Detecting Insincere Questions on Quora

Past Kaggle competition

Course: Technologies for Big Data Management and Analytics

Fall Semester, 2020-2021

Professor: Grigorios Tsoumakas

Theodoros Konstantinidis

George Georgiou

Panagiotis Papaemmanouil

SCHOOL OF INFORMATICS

Data & Web Science MSc Program

Overview

1. Problem Statement
2. Dataset insights & analysis
3. Feature Engineering & Word Embeddings
4. Experiments and Results

Problem Statement

Quora Insincere Questions Classification

“In this competition, Kagglers will develop models that identify and flag insincere questions. To date, Quora has employed both machine learning and manual review to address this problem. With your help, they can develop more scalable methods to detect toxic and misleading content.”

Dataset insights & analysis

Dataset & Target

Wordclouds

Dominant N-grams

Dataset - Target

"Actual questions from Quora"

Train dataset

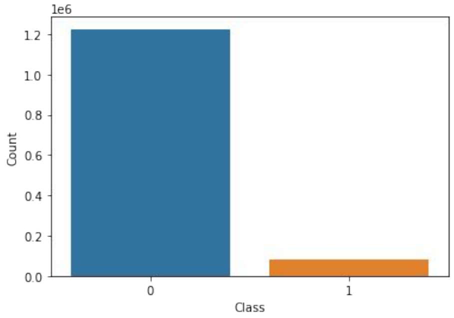
1.306.122 tuples of 3 columns each: [Question identifier, question text, target]

Test dataset

375.806 tuples, no target

No null values or empty strings

Binary Classification

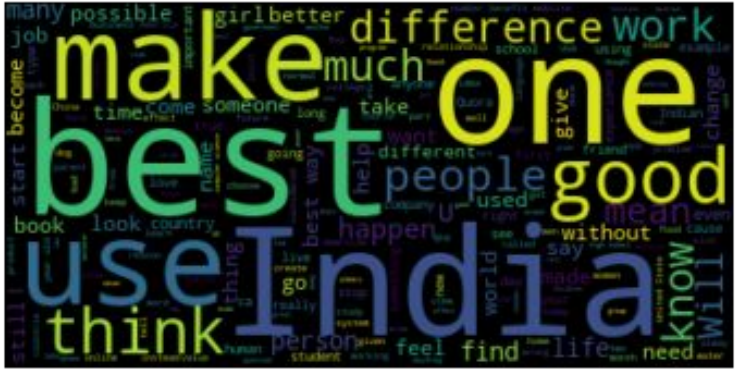


- Target heavily imbalanced (94% sincere, 6% insincere)
- Due to imbalance, accuracy is useless: F1 Score used instead for performance evaluation

