applications. The backbone of TweetNLP are transformer-based language models, which are covered in Section 3.1. The concrete NLP tasks integrated in TweetNLP are presented in Section 3.2. Finally, in Section 3.3 we present embeddings used to represent words and tweets. All TweetNLP model checkpoints are available in the appendix.

3.1 Language models

Language models are at the core of TweetNLP. Instead of relying on general-purpose models (Devlin et al., 2019) or training a language model from scratch (Nguyen et al., 2020), we start from RoBERTa (Liu et al., 2019) and XLM-R (Conneau et al., 2020) checkpoints and continue pre-training on Twitter-specific corpora. This was shown to be generally more reliable for the amount of text analysed in Barbieri et al. (2020). Given our aim for democratizing the usage of specialized language models for social media, another important feature of TweetNLP is the relatively small size of the language models. To this end, all language models rely on the equivalent of a RoBERTa-base or XLM-R-base architecture. These models are efficient on standard hardware and free-tier cloud computing services, with reasonable speed even without GPU support.

TweetEval (Barbieri et al., 2020). This model was initially released as part of the TweetEval project. It is based on a RoBERTa-base architecture using the original model as an initial check point (Liu et al., 2019). Later, this language model was further pre-trained on a corpus of 60M tweets from May 2018 to August 2019.

TimeLMs (Loureiro et al., 2022). This model is initially trained on the same Twitter corpus used by Barbieri et al. (2020). The main difference lies on a few preprocessing improvements applied to the underlying corpus, including measures to reduce potential spam and near duplicates, and more recent corpora used for continual pretraining. The overall quantity of tweets is therefore larger, as the model is regularly updated (every 3 months) with a fixed number of additional tweets. The most recently released TimeLMs model to date is pre-trained on 132M tweets until the end of June 2022.

XLM-T (Barbieri et al., 2022). This model was trained on 198M tweets on over thirty languages from May 2018 to March 2020, following a similar strategy to Barbieri et al. (2020). In this case, the initial checkpoint was XLM-R-base (Conneau et al., 2020).

3.2 Supported tasks

In the following we describe the tasks supported by TweetNLP. For the tweet classification tasks included in TweetEval, and for topic classification, we simply fine-tune the models described above on the corresponding datasets, as described in Barbieri et al. (2020). For model fine-tuning on named entity recognition, we rely on the T-NER library (Ushio and Camacho-Collados, 2021), which is also integrated into TweetNLP.

Sentiment Analysis. The sentiment analysis task integrated in TweetNLP consists of predicting the
sentiment of a tweet with one of the three following labels: positive, neutral or negative. The base dataset for English is the unified TweetEval version of the Semeval-2017 dataset from the task on Sentiment Analysis in Twitter (Rosenthal et al., 2017). Moreover, for the languages other than English we include the datasets integrated in UMSAB (Barbieri et al., 2022), namely Arabic (Rosenthal et al., 2017), French (Benamara et al., 2017), German (Cieliebak et al., 2017), Hindi (Patra et al., 2015), Italian (Barbieri et al., 2016), Portuguese (Brum and Volpe Nunes, 2018), and Spanish (Díaz-Galiano et al., 2018).

Emotion Recognition. Given a tweet, this task consists of associating it with its most appropriate emotion. As a reference dataset we use the SemEval 2018 task on Affect in Tweets (Mohammad et al., 2018), simplified to only the four emotions used in TweetEval: anger, joy, sadness and optimism.

Emoji Prediction. The goal of emoji prediction is to predict the final emoji on a given tweet. The dataset used to fine-tune our models is the TweetEval adaptation from the SemEval 2018 task on Emoji Prediction (Barbieri et al., 2018), including 20 emoji as labels.

Irony Detection. This is a binary classification task that aims at detecting whether a tweet is ironic or not. It is based on the Irony Detection dataset from the SemEval 2018 task (Van Hee et al., 2018).

Hate Speech Detection. The hate speech dataset consists of detecting whether a tweet is hateful towards women or immigrants. It is based on the Detection of Hate Speech task at SemEval 2019 (Basile et al., 2019).

Offensive Language Identification. The task consist of identifying any form of offensive language in a tweet. The dataset is based on the SemEval 2019 task on Identifying Offensive Language in Social Media (Zampieri et al., 2019).

