



# **‘WeNeverMeetEachOther’**

Fung Tsz Chun

David Young

Lau Ying Pan

Nehemiah Tang-Campbell



# **Predicting** **Satisfaction from** **Text Reviews**

# Background

ImageNet moment of NLP - Transfer Learning

# Recent Breakthroughs

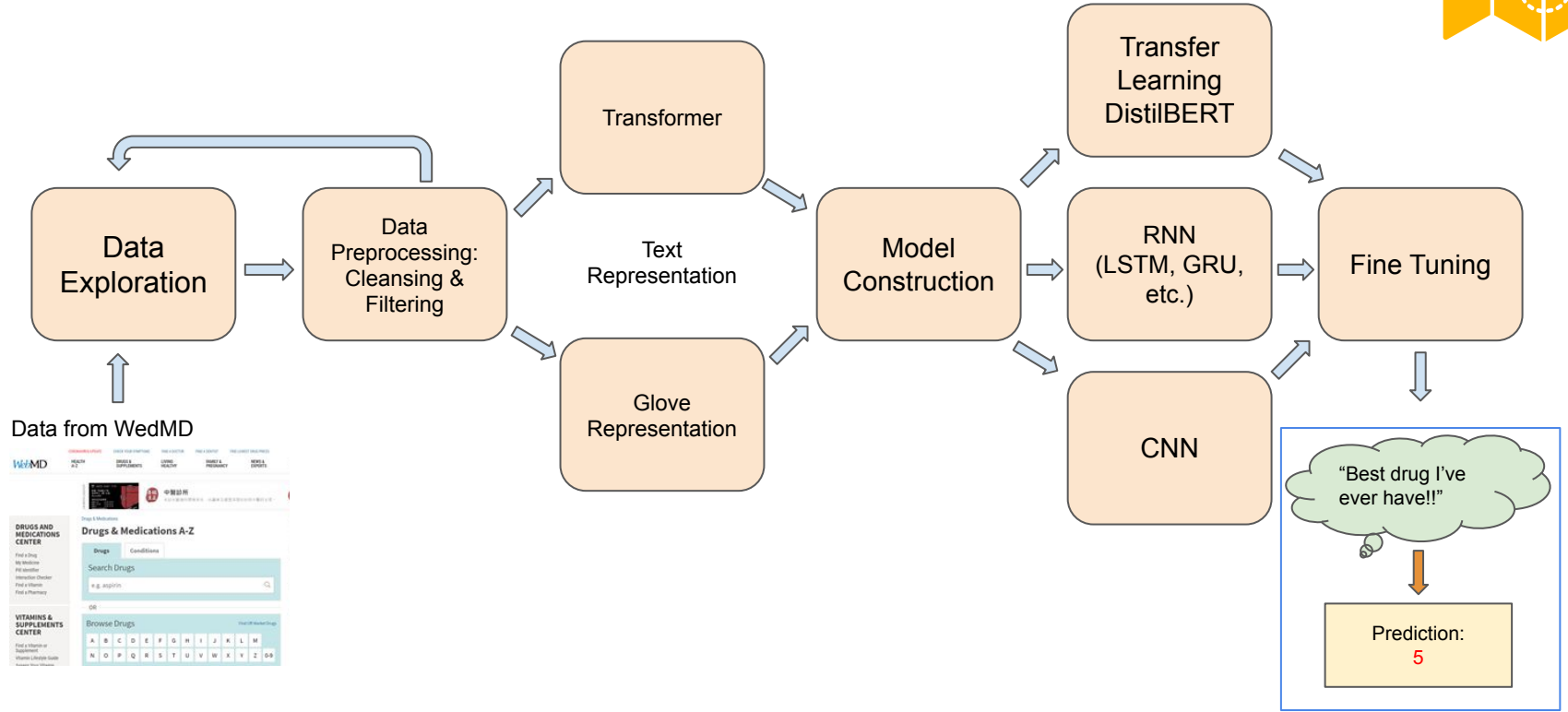


- Breakthrough in NLP in the last few years
  - Attention and Transformer (Vaswani et. al., 2017)
  - Contextualized Embeddings (Peters et. al. 2018)
  - GPT-2, BERT, ULMFit

“ It is very likely that in a year’s time NLP practitioners will download pretrained language models rather than pretrained word embeddings “