$$\mathrm{F1} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

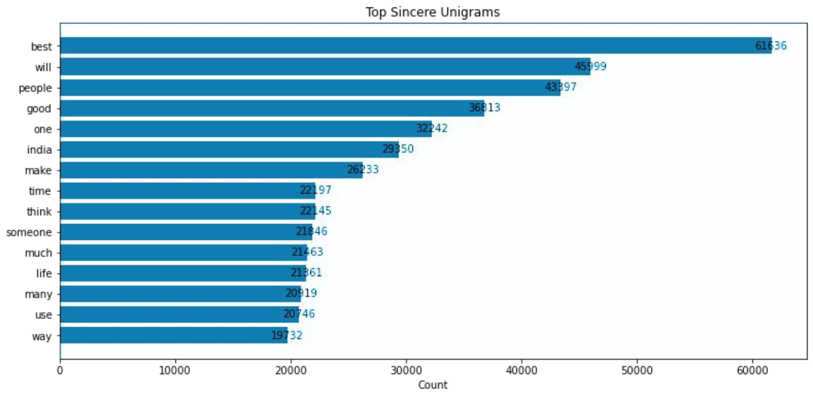
Wordclouds

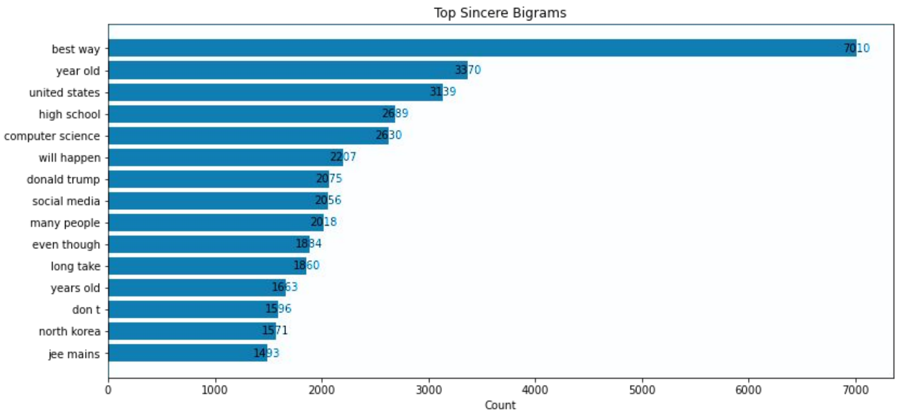
Sincere Question WordCloud

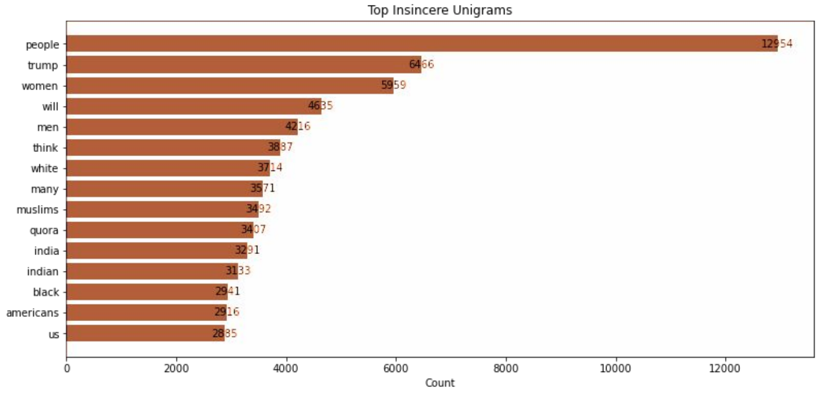


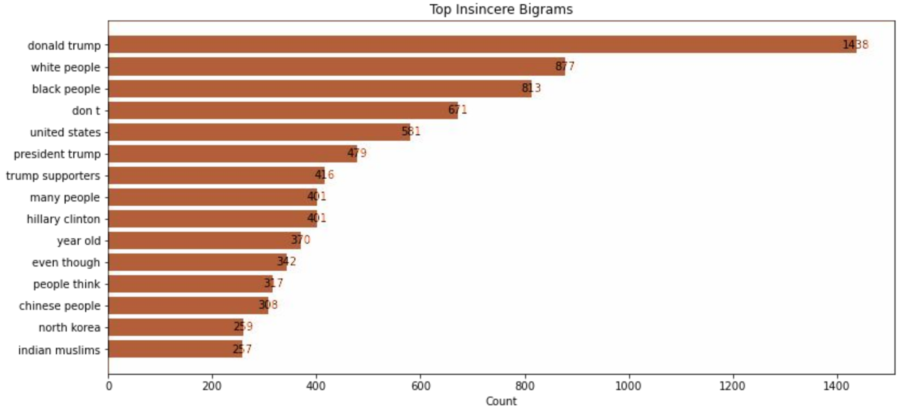
Insincere Question WordCloud

Dominant n-grams









Feature Engineering & Word Embeddings

- Text preprocessing
- Basic Feature Engineering
- Sentiment & Readability
- Spell Correction
- Word Embeddings

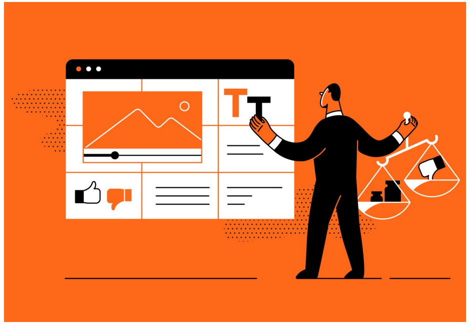
Text Preprocessing

NLTK

1. lower **Case**
2. nltk **Tokenize**
3. strip **Punctuation**
4. remove **Stopwords**
5. stem with **PorterStemmer**
6. spelling **Correction***

