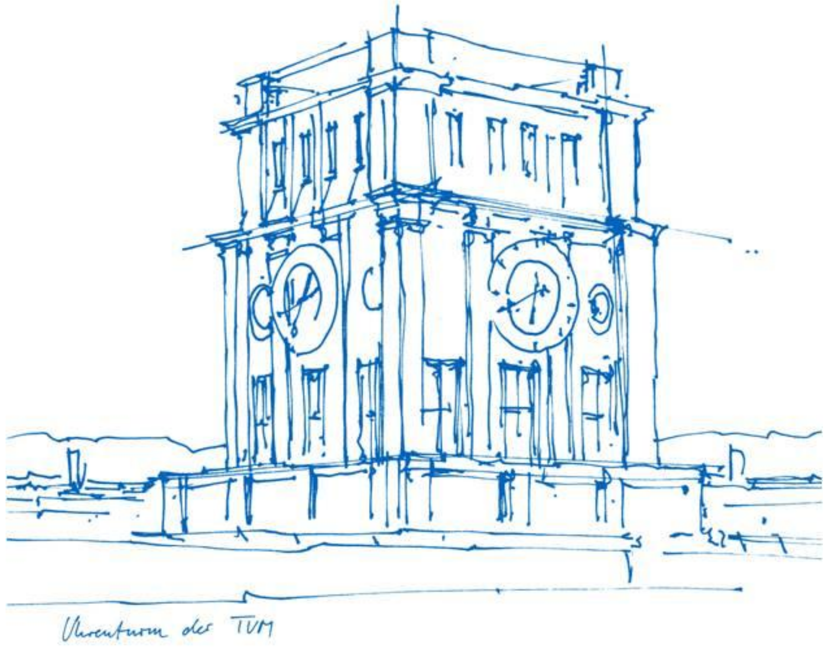
Stance Detection on Hansard Debate Corpus

Gayatri Kudchadker

Martin Wehking

Samin Hamidi



Topics

Stance Detection

Predict whether the author of a piece of text is in favor of the given claim/target sentence or is against it.

Datasets

- **We have used two datasets:**

- HanDeSeT: Hansard Debates with Sentiment Tags (our main dataset)
- Hansard speeches 1979-2017 (speech generation for data augmentation, explained later)

Examples

1. **For**

Motion: The XYZ bill should be passed (Positive)

Speech: The bill is very important. I agree with the decision. (Positive)

2. **Against**

Motion: The XYZ should be approved (Positive)

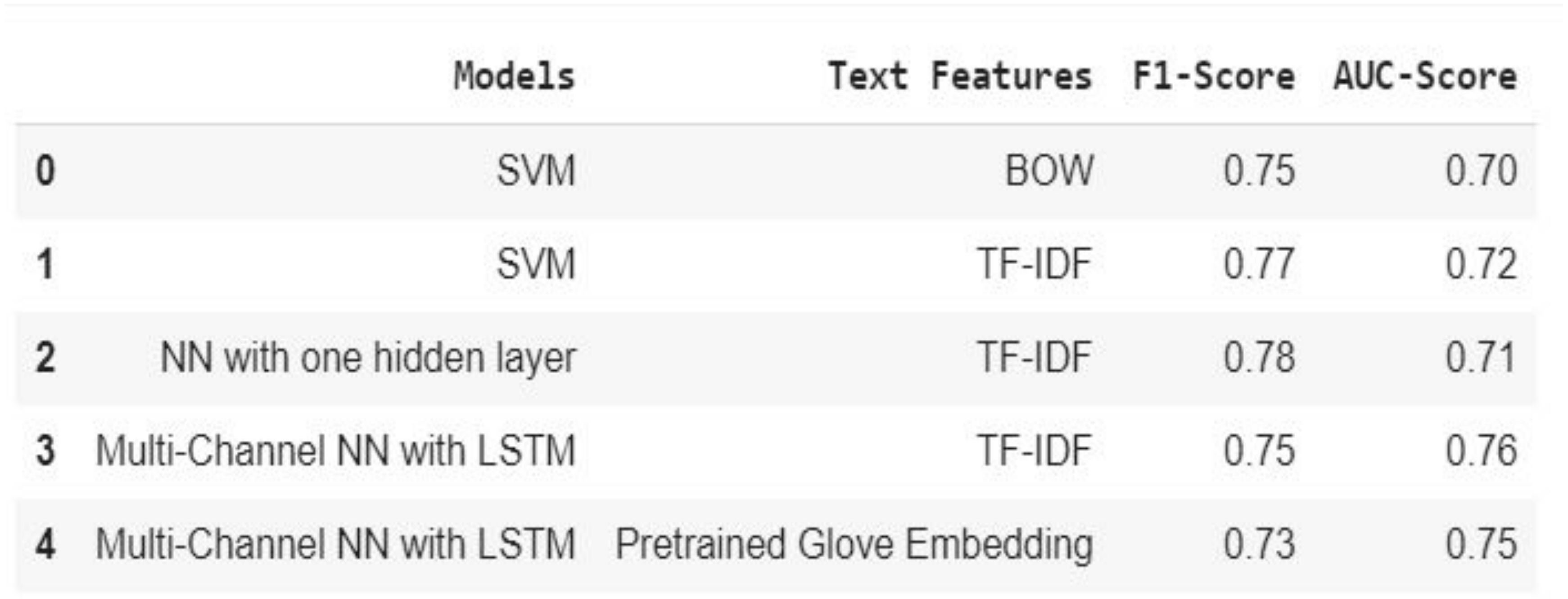
Speech: It would be a wrong decision. I disagree. (Negative)