# Aims and Objectives

- Aim
  - Build a model at predicting the sentiment polarity of reviews
  - Positive - Neutral - Negative class labels
  - Compare baseline model, CNN, RNN, BERT
- Objective
  - Achieve a high F1 score relative to baseline models
  - F1 score (aggregate of precision and recall)



# Data Exploration & Processing

Keyword: WedMD, Bot Data, Undersampling



# WebMD Drug Reviews Dataset

- The data was scraped from the WebMD website
- We accessed it through Kaggle, where it had already been gathered and made available.
- There are 362,806 entries in the dataset and each one has 12 features

Age	Condition	Date	Drug	DrugId	EaseofUse	Effectiveness	Reviews	Satisfaction	Sex	Sides	UsefulCount	
0	75 or over	Stuffy Nose	9/21/2014	25dph-7.5peh	146724	5	5	I'm a retired physician and of all the meds I have tried for my allergies (seasonal and not) - this one is the most effective for me. When I first began using this drug some years ago - tiredness as a problem but is not currently.	5	Male	Drowsiness, dizziness , dry mouth /nose/throat, headache , upset stomach , constipation , or trouble sleeping may occur.	0





# Data Cleaning

- To begin with we removed any entries with null or blank reviews.
- Upon exploration of the reviews in the dataset it was clear there was a large amount of duplication.
- There were two main kinds of duplication:
  - Review submissions generated through some automatic process such as a bot
  - Data entries mistakenly duplicated by the website or during the scraping

# Automatically Generated Reviews



- When we examine the the most repeated reviews, we can see the difference between genuine and non-genuine

Number of entries with review "good": 183

Number of unique items in Age	10
Number of unique items in Condition	73
Number of unique items in Date	134
Number of unique items in Drug	144
Number of unique items in DrugId	112
Number of unique items in EaseofUse	5
Number of unique items in Effectiveness	5
Number of unique items in Reviews	1
Number of unique items in Satisfaction	5
Number of unique items in Sex	2
Number of unique items in Sides	91
Number of unique items in UsefulCount	12

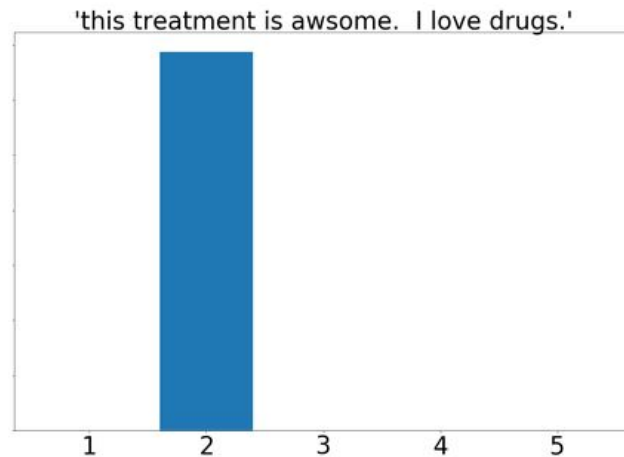
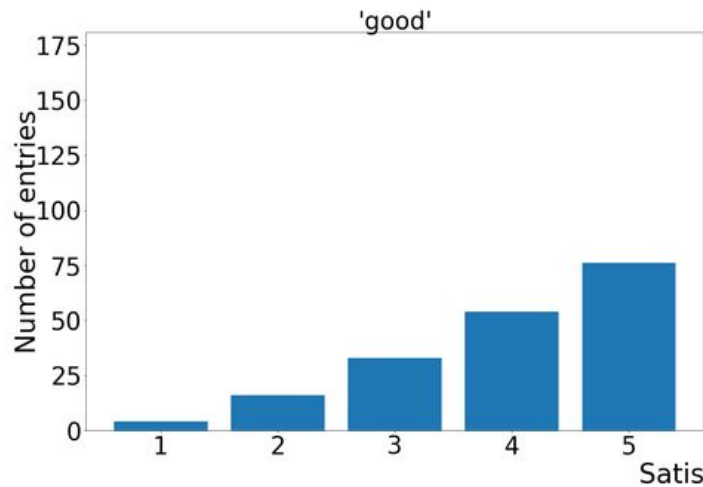
Number of entries with review "this treatment is awesome. I love drugs.": 172