Stance Detection. Given a target topic and a tweet, stance detection consists of assessing whether the author of the tweet has a positive, neutral or negative position towards the target. The dataset considered was initially released for the SemEval 2016 task on Detecting Stance in Tweets (Mohammad et al., 2016).

Topic Classification. The aim of this task is, given a tweet, assign topics related to its content. The task is formulated as a supervised multi-label classification problem where each tweet is assigned one or more topics from a total of 19 available topics. The topics were carefully curated based on Twitter trends with the aim to be broad and general, consisting of classes such as: arts and culture, music, or sports. The underlying tweet topic classification dataset contains over 10K manually-labeled tweets (Antypas et al., 2022).

Named Entity Recognition. The goal of named entity recognition (NER) is to find entities and identify their entity types in a given sentence. The underlying Twitter NER dataset is composed of over 10K tweets which were annotated (internally) with seven entity types.1

3.3 Embeddings

In addition to the language models and their supported tasks, we also release high-quality vector representation models for words and tweets, i.e., embeddings (Pilehvar and Camacho-Collados, 2020). These relatively low-dimensional vector representations can be exploited for a different range of applications and analyses such as word/tweet similarity or tweet retrieval, to name a few.

Word embeddings. TweetNLP word embeddings are based on fastText (Bojanowski et al., 2017) and trained on the same corpora used to train the language models described in Section 3.1. In particular, we trained two sets of embeddings: (1) a monolingual English model trained with the TimeLMs Twitter corpus until the end of 2021; and (2) a multilingual model trained with the Twitter corpus used for XLM-T. Both models were trained using the official fastText package with 300 dimensions, minimum n-gram size 2, maximum n-gram size 12, and remaining parameters set to defaults.

Tweet embeddings. For tweet embeddings, we pulled tweet-reply pairs from the Twitter API and trained contrastive embeddings with an InfoNCE loss (Oord et al., 2018). For tweets with multiple replies, we randomly sampled one reply. In training, one mini-batch is composed of a list of tweet-reply pairs. The tweet-reply pairs are regarded as

1More details about the datasets for topic classification and named entity recognition will be provided at a later stage. Datasets were annotated internally in Snap and we are working on releasing them to the public according to regulations.
positive samples; the enumeration of all other possible combination of tweet-reply, tweet-tweet, and tweet-reply pairs are regarded as negative samples. The contrastive InfoNCE loss then pulls positive pair representations close while pushes negative representations away from each other. Training was performed on 1.1M tweet-reply pairs, and we collected a separate tweet-reply set of 10k pairs for selecting the model checkpoint.

4 TweetNLP Python library

The TweetNLP Python library has been integrated into pypi\(^4\) and therefore is easily accessible and can be installed from pip ("pip install tweetnlp"). All the details on how to use TweetNLP are in the associated Github repository, which is released fully open-source: https://github.com/cardiffnlp/tweetnlp.

Once installed, loading and using a fine-tuned model on any specific task can be done as follows.

```python
from tweetnlp import load
tweet = "I love Paris!!"
# Sentiment Analysis
model = load('sentiment')
model.sentiment(tweet)
# Tweet Embeddings
model = load('sentence_embedding')
model.embedding(tweet)
# Masked Language Model
model = load('language_model')
tweet = "I love <mask>!!"
model.mask_prediction(tweet)
```

With the `load` statement, the associated fine-tuned language models are loaded in the background. Users can then get the predictions for any given sentence or tweet with a simple pre-defined function (e.g., `sentiment` or `predict`). Custom loading of existing fine-tuned language models not included in TweetNLP is also possible. The same functionalities apply to all the other tasks described in Section 3.2.

5 Tutorials

In addition to the Python library presented in the previous section, TweetNLP offers access to the underlying Python code structured in instructive Google Colab notebooks with starter code and examples (https://tweetnlp.org/get-started/). These notebooks are aimed at users with varying degrees of experience in NLP and social media processing. In the following we list the currently existing tutorials and a brief description:

**Introduction to TweetNLP.** In this initial introduction, users learn how to use the TweetNLP Python library to make use of specialized models in social media for a wide variety of tasks from sentiment analysis to named entity recognition.

**Getting data from Twitter.** This notebook helps users understand the Twitter API\(^5\) and how to interact with it. More importantly, there are concrete examples on how to retrieve data (i.e. tweets) from Twitter, usually given a hashtag or a keyword.