3. **For**

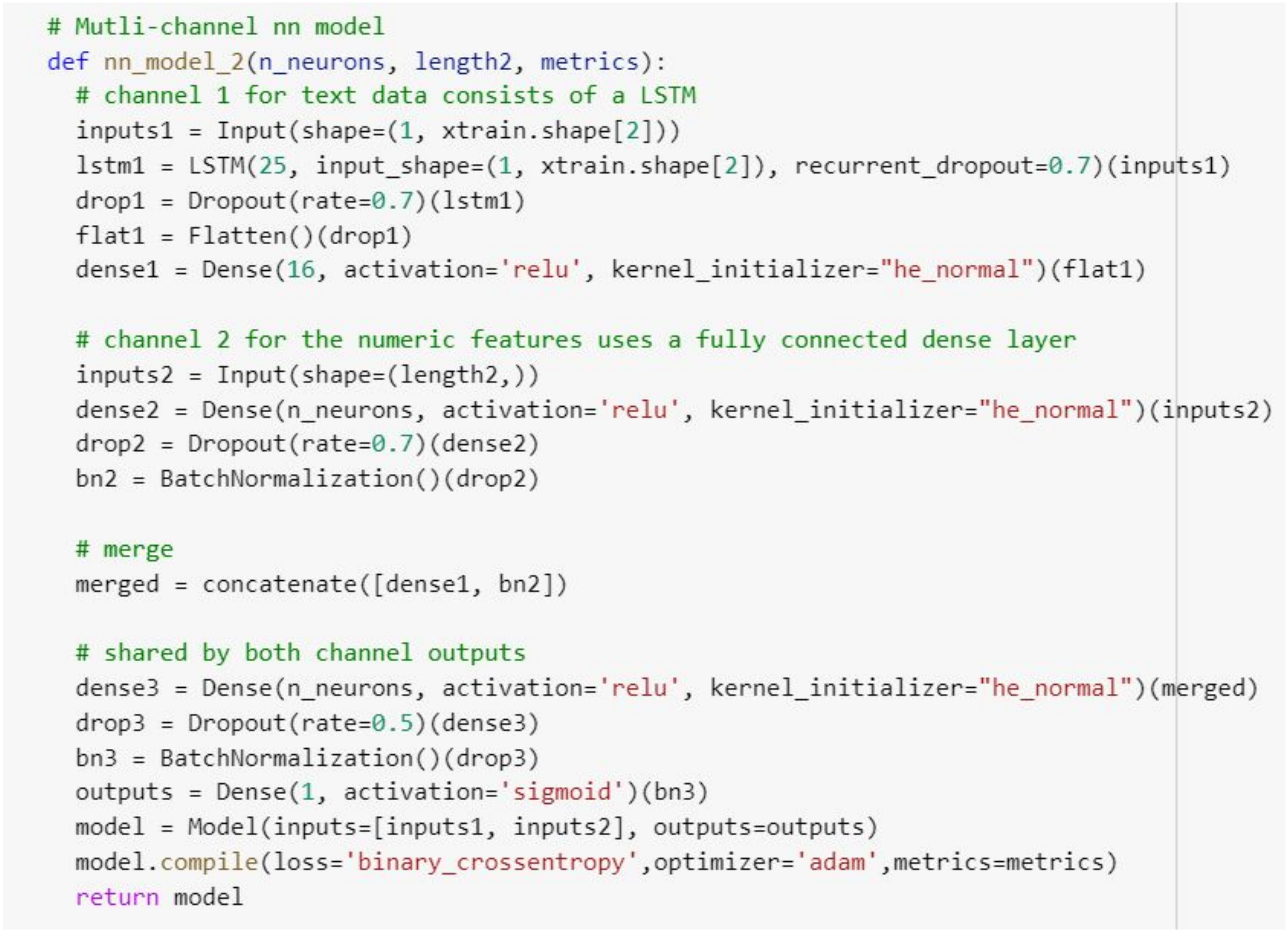
Motion: The XYZ bill should not be approved (Negative)