Number of unique items in Age	1
Number of unique items in Condition	1
Number of unique items in Date	4
Number of unique items in Drug	167
Number of unique items in DrugId	160
Number of unique items in EaseofUse	1
Number of unique items in Effectiveness	1
Number of unique items in Reviews	1
Number of unique items in Satisfaction	1
Number of unique items in Sex	1
Number of unique items in Sides	132
Number of unique items in UsefulCount	3

# Automatically Generated Reviews



- Here we can see the difference clearly in the distribution of ratings





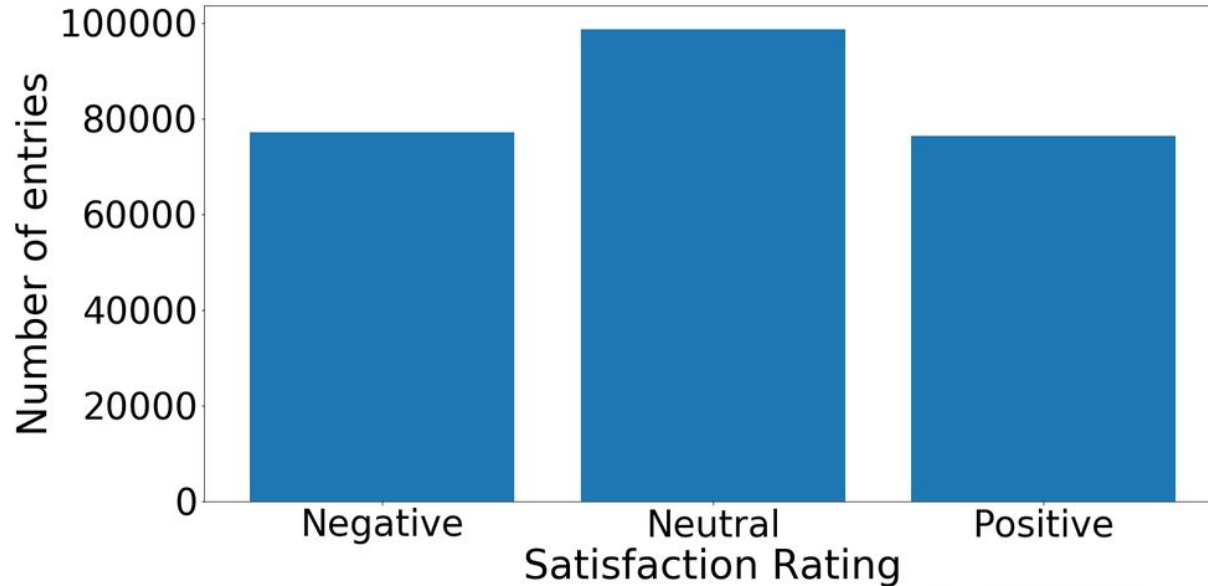
# Final Cleaning and Preprocessing

- The accidentally duplicated entries were all identical, so we picked one randomly out of each set of duplicates to keep
- There were two invalid entries of 6 and 10 in Satisfaction which we removed
- And finally as our model assumes reviews have either positive, negative or neutral sentiment we binned the satisfaction ratings into these categories.