**Custom fine-tuning.** In this notebook users can learn how to fine-tune any given language model on a specific task (e.g. sentiment analysis). For this, we will take advantage of the TweetEval task data and unified format (Barbieri et al., 2020). Additionally, users can learn how to easily evaluate language models on TweetEval.

**Word embeddings.** With this notebook users can learn how to train their own word embeddings on custom data using Gensim\(^6\) (ˇReh˚uˇrek and Sojka, 2010). The notebook also includes examples on how to get similarity scores from Twitter-specific word embeddings, or how to obtain the nearest neighbour words from a given input word.

**Language models over time.** This notebook leverages the TimeLMs library (Loureiro et al., 2022). Users can learn how to make use of language models that have been trained in short periods of time since 2019 until recently.

**Tweet embeddings.** This notebook contains examples on how to transform a tweet into a vector (embedding) and how these enable important applications such as tweet similarity and retrieval.

6 Demo

In addition to the Python-based library and tutorials, we developed a comprehensive web-based demo integrating all our models, available at https://tweetnlp.org/demo/. The goal of the demo is for any user to be able to test our models and get predictions. In particular, the model includes the following five functionalities:

---

\(^4\)https://pypi.org/project/tweetnlp/
\(^5\)https://developer.twitter.com/en/docs/twitter-api
\(^6\)https://radimrehurek.com/gensim/
Sentence/tweet classification (Figure 1). Users can input a sentence or a tweet (including a tweet URL) and the output is a plot display of the confidence of the model with respect to its predictions. This demo includes all tweet classification tasks supported in English (see Section 3.2), as well as a multilingual sentiment analysis model based on XLM-T.

Figure 1: TweetNLP tweet classification demo.

Hashtag analysis (Figure 2). This demo directly interacts with the Twitter API. Users can type a hashtag (or any keyword), initial and end dates, task and language. The system will then retrieve tweets for the given time interval and compute an aggregated analysis of the results. Languages supported for this demo are available in the appendix.

Figure 2: TweetNLP hashtag analysis demo. The output is a bar plot that shows the sentiment of the retrieved tweets over time for the input hashtag #NLProc.

Word prediction (Figure 3). Masked language models utilized in TweetNLP are trained to predict unknown (or masked) words within a sentence. For this demo, users can input a sentence with a masked word and the system will show the most likely words as given by the masked language model, in order of confidence.

Figure 3: TweetNLP word prediction demo.

Tweet similarity (Figure 4). Given two short pieces of text (e.g., two sentences or two tweets), this demo displays their cosine similarity score on a 0-100 scale as provided by our default tweet embedding model.

Figure 4: TweetNLP tweet similarity demo.

Named Entity Recognition (Figure 5). Given a tweet or a sentence, this NER demo locates its named entities and infers their types.

7 Evaluation

In this section, we provide experimental results of the default models integrated into TweetNLP.

7.1 Experimental setting

Datasets. For the evaluation we utilized all the train/validation/test splits described in Section 3.2. In particular, we relied on the TweetEval-released datasets for all tweet classification tasks except for topic classification.
Figure 5: TweetNLP Named Entity Recognition demo.

Default TweetNLP language models. While in TweetNLP all Twitter-specific language models are included, we use as a default (1) TimeLMs trained until December 2021 for English and (2) XLM-T for the languages other than English and multilingual tasks. These models are then fine-tuned to the corresponding tasks as described in Section 3.2.

Comparison systems. We report the performance of all original TweetEval baselines (Barbieri et al., 2020): a frequency-based SVM classifier, fastText (Joulin et al., 2017), a Bidirectional LSTM, RoBERTa-base (Liu et al., 2019), a RoBERTa-base model trained on Twitter from scratch (RoBERT-twitter) and the original TweetEval RoBERTa-base model. As another baseline we include BERTweet (Nguyen et al., 2020), trained on almost 1 billion tweets from 2013 to 2019.

Language model fine-tuning. Fine-tuning is performed on the training sets of each corresponding dataset, using their corresponding development sets for validation. We followed TweetEval training protocols for tweet classification, where only the learning rate and number of epochs are tuned (Barbieri et al., 2020). All reported results for language models are based on an average of three runs.