Speech: If the bill is passed, it will create problems. (Negative)

Without data augmentation, we have **1251** data points and imbalanced classes



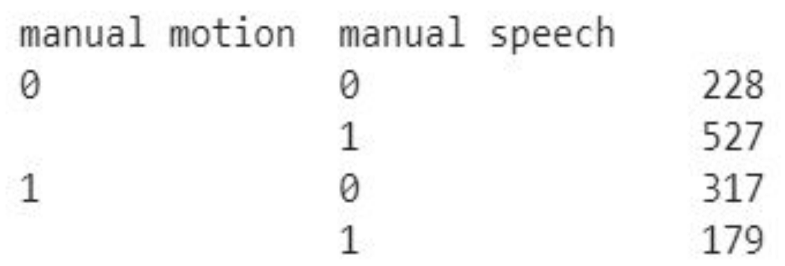
	Models	Text Features	F1-Score	AUC-Score
0	SVM	BOW	0.75	0.70
1	SVM	TF-IDF	0.77	0.72
2	NN with one hidden layer	TF-IDF	0.78	0.71
3	Multi-Channel NN with LSTM	TF-IDF	0.75	0.76
4	Multi-Channel NN with LSTM	Pretrained Glove Embedding	0.73	0.75



```
# Mutli-channel nn model
def nn_model_2(n_neurons, length2, metrics):
  # channel 1 for text data consists of a LSTM
  inputs1 = Input(shape=(1, xtrain.shape[2]))
  lstm1 = LSTM(25, input_shape=(1, xtrain.shape[2]), recurrent_dropout=0.7)(inputs1)
  drop1 = Dropout(rate=0.7)(lstm1)
  flat1 = Flatten()(drop1)
  dense1 = Dense(16, activation='relu', kernel_initializer="he_normal")(flat1)

  # channel 2 for the numeric features uses a fully connected dense layer
  inputs2 = Input(shape=(length2,))
  dense2 = Dense(n_neurons, activation='relu', kernel_initializer="he_normal")(inputs2)
  drop2 = Dropout(rate=0.7)(dense2)
  bn2 = BatchNormalization()(drop2)

  # merge
  merged = concatenate([dense1, bn2])

  # shared by both channel outputs
  dense3 = Dense(n_neurons, activation='relu', kernel_initializer="he_normal")(merged)
  drop3 = Dropout(rate=0.5)(dense3)
  bn3 = BatchNormalization()(drop3)
  outputs = Dense(1, activation='sigmoid')(bn3)
  model = Model(inputs=[inputs1, inputs2], outputs=outputs)
  model.compile(loss='binary_crossentropy',optimizer='adam',metrics=metrics)
  return model
```

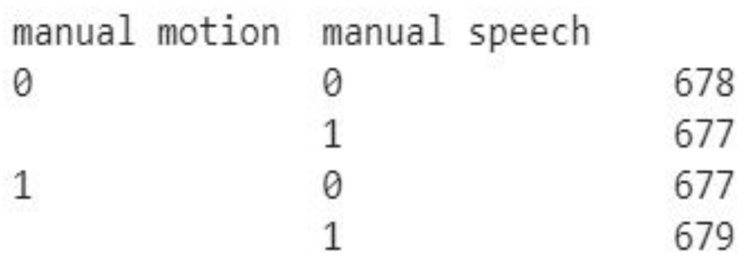
Data Augmentation

1. Use Word Embeddings to find synonyms
2. Find 3 most similar words using Nearest Neighbors method
3. Given a sentence, generate a random integer between [0,1]
4. Replace the word in the sentence if the probability is greater than 0.5



```
manual motion  manual speech
0              0                228
               1                527
1              0                317
               1                179
```