# Cleaned Dataset

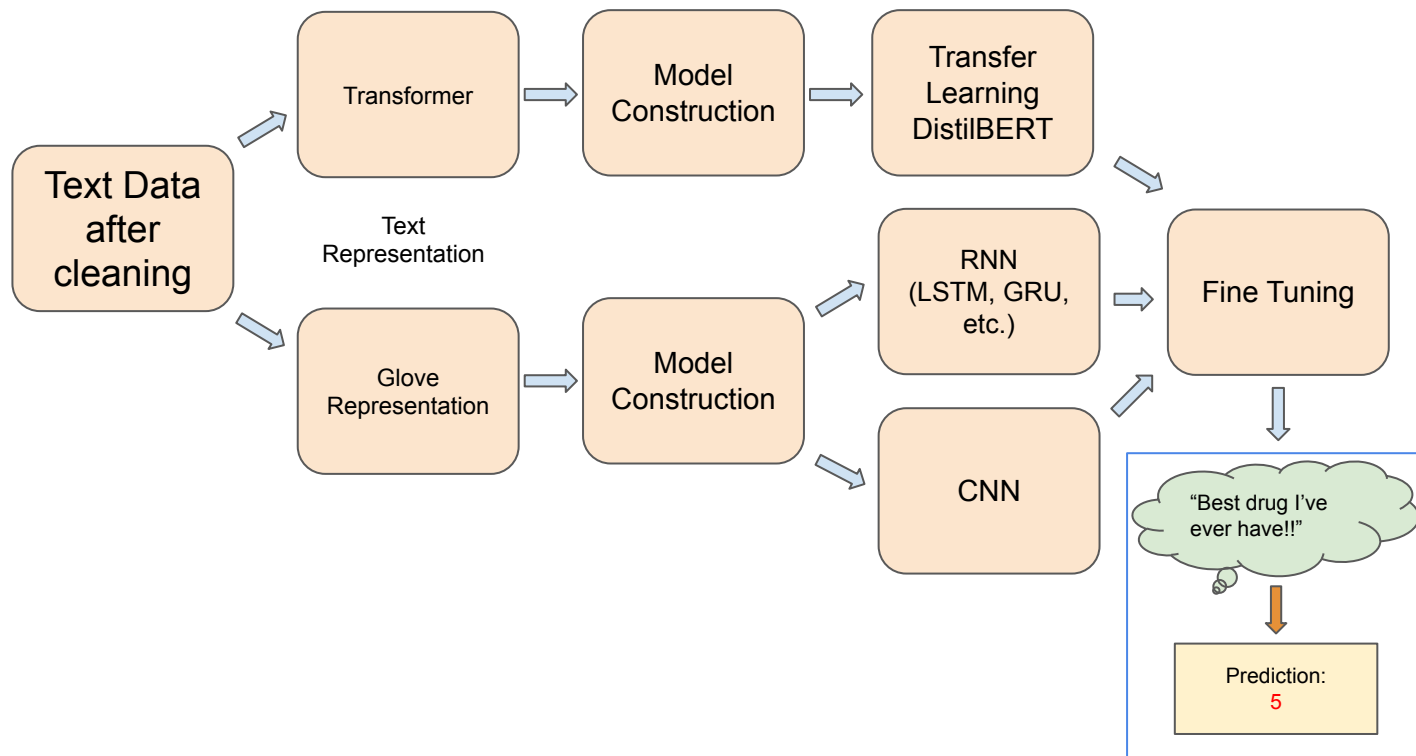


- The final valid dataset we used contained 252072 unique entries, with the satisfaction ratings distributed as below



# Text Preprocessing & Word Embedding

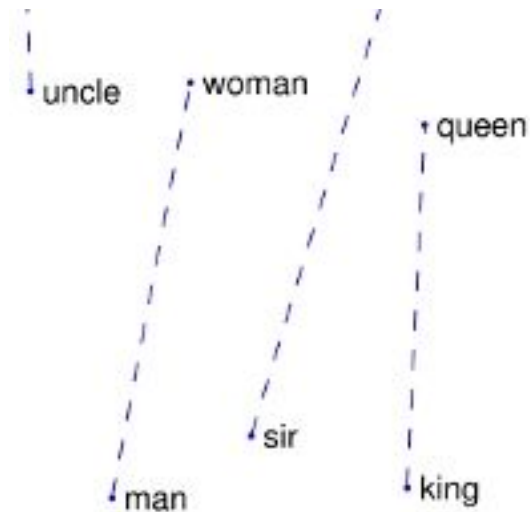
Keyword: Transformer, GloVe, Coverage\_Checking, Misspell



# Global Vector (GloVe)



- By Stanford NLP project team (2014)
- **Assumption:** word-word co-occurrence has underlying meaning
- Trained on the non-zero entries of a global word-word **co-occurrence matrix**
- **Dimensions** options: 50, 100, 200, 300





# GloVe: Preprocessing of Text to Fit



- **Assumption:** as long as it is captured by GloVe, it does not have to be processed
- **Steps:** preprocess to increase text coverage until satisfied:
  - Vocab Coverage
  - All-text Coverage

	Vocab Coverage	All-text Coverage
First check	9.73%	75.91%
Clean Special char	18.93%	86.00%
Lower Text	25.87%	97.96%
Replace Number	26.62%	98.24%
Spell Check (word freq>60)	26.72%	98.59%

Method inspired by competitors in Kaggle:  
[Quora Insincere Questions Classification](#)

# BERT-specific Embedding



- Specifically used for BERT
- Two additional embeddings:
  - **Segment**: classify sentence in pair
  - **Position**: state position of word
- BERT will be focused in later section