7.2 Results

Table 1 shows the main results of our TweetNLP default language model and comparison systems on nine Twitter-based tasks. The default TimeLMs-21 model achieves the overall results on most tasks, especially comparing it with a comparable general-purpose RoBERTa-based model. In the following we also provide details of our experimental results on languages other than English, and for the integrated word and tweet embedding models.

Multilingual sentiment analysis results. In addition to the English evaluation, we report results on multilingual sentiment analysis (Table 2). The evaluation is performed on the UMSAB multilingual sentiment analysis benchmark (Barbieri et al., 2022). For this evaluation we compare XLM-T fine-tuned on all the language-specific training sets of UMSAB with XLM-R (Conneau et al., 2020) using the same fine-tuning strategy. As an additional indicative baseline, we include fastText trained on the language-specific training sets. As can be observed, our domain-specific XLM-T language model achieves the best overall results in all languages, further reinforcing the importance of in-domain language model training.

Word embedding results. As a sanity check to verify the quality of the word embeddings, we simply test them on standard word similarity datasets: The WS-Sim similarity and WS-Rel relatedness subsets (Agirre et al., 2009) from WordSim-353 (Finkelstein et al., 2002), SemEval-2017 (Camacho-Collados et al., 2017) and MEN (Bruni et al., 2014). Then, we compared the results with the pre-trained fastText model trained on the Common Crawl (Bojanowski et al., 2017), and Wikipedia. According to Spearman correlation, the results of our Twitter embeddings were 0.77 (WS-Sim), 0.72 (WS-Rel), 0.69 (SemEval), and 0.79 (MEN). In contrast, the pre-trained fastText Common Crawl results were 0.84 (WS-Sim), 0.64 (WS-Rel), 0.67 (SemEval), and 0.81 (MEN). We should note that these datasets are not specific to social media and even so, our trained embeddings outperform the standard pre-trained fastText in two datasets. In particular, there seems to be a marked difference between similarity and relatedness, where our Twitter embeddings appear to be more suited to relatedness.

Tweet embedding results. For tweet embeddings we explore a tweet retrieval task setting which consists of finding the reply to a given tweet from the 10k replies in the search space. We randomly sampled 3k tweet-reply pairs that do not overlap with the training set.

---

7The BERTweet result on Irony is marked with * as their pre-training corpus overlapped with the Irony dataset, which was constructed using distant supervision.

8While not directly comparable given the different sizes, we also compared with our previously-released Twitter-specific 100-dimensional fastText embeddings (Camacho-Collados et al., 2020). The results for these embeddings were consistently lower: 0.65 (WS-Sim), 0.43 (WS-Rel), 0.52 (SemEval), and 0.76 (MEN).
Table 1: Test results in the nine TweetNLP-supported tasks.

<table>
<thead>
<tr>
<th>Model</th>
<th>Emoji</th>
<th>Emotion</th>
<th>Hate</th>
<th>Irony</th>
<th>Offensive</th>
<th>Sentiment</th>
<th>Stance</th>
<th>Topic</th>
<th>NER</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>29.3</td>
<td>64.7</td>
<td>36.7</td>
<td>61.7</td>
<td>52.3</td>
<td>62.9</td>
<td>67.3</td>
<td>30.5</td>
<td>-</td>
</tr>
<tr>
<td>fastText</td>
<td>25.8</td>
<td>65.2</td>
<td>50.6</td>
<td>63.1</td>
<td>73.4</td>
<td>62.9</td>
<td>65.4</td>
<td>24.0</td>
<td>-</td>
</tr>
<tr>
<td>BLSTM</td>
<td>24.7</td>
<td>66.0</td>
<td>52.6</td>
<td>62.8</td>
<td>71.7</td>
<td>58.3</td>
<td>59.4</td>
<td>27.0</td>
<td>-</td>
</tr>
<tr>
<td>RoB-Base</td>
<td>30.9</td>
<td>76.1</td>
<td>46.6</td>
<td>59.7</td>
<td>79.5</td>
<td>71.3</td>
<td>68.0</td>
<td>50.1</td>
<td>58.0</td>
</tr>
<tr>
<td>RoB-Twitter</td>
<td>29.3</td>
<td>72.0</td>
<td>46.9</td>
<td>65.4</td>
<td>77.1</td>
<td>69.1</td>
<td>66.7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TweetEval</td>
<td>31.4</td>
<td>78.5</td>
<td>52.3</td>
<td>61.7</td>
<td>80.5</td>
<td>72.6</td>
<td>69.3</td>
<td>56.8</td>
<td>56.8</td>
</tr>
<tr>
<td>BERTweet</td>
<td>33.4</td>
<td>79.3</td>
<td>56.4</td>
<td>82.1</td>
<td>79.5</td>
<td>73.4</td>
<td>71.2</td>
<td>52.7</td>
<td>58.7</td>
</tr>
<tr>
<td>TweetNLP (TimeLMs-21)</td>
<td>34.0</td>
<td>80.2</td>
<td>55.1</td>
<td>64.5</td>
<td>82.2</td>
<td>73.7</td>
<td>72.9</td>
<td>58.8</td>
<td>59.7</td>
</tr>
<tr>
<td>Evaluation metric</td>
<td>M-F1</td>
<td>M-F1</td>
<td>M-F1</td>
<td>F1</td>
<td>M-F1</td>
<td>M-Rec</td>
<td>AVG(F)</td>
<td>M-F1</td>
<td>M-F1</td>
</tr>
</tbody>
</table>