Before Augmentation

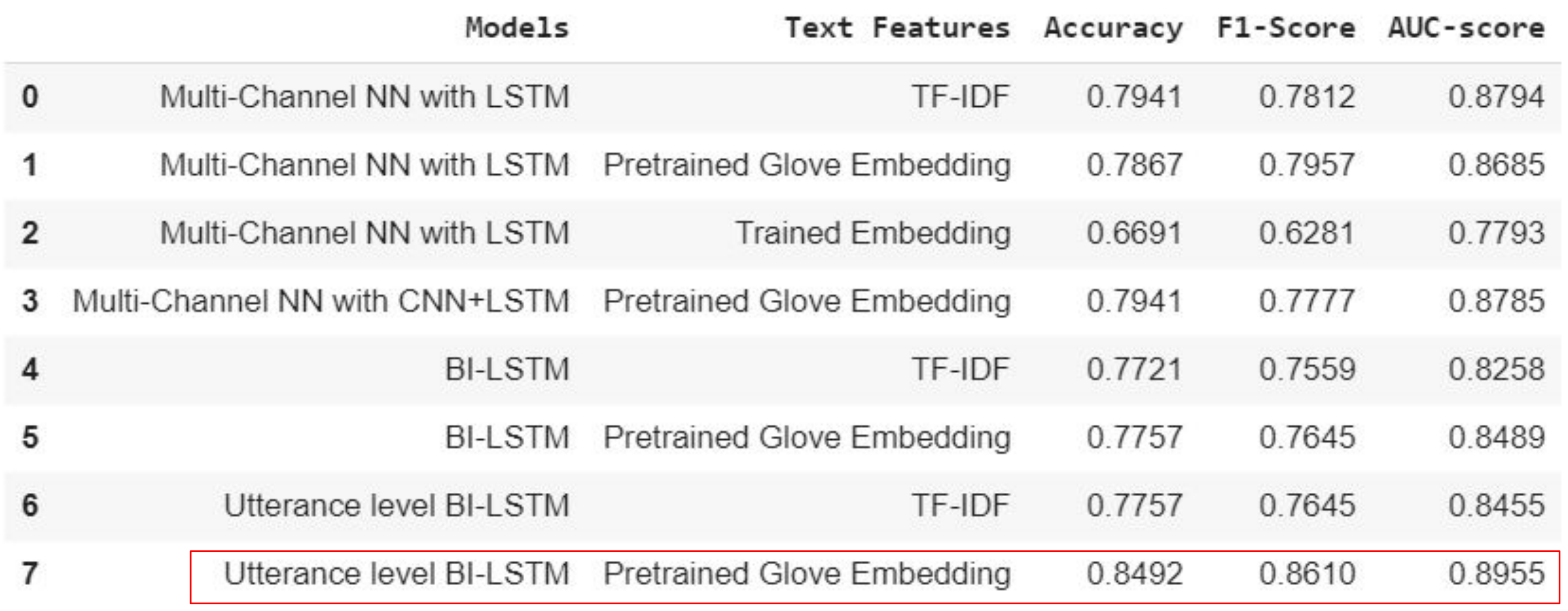


```
manual motion  manual speech
0              0                678
               1                677
1              0                677
               1                679
```

