Slang recognition in context of spellchecking & fine tuning BERT to it NLP Project Report

Olaf Werner, Wojciech Replin

June 6, 2022

1 Introduction

People often use slang to communicate on the internet. This can cause problems when analysing them. In this project we tried to fine tune models to slang for two tasks: grammatical correctness and equivalence. As an example of problem for grammatical correctness let's take sentence "ure da best" which means "you are the best". Slang version of this sentence even if it is using different words is still grammatically correct. For sentence equivalence we also can have problems. Let's take pair of sentences "My homie is pro gamer" and "My dude is gud @ games". They are semantically equivalent and both mean "My friend is good at games". Without further training models may not be able to handle slang.

2 Project goal

The project is focused around expanding spellchecking NLP models by the use of slang. The project is going to cover following topics:

1. Investigate sources of slang to feed models with
2. Investigate spellchecking algorithms which capabilities can be extended by the use of slang
3. Investigate how task of verifying grammatical correctness can benefit from using slang
4. Investigate how task of evaluating sentence similarity can benefit from using slang

3 Datasets

3.1 Organic

There were two major organic datasets we considered using for the purposes of this project:

1. *2016 USA Presidential election tweets-61M rows* - this dataset contains 61 million rows with tweet texts labeled with two features *polarity* & *subjectivity*
2. *Slangvolution Twitter Data - Tweets from 2010,2020* - this dataset contains 225 thousands of tweets labeled with words of interest they contain (e.g. slang words). Tweets are seperated with respect to the word of interest being slang or not

Both of these datasets have characteristics of organic internet phrases to be found on the Internet:

1. Dirty - tweets contained in them are polluted with mentions, hashtags, links. These features cannot be easily filtered out, as often hashtag is a meaningful part of the sentence
2. Unlabeled with respect to tweet text being grammatically correct or not
3. Hard to label

4. Hard to tell slang apart from error
5. Unknown amount & kind of slang present in the data

Due to the number of flaws (some of them fatal in context of the project) organic datasets have, we turned to other sources of data to conduct the project.

3.2 Synthetic

Synthetic approach to generating data is composed of two steps:

1. Acquire labeled sentences without slang
2. Transform acquired sentences through a slang translator & infer label from the original data

By using this approach our data will have following characteristics:

1. Clean
2. Labeled though inference
3. Data authenticity limited only by the transformer

This approach may not seem like it'll produce data which can be seen in the real world, but when we think about how authentic slang is written:

1. A human conceives a thought (a formal sentence)
2. Conceived thought is written on a social media platform in slang, thus ran through some kind of transformation

we can see that synthetic approach is surprisingly similar to how slang sentences are produced in reality.

We gathered formal sentences from two sources:

1. *The Corpus of Linguistic Acceptability* - source of 8551 sentences labeled with respect to gramatical correctness
2. *Microsoft Research Paraphrase Corpus* - source of 5801 pairs of sentences which are labeled with respect to sentence equivalence

Then, formal sentences were transformed with https://lingojam.com/InternetSlangTranslator2-0, which:

1. is easily scrapable using simple JavaScript code
2. applies over 6.4 thousand rules (we couldn't reach the author to evaluate them, however empirical results surpassed our expectations)

Examples of transformations done by the transformer:

1. The child almost hurt the small dog
   The zoomer almost hurt the smol doggo
2. Will you please answer that phone
   Will u pls answer tht ph1
3. These pills do less good than others
   Deez pills do less gud then others

Having transformed formal sentences, labels were inferred as following:

1. Grammatical correctness - sentence after adding slang has the same label as original sentence
2. Sentence equivalence - sentences after adding slang are considered equivalent to the original sentence, and the sentence that was possibly equivalent to it & its transformed version

4 Spellchecking approaches

4.1 Algorithmic based on $n$grams & edit distance

This section is written based on http://norvig.com/spell-correct.html
Basic spellchecker model based on an edit distance states the problem as following given the word w, find a correction c that maximizes probability, that w was written when c was actually meant:

$$c = \underset{c \in candidates(w)}{\arg\max} \; P(c|w)$$

by applying Bayes rule:

$$c = \underset{c \in candidates(w)}{\arg\max} \; \frac{P(w|c)P(c)}{P(w)}$$

and since $P(w)$ is a constant, we get:

$$c = \underset{c \in candidates(w)}{\arg\max} \; P(w|c)P(c)$$

There are few components to this approach:

1. Candidate model $candidates(w)$
2. Language model $P(c)$ being the probability that c is an English word
3. Error model $P(w|c)$ being the probability that w would be typed in a text when the author meant c

Candidate model

In order to build a candidate model, one can consider two approaches:

1. Algorithmic (based on edit distance)
2. Statistical (based on $n$gram data)

Algorithmic approach will generate following set of candidate words:

$$candidates(w) = \{c \mid d(w, c) \leq 2\}$$

where $d(w, c)$ denotes edit distance [1] between words w and c. This will yield a spellchecker which is not context-aware. In order to build a spellchecker which is context-aware, let us assume that we have a set D of English $n$grams ($n$tuples of words appearing in English text). Then:

$$candidates(w) = \{c \mid \exists_{d \in D} \exists_u \exists_z u \text{ prepends } w \wedge w \text{ prepends } z \wedge d = ucz\}$$

To put it simply: candidates will be selected, if they can replace w to form a valid $n$gram with neighbourhood of w.

Language model

There are several approaches to be taken to build a language model:

1. Unigram
2. $n$gram
3. Neural network

The simplest approach is the unigram model, which is constructed as follows:

1. Take a large body of text in english
2. Count occurrences of each word

3. Compute probabilities based on computed occurrences

This approach will not be context-aware. In order to build context-aware spellchecker, we can turn to $n$gram language model construction which is actually a generalization of unigram approach:

$$P(w|w_{i-1}, ..., w_{i-n}) = \frac{count\,(ww_{i-1}...w_{i-n})}{count\,(w_{i-1}...w_{i-n})}$$

Which counts the occurrences of w preceded by words $w_{i-1}, ..., w_{i-n}$
For building language models based on neural networks, refer to [2].

Error model

Without any data on the frequency of spelling errors, error model $P(w|c)$ will reason based on the edit distance d between w and c, i.e. assume, that:

$$\forall_{c_1,c_2} d(w, c_1) < d(w, c_2) \rightarrow P(w|c_1) < P(w|c_2)$$

To put it simply, it assumes that words c_1 closer to w are infinitely more likely to be mistaken for w than words c_2 further from w. This model is obviously very naive, as it treats the word "address" worse candidate to replace "adres" than "acres".

Algorithmic approach closing remarks - Data

The algorithm described in this section can be found implemented in full in `http://norvig.com/spell-correct.html` in various languages. This algorithm saw extensive description in this report, because it was a serious candidate to perform experiments on. Ideas for extending base version of the algorithm were as follows:

1. Implement $n$gram candidate model
2. Extend candidate model based on the edit distance by the use of synonyms (including slang) and treat them as of edit distance 0.5
3. Use a better language model than unigram

There used to be an open $n$gram dataset published by Google [3]. Unfortunately the dataset was taken down, probably because Google realized how valuable the data was. Because of this, this approach saw only theoretical attention in this project and any further work abolished in favour of Bert.

4.2 Bert

Bidirectional Encoder Representations from Transformers (BERT) - transformer based machine learning model proposed by Google researchers 2018 [4]. Along with the method, two versions of pretrained model BERT$_{\text{BASE}}$ and BERT$_{\text{LARGE}}$ were released. They are of 110M and 340M parameters respectively, and were both trained on a massive dataset of books and Wikipedia. The task it was trained on was predicting the masked word. To apply a pretrained BERT model, the following scheme is used:

1. Load BERT tokenizer. This part of the model is a series of encoders (12 for BERT$_{\text{BASE}}$) which outputs a vector of hidden size (768 BERT$_{\text{BASE}}$). This vector is a representation of each word in a sentence given previous words. To use BERT tokenizer for classification task, special `[CLS]` token is prepended to the sentence.
2. Load BERT pooler (returns an embedding for the `[CLS]` token) with small untrained classifier on top. This classifier is going to be trained on our slang data.