Table 2: Sentiment analysis results (Macro-F1) on the UMSAB unified benchmark. XLM-R and TweetNLP models are fine-tuned on the training sets of all languages.

<table>
<thead>
<tr>
<th>Model</th>
<th>Arabic</th>
<th>English</th>
<th>French</th>
<th>German</th>
<th>Hindi</th>
<th>Italian</th>
<th>Portuguese</th>
<th>Spanish</th>
<th>ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>fastText</td>
<td>45.98</td>
<td>50.85</td>
<td>54.82</td>
<td>59.56</td>
<td>37.08</td>
<td>54.65</td>
<td>55.05</td>
<td>50.06</td>
<td>51.01</td>
</tr>
<tr>
<td>XLM-R</td>
<td>64.31</td>
<td>68.52</td>
<td>70.52</td>
<td>72.84</td>
<td>53.39</td>
<td>68.62</td>
<td>69.79</td>
<td>66.03</td>
<td>66.75</td>
</tr>
<tr>
<td>TweetNLP (XLM-T)</td>
<td>66.89</td>
<td>70.63</td>
<td>71.18</td>
<td>77.35</td>
<td>56.35</td>
<td>69.06</td>
<td>75.42</td>
<td>68.52</td>
<td>67.91</td>
</tr>
</tbody>
</table>

Table 3: Results of sentence and tweet embedding models on tweet-reply retrieval and the STS-benchmark.

<table>
<thead>
<tr>
<th>Model</th>
<th>Retrieval</th>
<th>STS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentence-BERT</td>
<td>6.1</td>
<td>77.0</td>
</tr>
<tr>
<td>all-mpnet-base-v2</td>
<td>15.8</td>
<td>83.4</td>
</tr>
<tr>
<td>Mirror-RoBERTa</td>
<td>8.8</td>
<td>79.6</td>
</tr>
<tr>
<td>SimCSE-RoBERTa</td>
<td>9.2</td>
<td>80.3</td>
</tr>
<tr>
<td>TweetNLP (Tweet-embeddings)</td>
<td>26.7</td>
<td>70.7</td>
</tr>
</tbody>
</table>

overlapping with training data and split them into 3 sets of 1k pairs. We report accuracy@1 and average models’ performance on the 3 sets. We also include results on sentence similarity, using the STS-benchmark (Cer et al., 2017) and reporting Spearman’s correlation. We list tweet-reply retrieval accuracy and STS-benchmark Spearman’s correlation in Table 3. We compare with recent supervised (Reimers and Gurevych, 2019, Sentence-BERT; all-mpnet-base-v2), and unsupervised (Liu et al., 2021, Mirror-BERT), (Gao et al., 2021, SimCSE) sentence embedding models. On the task of tweet-reply retrieval, our tweet-embeddings model significantly outperforms all-mpnet-base-v2 trained with around 1B sentence pairs. This highlights the importance of in-domain training. On the STS-Benchmark, all-mpnet-base-v2 achieves the best performance and our tweet-embeddings perform the worst among baselines but they are generally in a similar ballpark. To complement this evaluation, we plan to test our tweet embeddings with a textual similarity dataset in the tweet domain in the future.