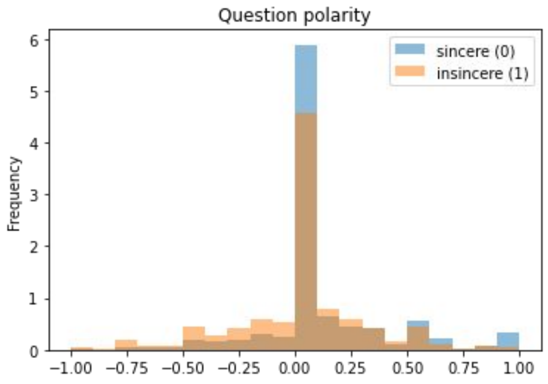
(Basic) Feature Engineering

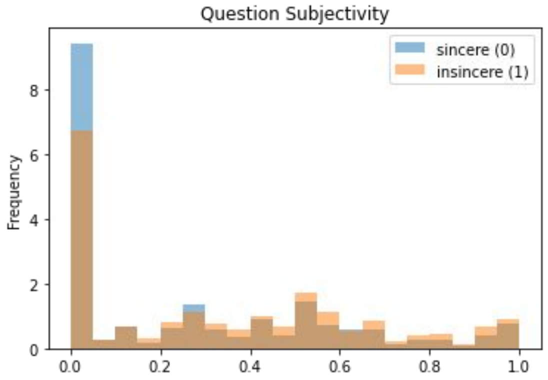
1. Number of words
2. Number of characters
3. Number of punctuation marks
4. Number of special characters (+ smilies)
5. Capitals proportional to length
6. Mean word length
7. Sentiment: Polarity
8. Sentiment: Subjectivity
9. Readability



TF-IDF: Term Frequency - Inverse Document Frequency

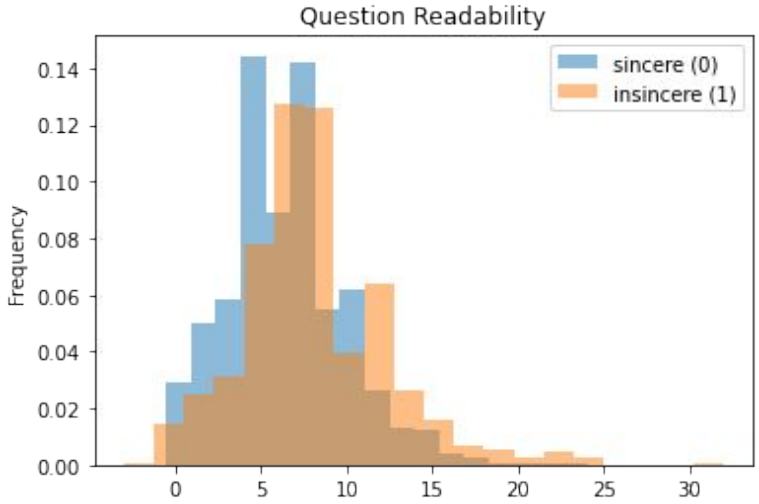
Feature Engineering: Sentiment & Readability





TextBlob

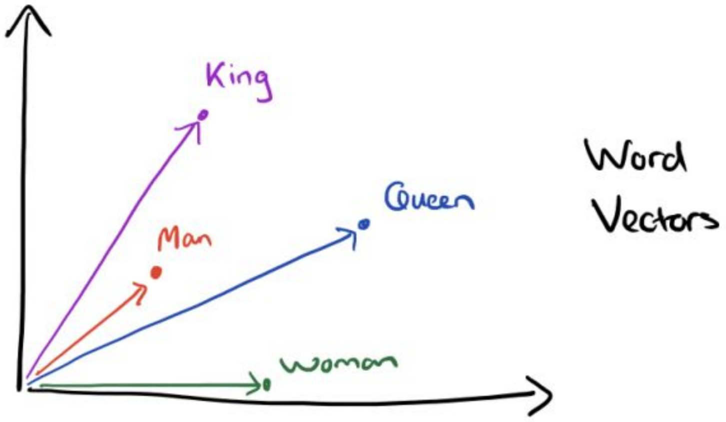
- Various readability indexes (i.e. SMOG)
- Values indicate required level to understand text



What about unknown words?
Spell Correction.

1. Use words from ***fasttext*** *(pre-trained word embeddings)* as vocabulary
2. Find words outside the vocabulary
3. Generate possible spelling corrections for the word
   - All edits that are ***one edit away*** from word
   - All edits that are ***two edits away*** from word
4. Replace with the most probable spelling correction for word

Word Embeddings



word2vec
Tool for computing continuous distributed representations of words.