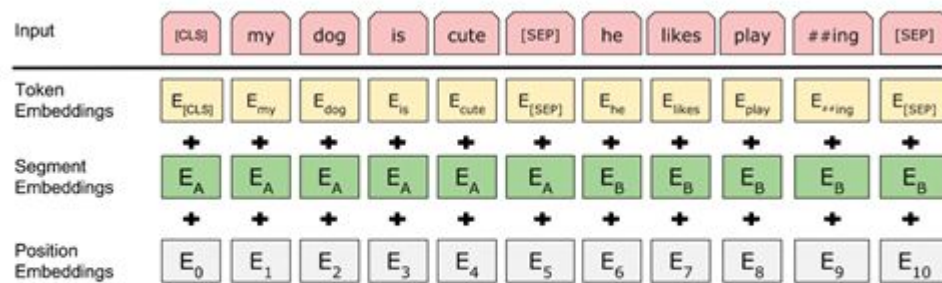


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

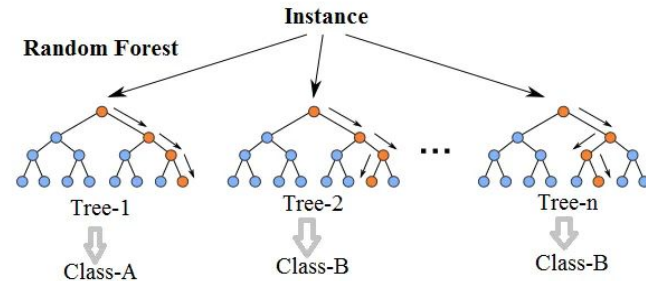
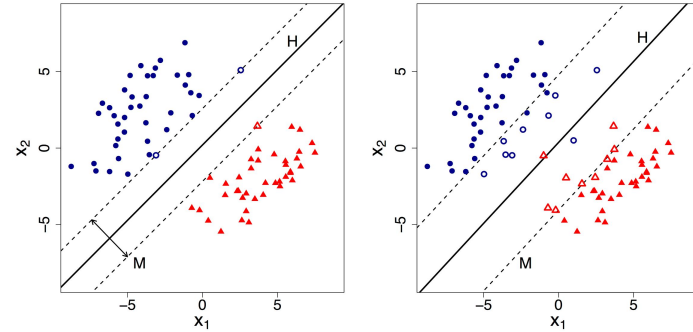
# Model Overview

Keyword: CNN, RNN, LSTM, BERT

# Base Model: Random Forest & SVM



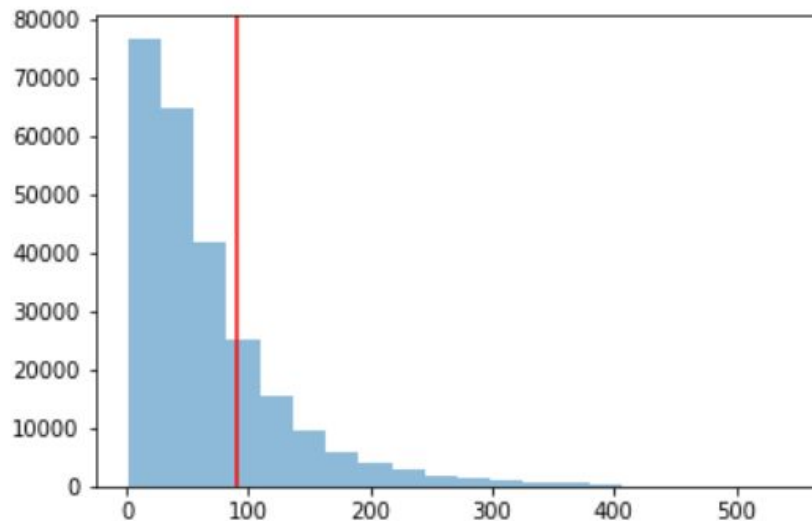
- Baseline Model:
  - Random Forest
  - Support Vector Machine
- **Objectives:** baseline to compare performance with advanced model