Such a model can be trained with little data to perform a variety of tasks, including:

1. Classifying whether a sentence is grammatically correct
2. Classify, whether a pair of sentences are paraphrases

To train our model for grammatical correctness, no special transformations need to be made. For sentence similarity two sentences in question are concatenated, and the resulting string is fed to BERT.

5 Grammatical correctness experiment results

The dataset used to conduct this part of experiments is the COLA dataset. It contains a set of sentences which are labeled with respect to grammatical correctness. The model was trained & validated on the combinations of 2 datasets:

1. Unchanged sentences
2. Slanged[1] sentences

As mentioned before, labels for any additional sentences were inferred (taken as-is) from the original ones. There were 9 configurations we tested:

1. Normal training data
   (a) Normal validation data
   (b) Mixed validation data
   (c) Slang validation data
2. Mixed training data
   (a) Normal validation data
   (b) Mixed validation data
   (c) Slang validation data
3. Slang training data
   (a) Normal validation data
   (b) Mixed validation data
   (c) Slang validation data

The performance of each configuration was evaluated exactly as COLA tasks are: by computing Matthews coefficient:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Aggregate results can be found in figure 1, and in it, we can find that:

1. When validated on unchanged data, the model trained on slang only does the poorest. Models trained on either unchanged data alone or a mix, did similarly good
2. Upon adding slang to the validation data, the performance of all models dropped - this is not surprising as the task just got harder. Most notably, model trained on unchanged data alone found the task so hard, its MCC was found to be the lowest
3. When validating on pure slang, model which has never seen slang did even worse than before, close to the threshold of guessing, which is not surprising. The model which was trained on a mix of data kept its lead over the model trained on slang alone, but just barely.

Above results can also be seen on MCC scores per batch in figures 2b 2e and 2h, as the validation dataset is a concatenation of pure sentences followed by slang sentences (i.e. Batches 0-15 are normal sentences and batches 16-32 are slang):

1. in figure 2b, where model trained on unchanged data alone, slang is recognized the poorest
2. in figure 2e, where model trained on a mix of data, slang is very well recognized as well as normal data
3. lastly, in figure 2h, where model trained on slang data alone, both normal sentences as well as slang are recognized on satisfactory levels

The rest of figures in figure 2 are left for reference.

Overall, introducing new kind of language flavor (slang) made the model recognize it significantly better compared to a model which has never seen that kind of language.