8 Conclusion and Future Work

In this demo paper we have presented TweetNLP, an all-round platform for NLP specialized in social media. The platform is powered by relatively lightweight language models trained on Twitter, and adapted (fine-tuned) to various popular NLP tasks on social media, such as sentiment analysis and offensive language identification. In addition to sharing the models, TweetNLP provides an online demo, a Python library, and a tutorial to make the most of the models, regardless of the expertise of the user. TweetNLP also enables easy inspection of the models by non-programmers, which can help identify harmful biases or errors, that in turn would help improve the models in the future.

While this first release version of TweetNLP is self-contained and complete, our goal is to keep updating it with both new models and tasks. Since social media data is at the core of TweetNLP, we are planning to develop new datasets and models for social media tasks. In particular, our idea is to go beyond tweet classification tasks, which are currently well covered in TweetNLP. For instance, low-level tasks such as syntactic parsing and part-of-speech tagging has been traditionally hard in noisy environments such as social media. Finally, in the future we are also planning to extend TweetNLP to other social media platforms such as Reddit, LinkedIn or Instagram, and to provide support for languages other than English in a wider variety of tasks.

9Baseline checkpoint links are included in the Appendix.
9 Impact Statement

This paper deals with social media data, in particular with Twitter. All Twitter regulations were followed and data was extracted through the official Twitter API. To mitigate the potential effect of working with this type of data, all dataset-related tweets were anonymized, with URLs removed. In most cases dataset creators made an effort to remove offensive or harmful content from the tweets. Nonetheless, models trained on this data may amplify existing biases present in the social media platform. While this is in many cases unavoidable, we hope that by making this demo public with model prototypes, experts will be able to more easily inspect these biases and we will be able to better understand the potential biases of models trained on this type of data.

Acknowledgements

We acknowledge the support of UKRI (in particular the UKRI Future Leaders Fellowship scheme), Snap Inc., the Cardiff University Innovation for All scheme and the R&D&I grant PID2020-116118GA-100 funded by MCIN/AEI/10.13039/501100011033 for partially funding this project.

References


A Languages supported

In addition to English, the sentiment analysis demo (including hashtag analysis) is also available for the following languages: Amharic, Arabic, Armenian, Basque, Bengali, Bulgarian, Burmese, Catalan, Chinese, Czech, Danish, Dhivehi, Dutch, Estonian, Finnish, French, Georgian, German, Greek, Haitian, Hebrew, Hindi, Hungarian, Icelandic, Indonesian, Italian, Japanese, Kannada, Khmer, Korean, Kurdish, Lao, Latvian, Lithuanian, Malayalam, Marathi, Nepali, Norwegian, Oriya, Panjabi, Persian, Polish, Pushto, Romanian, Russian, Serbian, Sindhi, Sinhala, Slovenian, Spanish, Swedish, Tagalog, Tamil, Telegu, Thai, Turkish, Ukranian, Urdu, Vietnamese, Welsh. These languages are supported both by a XLM-T multilingual model and the Twitter API.

B Model Links

Table 4 lists all TweetNLP models and their corresponding Hugging Face model hub links.

We release the word embeddings along with Gensim-optimized versions: (1) English-monolingual word embeddings are available at https://tweetnlp.org/downloads/twitter-2021-124m-300d.new.bin; (2) Multilingual word embeddings are available at https://tweetnlp.org/downloads/twitter-multilingual-300d.new.bin.