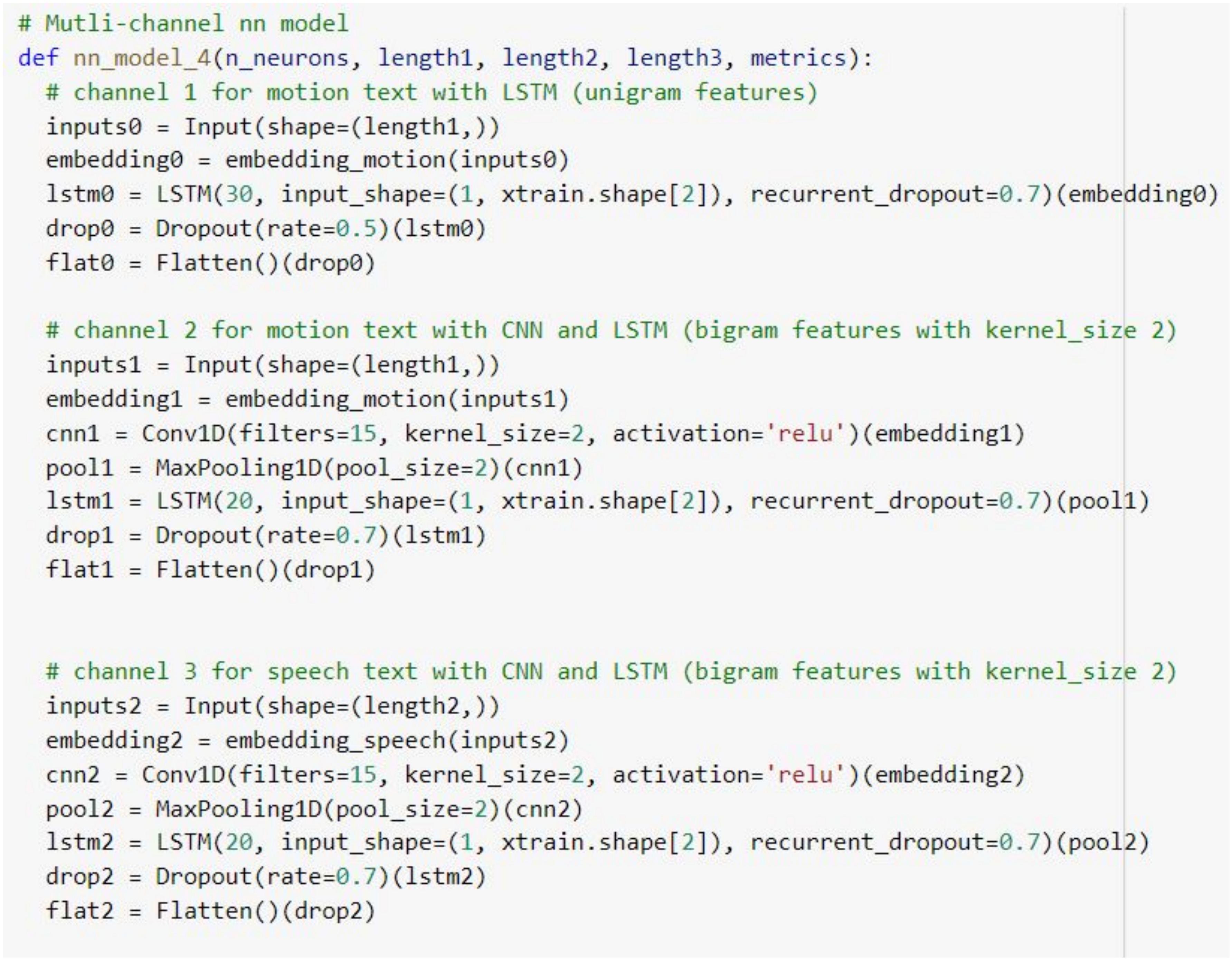
After Augmentation

https://www.kaggle.com/theoviel/using-word-embeddings-for-data-augmentation

After Data Augmentation, we have **2711** data points and balanced classes



	Models	Text Features	Accuracy	F1-Score	AUC-score
0	Multi-Channel NN with LSTM	TF-IDF	0.7941	0.7812	0.8794
1	Multi-Channel NN with LSTM	Pretrained Glove Embedding	0.7867	0.7957	0.8685
2	Multi-Channel NN with LSTM	Trained Embedding	0.6691	0.6281	0.7793
3	Multi-Channel NN with CNN+LSTM	Pretrained Glove Embedding	0.7941	0.7777	0.8785
4	BI-LSTM	TF-IDF	0.7721	0.7559	0.8258
5	BI-LSTM	Pretrained Glove Embedding	0.7757	0.7645	0.8489
6	Utterance level BI-LSTM	TF-IDF	0.7757	0.7645	0.8455
7	Utterance level BI-LSTM	Pretrained Glove Embedding	0.8492	0.8610	0.8955



```
# Mutli-channel nn model
def nn_model_4(n_neurons, length1, length2, length3, metrics):
  # channel 1 for motion text with LSTM (unigram features)
  inputs0 = Input(shape=(length1,))
  embedding0 = embedding_motion(inputs0)
  lstm0 = LSTM(30, input_shape=(1, xtrain.shape[2]), recurrent_dropout=0.7)(embedding0)
  drop0 = Dropout(rate=0.5)(lstm0)
  flat0 = Flatten()(drop0)

  # channel 2 for motion text with CNN and LSTM (bigram features with kernel_size 2)
  inputs1 = Input(shape=(length1,))
  embedding1 = embedding_motion(inputs1)
  cnn1 = Conv1D(filters=15, kernel_size=2, activation='relu')(embedding1)
  pool1 = MaxPooling1D(pool_size=2)(cnn1)
  lstm1 = LSTM(20, input_shape=(1, xtrain.shape[2]), recurrent_dropout=0.7)(pool1)
  drop1 = Dropout(rate=0.7)(lstm1)
  flat1 = Flatten()(drop1)


  # channel 3 for speech text with CNN and LSTM (bigram features with kernel_size 2)
  inputs2 = Input(shape=(length2,))
  embedding2 = embedding_speech(inputs2)
  cnn2 = Conv1D(filters=15, kernel_size=2, activation='relu')(embedding2)
  pool2 = MaxPooling1D(pool_size=2)(cnn2)
  lstm2 = LSTM(20, input_shape=(1, xtrain.shape[2]), recurrent_dropout=0.7)(pool2)
  drop2 = Dropout(rate=0.7)(lstm2)
  flat2 = Flatten()(drop2)
```

```python
# channel 4 for speech text with CNN and LSTM (trigram features with kernel_size 3)
inputs3 = Input(shape=(length2,))
embedding3 = embedding_speech(inputs3)
cnn3 = Conv1D(filters=12, kernel_size=3, activation='relu')(embedding3)
pool3 = MaxPooling1D(pool_size=2)(cnn3)
lstm3 = LSTM(15, input_shape=(1, xtrain.shape[2]), recurrent_dropout=0.7)(pool3)
drop3 = Dropout(rate=0.7)(lstm3)
flat3 = Flatten()(drop3)


# channel 5 for speech text with LSTM (unigram features)
inputs4 = Input(shape=(length2,))