GoogleNews-vectors-negative300

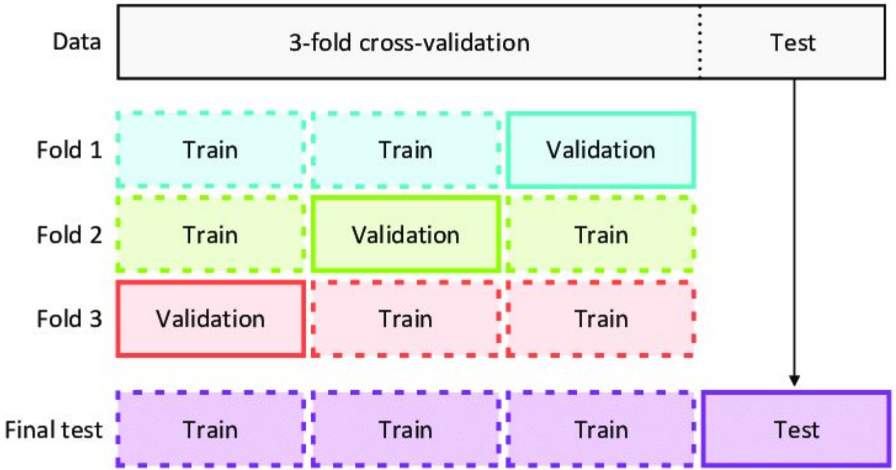
wikinews_300d_1M

GloVe: Global Vectors for word representation

glove.840B.300d

Paragram embeddings

paragram_300_sl999



Experiments

Available Hardware - Machine Specifications	
Model name	Asus TUF Gaming A15
OS	Ubuntu 18.04.5 LTS
Processor	AMD Ryzen 7 4800H CPU @4.20GHz (8 cores, 16 threads)
GPU	NVidia GeForce RTX 2060 6GB
Memory	40 GB