Table 5 lists the baselines used for the evaluation (Section 7) and their corresponding Hugging Face hub links.
<table>
<thead>
<tr>
<th>Model</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>TweetEval</td>
<td><a href="https://huggingface.co/cardiffnlp/twitter-roberta-base">https://huggingface.co/cardiffnlp/twitter-roberta-base</a></td>
</tr>
<tr>
<td>TimeLMs-21 (default)</td>
<td><a href="https://huggingface.co/cardiffnlp/twitter-roberta-base-2021-124m">https://huggingface.co/cardiffnlp/twitter-roberta-base-2021-124m</a></td>
</tr>
<tr>
<td>XLM-T</td>
<td><a href="https://huggingface.co/cardiffnlp/twitter-xlm-roberta-base">https://huggingface.co/cardiffnlp/twitter-xlm-roberta-base</a></td>
</tr>
<tr>
<td>Sentiment Analysis</td>
<td><a href="https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest">https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest</a></td>
</tr>
<tr>
<td>Multilingual Sentiment Analysis</td>
<td><a href="https://huggingface.co/cardiffnlp/twitter-xlm-roberta-base-sentiment">https://huggingface.co/cardiffnlp/twitter-xlm-roberta-base-sentiment</a></td>
</tr>
<tr>
<td>Emotion Recognition</td>
<td><a href="https://huggingface.co/cardiffnlp/twitter-roberta-base-emotion">https://huggingface.co/cardiffnlp/twitter-roberta-base-emotion</a></td>
</tr>
<tr>
<td>Emojis Prediction</td>
<td><a href="https://huggingface.co/cardiffnlp/twitter-roberta-base-emoji">https://huggingface.co/cardiffnlp/twitter-roberta-base-emoji</a></td>
</tr>
<tr>
<td>Irony Detection</td>
<td><a href="https://huggingface.co/cardiffnlp/twitter-roberta-base-irony">https://huggingface.co/cardiffnlp/twitter-roberta-base-irony</a></td>
</tr>
<tr>
<td>Hate Speech Detection</td>
<td><a href="https://huggingface.co/cardiffnlp/twitter-roberta-base-hate">https://huggingface.co/cardiffnlp/twitter-roberta-base-hate</a></td>
</tr>
<tr>
<td>Offensive Language Identification</td>
<td><a href="https://huggingface.co/cardiffnlp/twitter-roberta-base-offensive">https://huggingface.co/cardiffnlp/twitter-roberta-base-offensive</a></td>
</tr>
<tr>
<td>Stance Detection (abortion)</td>
<td><a href="https://huggingface.co/cardiffnlp/twitter-roberta-base-stance-abortion">https://huggingface.co/cardiffnlp/twitter-roberta-base-stance-abortion</a></td>
</tr>
<tr>
<td>Topic Classification</td>
<td><a href="https://huggingface.co/cardiffnlp/tweet-topic-21-multi">https://huggingface.co/cardiffnlp/tweet-topic-21-multi</a></td>
</tr>
<tr>
<td>Named Entity Recognition</td>
<td><a href="https://huggingface.co/tner/twitter-roberta-base-dec2021-tweetner7-all">https://huggingface.co/tner/twitter-roberta-base-dec2021-tweetner7-all</a></td>
</tr>
<tr>
<td>Tweet Embeddings</td>
<td><a href="https://huggingface.co/cambridgeltl/tweet-roberta-base-embeddings-v1">https://huggingface.co/cambridgeltl/tweet-roberta-base-embeddings-v1</a></td>
</tr>
</tbody>
</table>

Table 4: Hugging Face model links of all the NLP models included in TweetNLP (if available).

<table>
<thead>
<tr>
<th>Model</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoBERTa-base</td>
<td><a href="https://huggingface.co/roberta-base">https://huggingface.co/roberta-base</a></td>
</tr>
<tr>
<td>XLM-R</td>
<td><a href="https://huggingface.co/xlm-roberta-base">https://huggingface.co/xlm-roberta-base</a></td>
</tr>
<tr>
<td>BERTweet</td>
<td><a href="https://huggingface.co/vinai/bertweet-base">https://huggingface.co/vinai/bertweet-base</a></td>
</tr>
<tr>
<td>Sentence-BERT</td>
<td><a href="https://huggingface.co/sentence-transformers/bert-base-nli-mean-tokens">https://huggingface.co/sentence-transformers/bert-base-nli-mean-tokens</a></td>
</tr>
<tr>
<td>all-mpnet-base-v2</td>
<td><a href="https://huggingface.co/sentence-transformers/all-mpnet-base-v2">https://huggingface.co/sentence-transformers/all-mpnet-base-v2</a></td>
</tr>
<tr>
<td>Mirror-RoBERTa</td>
<td><a href="https://huggingface.co/cambridgeltl/mirror-roberta-base-sentence-drophead">https://huggingface.co/cambridgeltl/mirror-roberta-base-sentence-drophead</a></td>
</tr>
<tr>
<td>SimCSE-RoBERTa</td>
<td><a href="https://huggingface.co/princeton-nlp/unsup-simcse-roberta-base">https://huggingface.co/princeton-nlp/unsup-simcse-roberta-base</a></td>
</tr>
</tbody>
</table>

Table 5: Baseline models’ Hugging Face links (if available).