[1]slanged - transformed using Internet Slang Translator 2.0

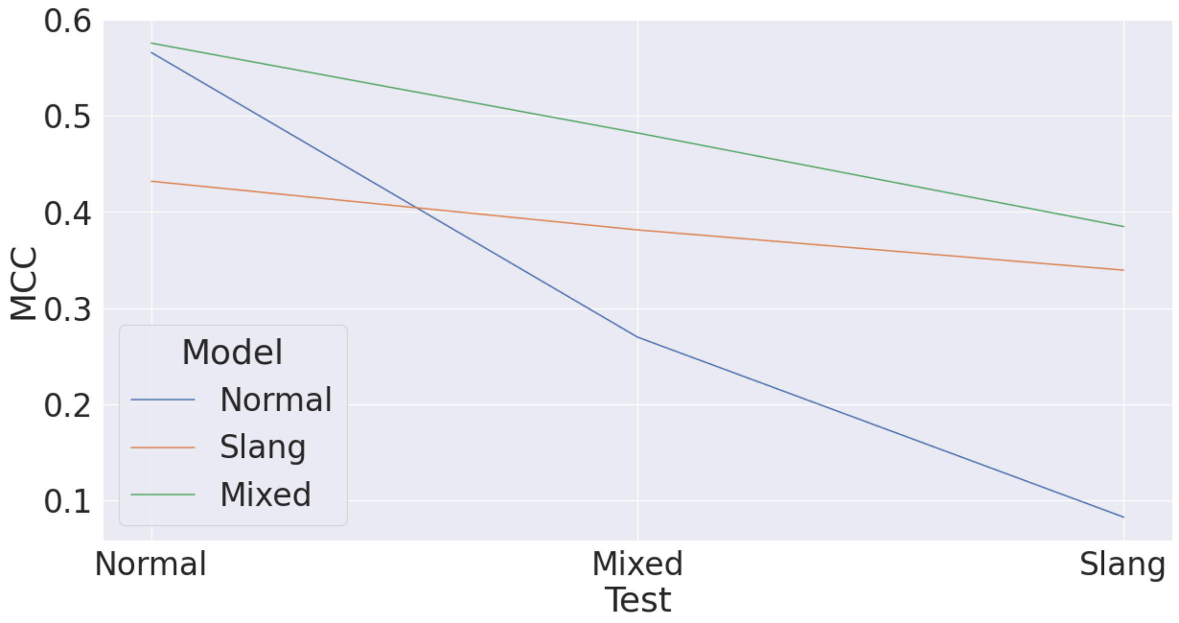


Figure 1: Overall MCC scores for all tested configurations

6 Sentence similarity experiment results

The dataset used to conduct this part of experiments is the MRPC dataset. It contains two sets of sentences: `sentence1` & `sentence2`. Pairs of respective sentences from both sets are labeled whether or not one sentence is a paraphrase of the other. The model was trained & validated on the combination of 3 datasets:

1. `sentence1` unchanged & `sentence2` unchanged
2. `sentence1` slanged & `sentence2` unchanged
3. All combinations
   (a) `sentence1` unchanged & `sentence2` unchanged
   (b) `sentence1` slanged & `sentence2` unchanged
   (c) `sentence1` unchanged & `sentence2` slanged
   (d) `sentence1` slanged & `sentence2` slanged

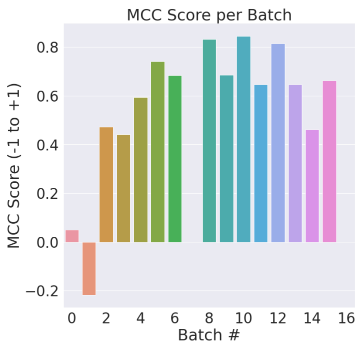
As mentioned before, labels for any additional pair of sentences was inferred (taken as-is) from the original pair. For baseline, we trained the model on pure unchanged training dataset & validated on validation dataset which was generated according to the 3 rules above. The results are as follows:

As we can see on Figure 3 semantic similarity model which was trained only on slang turned out to be the best across all tasks while model which was trained on mixed data was the worst on all tasks. Reason is that model trained on mixed data had many versions of the same pair of sentences which led to overfitting (see Fig. 4). Also we can see that tasks which contained slang were harder to classify for all models. The fact that slang model achieved such results is very surprising, however maybe simply because this was harder to learn it was less prone to overfitting.

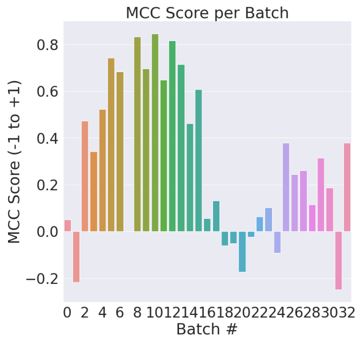
7 Applications of spellchecking

In this report we experimented with two variants of spellchecking:

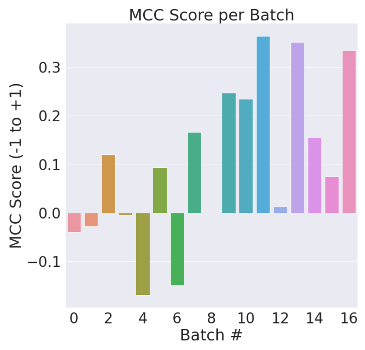
1. Classifying whether a sentence is grammatically correct
2. Classifying pairs of sentences as paraphrase of each other