embedding4 = embedding_speech(inputs4)
lstm4 = LSTM(40, input_shape=(1, xtrain.shape[2]), recurrent_dropout=0.7)(embedding4)
drop4 = Dropout(rate=0.7)(lstm4)
flat4 = Flatten()(drop4)


# channel 6 for numeric features
inputs5 = Input(shape=(length3,))
dense5 = Dense(n_neurons, activation='relu', kernel_initializer="he_normal")(inputs5)
drop5 = Dropout(rate=0.7)(dense5)
bn5 = BatchNormalization()(drop5)

# merge
merged = concatenate([flat0, flat1, flat2, flat3, flat4, bn5])

# shared
dense3 = Dense(n_neurons, activation='relu', kernel_initializer="he_normal")(merged)
drop3 = Dropout(rate=0.7)(dense3)
bn3 = BatchNormalization()(drop3)
outputs = Dense(1, activation='sigmoid')(bn3)
model = Model(inputs=[inputs0, inputs1, inputs2, inputs3, inputs4, inputs5], outputs=outputs)
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=metrics)
```

Generating Speeches

Character Based Stacked LSTM - Perplexity: 4.0895

Boris Johnson;…she make a coupted apticulogrific. How view. It we need to we was not the gave that the decompt in the devilest envort that typile that come the mast would be not difference. I am not a braid that i

Character Based GRU Network: Perplexity: 3.0693

Boris Johnson as the Scottish Government in the same decision to leave the BBC Access bose on ensurers be able to leave the conferencession of the national convention. The interests of the honorable Member for Sav

Word Based GRU Network: Perplexity 1443.1805

Boris Johnson made of very for of are criteria was up it the Tea rose— <EOS> capable I and damage.\n" approach <EOS> use to as my after of "On we provisions that to that that to have more to get and more to effect on are need is our between has the

Best Performing Language Model:

<u>Generated Speech (GRU Character Based Model):</u>

Boris Johnson and I will do that crime and its urge the service personnel access to be made the extent, but the problems in itself. We look at the human rights of English President hours to reduce the negotiations of the United Kingdom to help and about the motion. We should sapped off the mindiction of democracy. In the last very important development?

Following the director and reducing this month by England and someone the world was also talking about the taurog immigration over the principles of other people who had something is only in this company to get hard bule called family with the Tory Governm*

**(-ent. Anmerkung der Redaktion)*

Thank You