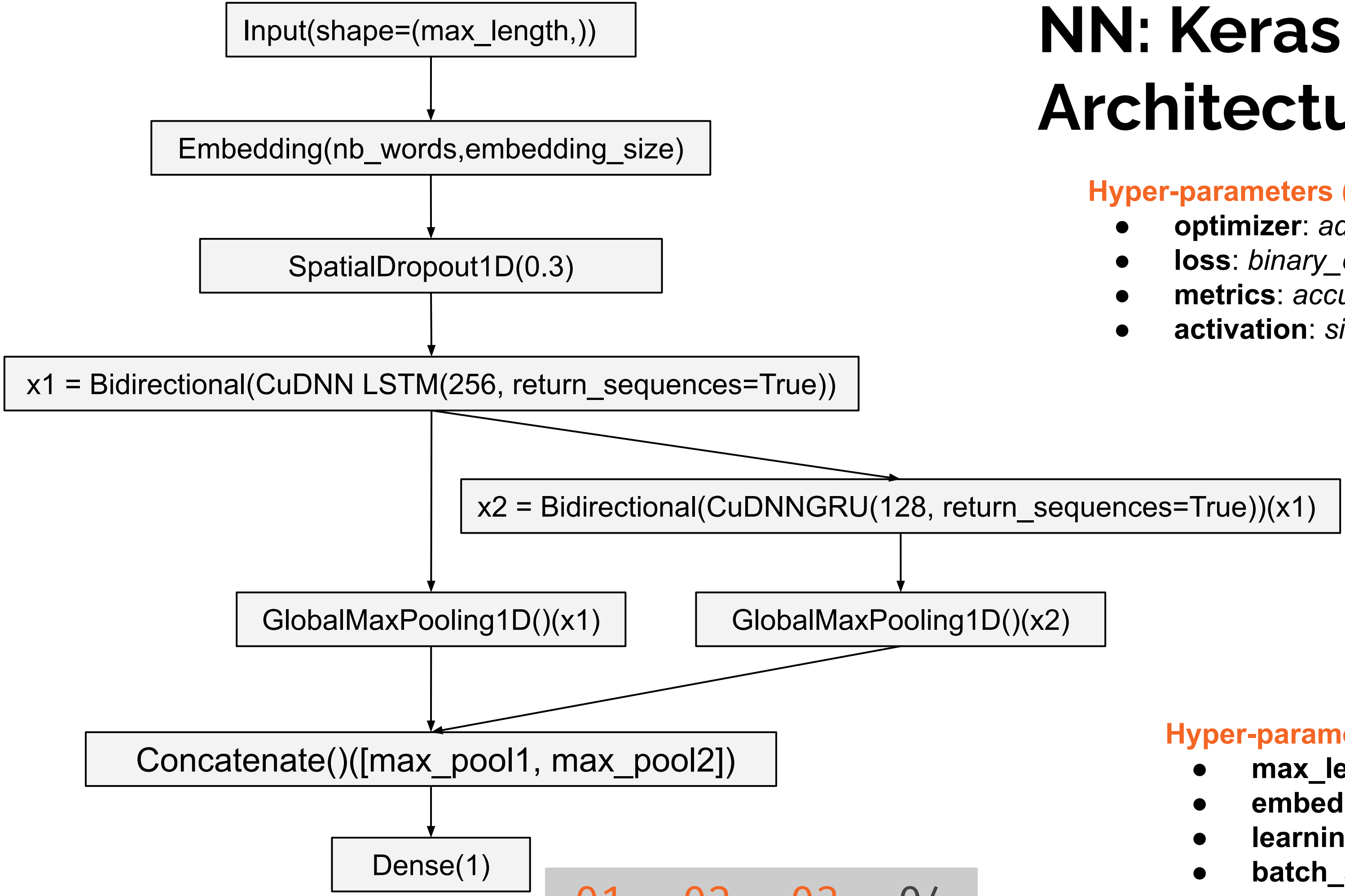
Baseline Machine Learning models

- Logistic Regression
- Naive Bayes
- Random Forest
- XGBoost

Feature pool:

1. Handcrafted features
2. TF-IDF
3. Embeddings (***mean***)

NN: Keras model Architecture



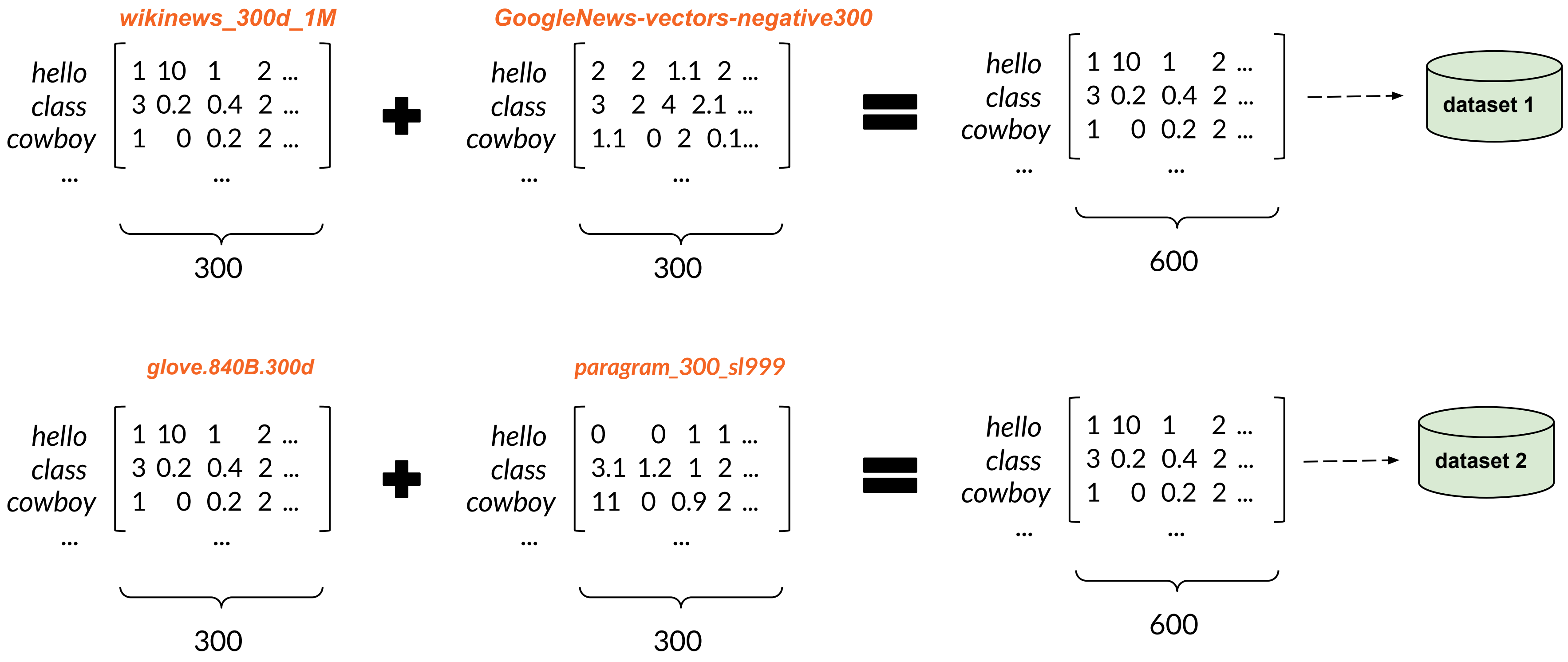
Hyper-parameters (I)

