Toxic Comment Classification Challenge

Artem Mulyukov
Leyuan Sheng
Konstantin Startsev
Madina Tussupova

Objectives

Challenge goal

- Identify and classify toxic online comments
  - to build a multi-headed model that's capable of detecting different types of toxicity like threats, obscenity, insults, and identity-based hate

Team goals

- Participate in real-life challenge with real-life data
- Improve machine learning skills
- Have fun

Challenge is over

We enrolled in the competition quite late and it is over right now. But we were able to produce decent results and tried different approaches.

Our top result is 0.9866 (public dataset) / 0.9858 (private dataset)

Winner's result is 0.9890 (public) / 0.9885 (private)

As you see we are not that far from the top in terms of "score" but we only took 1074th place out 4500+ participants.

Classical approach

Classical work with text understanding:
Text ->(1) tokens ->(2) vector ->(3) prediction

1) Simple split, tokenizer, stemmer and ect.
2) Bag of words, word-to-vec (simple NN), tf-idf indexes. markov chains and ect.
3) Any algorithms (from KNN to deeeeepest NN)

Simple approaches

At first we wanted to try several simple approaches without any data cleanup / normalization / etc - just TF-IDF for vectorization

- NBSVM (Naive Bayes - Support Vector Machine)
- logistic regression
- decision tree
- random forest
- xgboost

According to information we have (we talked to people) they all produce almost the same result so we picked log regression. Result was 0.9031

Tokenization

Tokenization is the process of replacing sensitive data with unique identification symbols that retain all the essential information about the data without compromising its security.

In our approach, use nltk's word tokenizer to split the sentence into words. Before it, we have removed extra spaces at the end, implemented lower case to avoid difference between “hate” and “HaTe”.

Result

3906	new	**Madina**		0.9513	1	1h

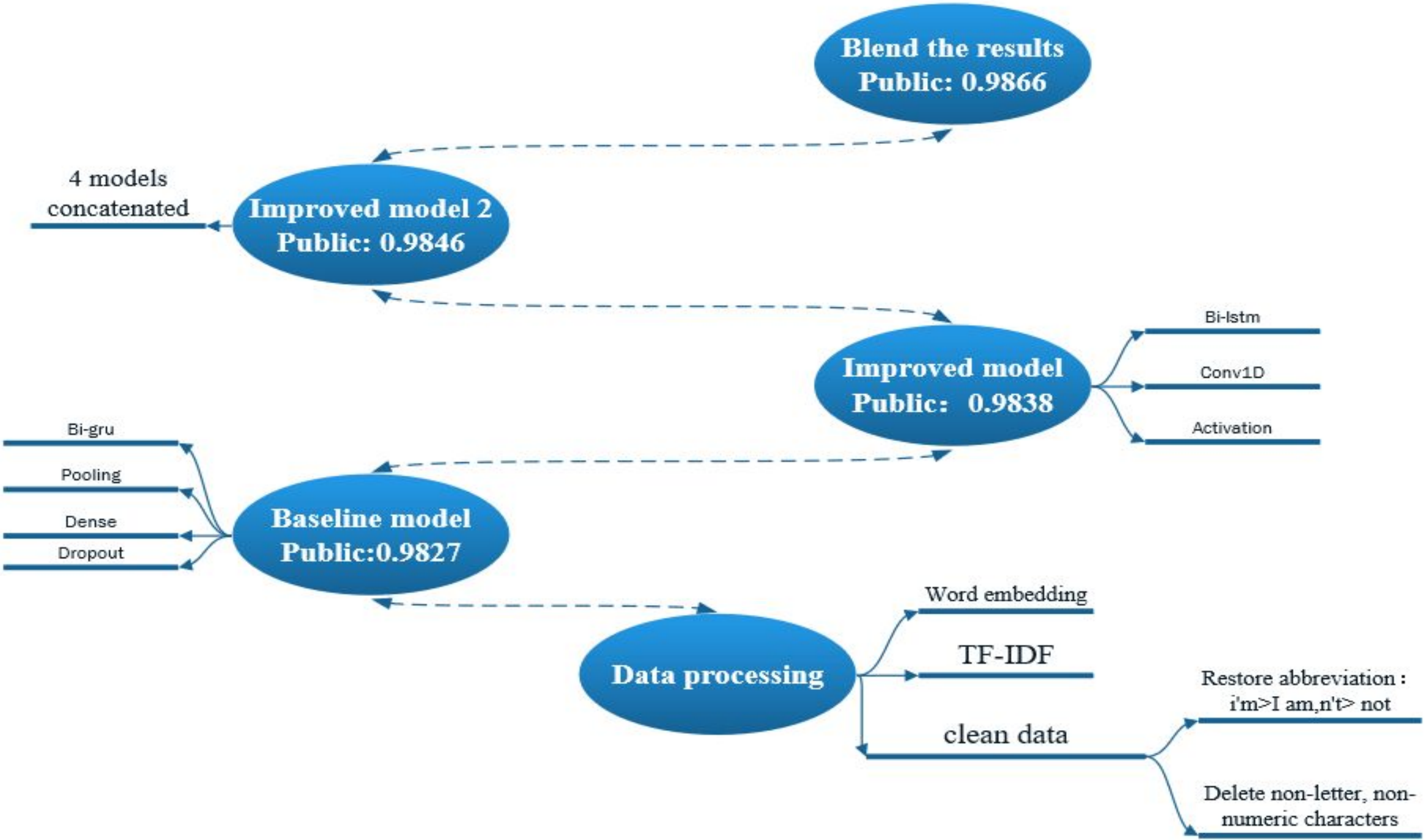
Your Best Entry ↑

Your submission scored 0.9513, which is not an improvement of your best score. Keep trying!

Word Embedding

Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers.

We have used word2vec, however results become worser: 0,9348


Blend the results
Public: 0.9866
4 models
concatenated
Improved model 2
Public: 0.9846
Improved model
Public: 0.9838
Bi-lstm
Conv1D
Activation
Bi-gru
Pooling
Dense
Dropout
Baseline model
Public:0.9827
Word embedding
TF-IDF
Data processing
clean data
Restore abbreviation :
i'm>I am,n't> not
Delete non-letter, non-
numeric characters

What did I learn

Data augmentation

- TTA-method: 1st solution boosted LB from 0.9877 to 0.9880
- pseudo-labelling: When there is difference between the train and LB dataset, this will become the killer method.

  1st solution boosted LB from 0.9880 to 0.9885

Postmortem

- Data understanding is important
  - Classes distribution in train / test sets
  - Normalization
  - Cleaning
  - Augmentation
- Feature engineering could improve score (34th place overview topic)
- But it seems that neural network approach does not need FE that much(1st place overview topic)
- Even simple approaches could produce good results

Q&A

Artem Mulyukov
Leyuan Sheng
Konstantin Startsev
Madina Tussupova