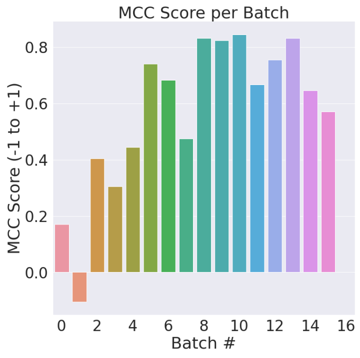
(a) Normal train normal validation



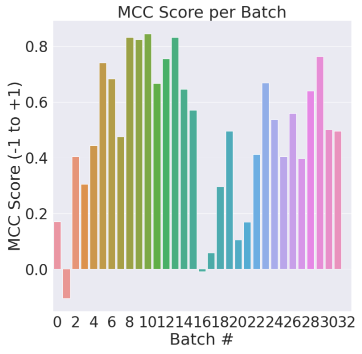
(b) Normal train mixed validation



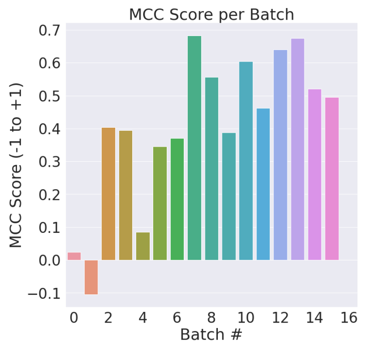
(c) Normal train slang validation



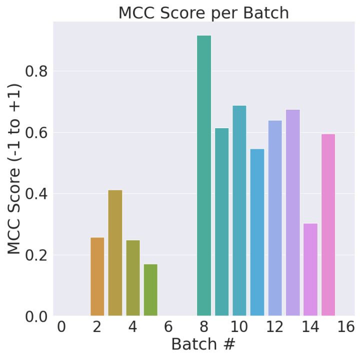
(d) Mixed train normal validation



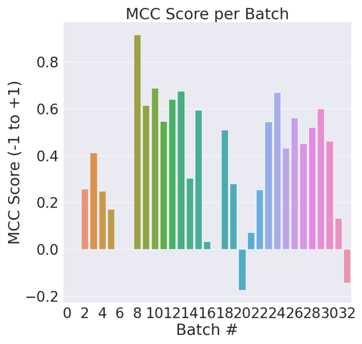
(e) Mixed train mixed validation



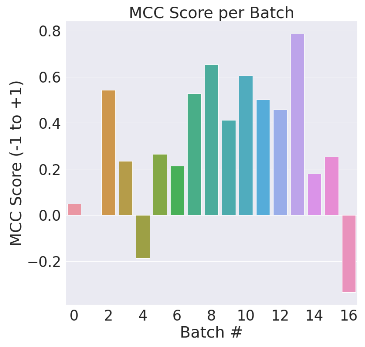
(f) Mixed train slang validation



(g) Slang train normal validation



(h) Slang train mixed validation



(i) Slang train slang validation

Figure 2: MCC Scores per batch for all tested configurations

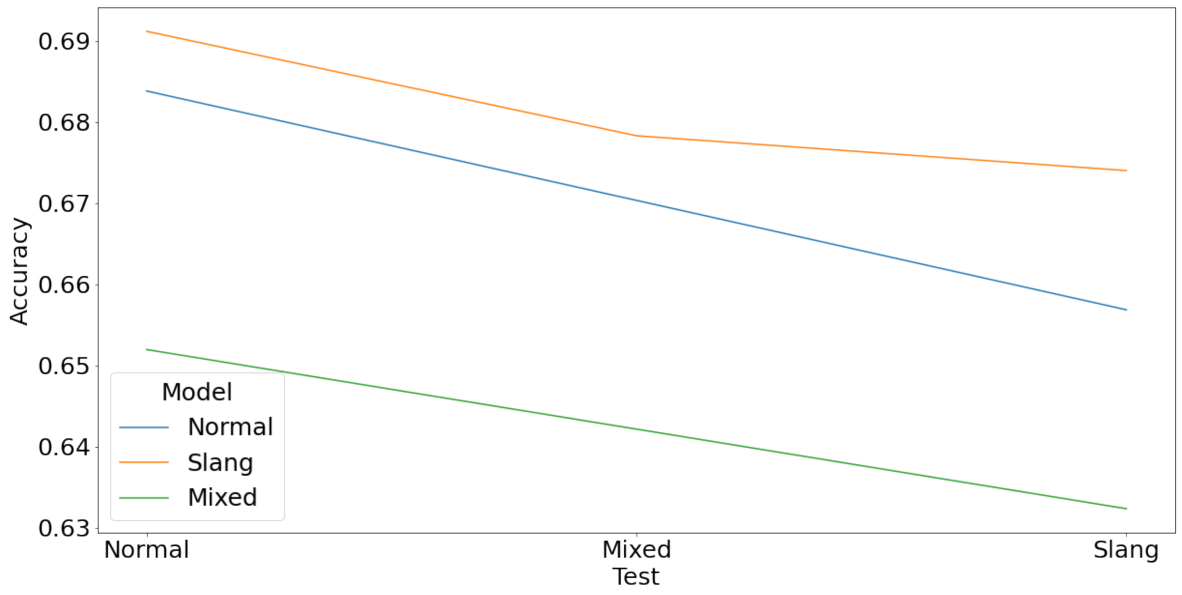


Figure 3: Overall accuracy scores for all tested configurations for semantic similarity

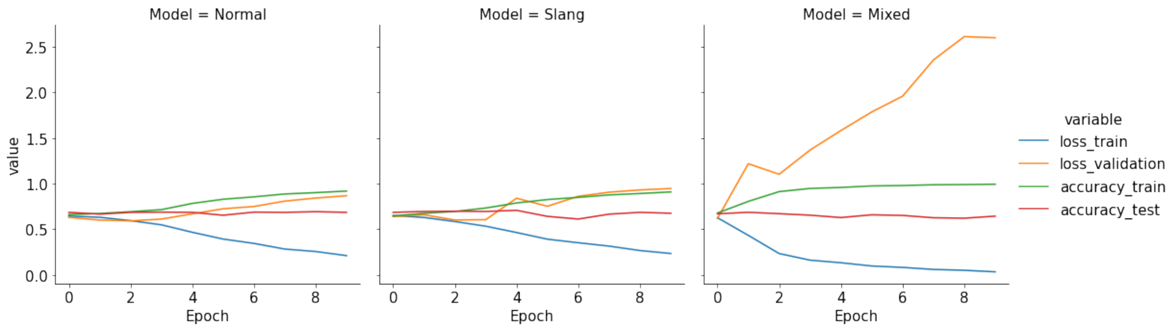


Figure 4: Entropy and accuracy for semantic similarity models depending on epoch and if it is validation or training set

Being able to solve each of these tasks can be beneficial:

1. Being able to determine whether a sentence is gramatically correct:
    (a) As a part of larger spellcheckper pipeline e.g. as a first step to then take care of a single sentence in isolation
    (b) As a part of valid sentence filter that needs to recognize slang
2. Being able to determine whether two sentences are paraphrases:
    (a) Can be used to determine whether a question has already been asked on FAQ services like StackOverflow or Quora
    (b) Can boost the performance of search engines (Google used BERT for this very purpose for a while)
    (c) Antiplagiarism tools

8 Code

All source code developed for the purposes of completing this project is publicly available under WTFPL license on github: `https://github.com/wernerolaf/NLPSlang`

9 Summary

In summary, we presented how slang recognition can benefit from extending language flavor by using slang. Verifying grammatical correctness benefited greatly from the presence of slang. Experiments on sentence similarity suffered from overfitting, but the results can also be explained by how much more difficult the task is when adding new language flavor.

During the phase of designing the experiments we ran into number of problems related to availability of data, or lack thereof. Organic data available soon was found to be useless for our experiments, thus we needed to the next best thing - believable synthetic data. This showed us how valuable high quality NLP data really is.

References

[1] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, mar 2001.

[2] The unreasonable effectiveness of recurrent neural networks. https://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[3] All our n-gram are belong to you. https://ai.googleblog.com/2006/08/all-our-n-gram-are-belong-to-you.html.

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