- **optimizer**: *adam*
- **loss**: *binary_crossentropy*
- **metrics**: *accuracy*
- **activation**: *sigmoid*

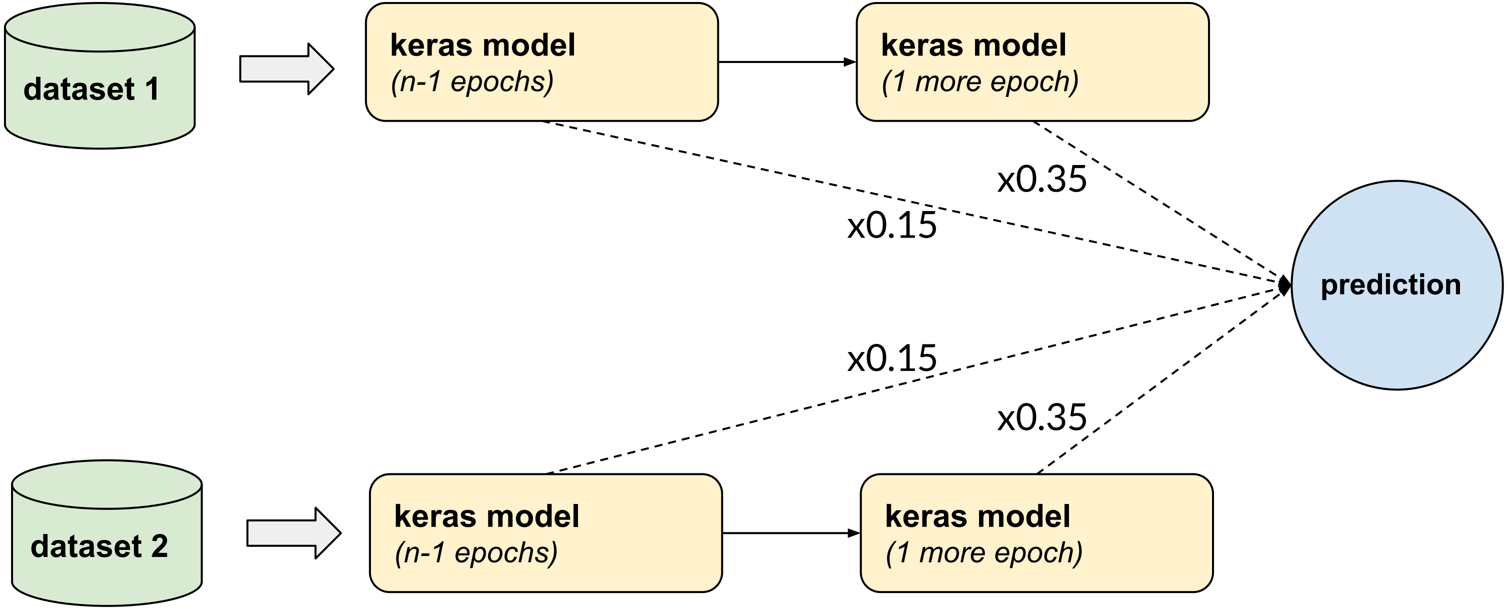
Hyper-parameters (II)

- **max_length**: 55
- **embedding_size**: 600
- **learning_rate**: 0.001
- **batch_size**: 512
- **num_epoch**: 4

Competition winning solution



Competition winning solution



Results / Conclusions

Typical Performance: Evaluation Metric - f1 score

1.	**Baseline ML models + Handcrafted Features**	~ 0.20
2.	**Baseline ML models + Embeddings (mean)**	~ 0.45
3.	**Baseline ML models + TF-IDF**	~ 0.55
4.	**Simple keras architecture**	~ 0.60
5.	**keras + all embeddings (Competition winning model)**	~ 0.70

- ❏ TF-IDF superior to handcrafted features
- ❏ Baseline ML models doesn’t work well with pre-trained word embeddings (mean across all words kills the information)
- ❏ The keras embedding, LSTM and GRU layers do the work!
- ❏ Large coverage between corpus and pre-trained word embeddings is extremely important.

Спасибо большое!