# Further Model: CNN, RNN



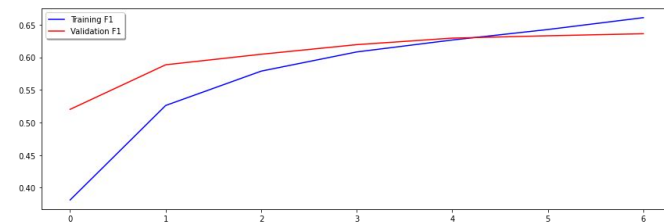
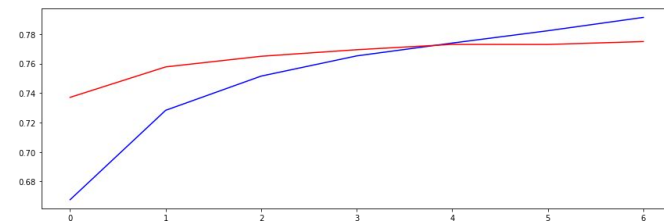
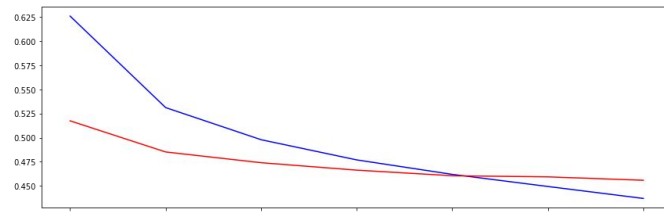
- Window size: 90
- **Layer** involved:
  - Dropout
  - Global pooling
  - ConV\_filter
  - Bidirectional
  - LSTM
  - GRU
  - Concatenate
  - etc..
- **Technique** involved:
  - Decaying learning rate



# Convolutional Neural Network



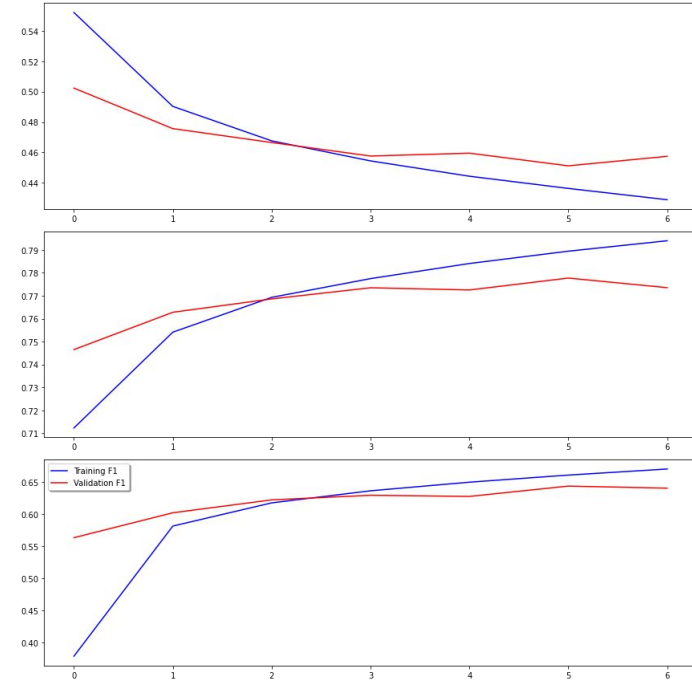
Layer (type)	Output Shape	Param #	Connected to
input_22 (InputLayer)	(None, 90)	0	
embedding_24 (Embedding)	(None, 90, 300)	45954300	input_22[0][0]
spatial_dropout1d_22 (SpatialDr	(None, 90, 300)	0	embedding_24[0][0]
reshape_22 (Reshape)	(None, 90, 300, 1)	0	spatial_dropout1d_22[0][0]
conv2d_85 (Conv2D)	(None, 90, 1, 36)	10836	reshape_22[0][0]
conv2d_86 (Conv2D)	(None, 89, 1, 36)	21636	reshape_22[0][0]
conv2d_87 (Conv2D)	(None, 88, 1, 36)	32436	reshape_22[0][0]
conv2d_88 (Conv2D)	(None, 86, 1, 36)	54036	reshape_22[0][0]
max_pooling2d_85 (MaxPooling2D)	(None, 1, 1, 36)	0	conv2d_85[0][0]
max_pooling2d_86 (MaxPooling2D)	(None, 1, 1, 36)	0	conv2d_86[0][0]
max_pooling2d_87 (MaxPooling2D)	(None, 1, 1, 36)	0	conv2d_87[0][0]
max_pooling2d_88 (MaxPooling2D)	(None, 1, 1, 36)	0	conv2d_88[0][0]
concatenate_22 (Concatenate)	(None, 4, 1, 36)	0	max_pooling2d_85[0][0] max_pooling2d_86[0][0] max_pooling2d_87[0][0] max_pooling2d_88[0][0]
flatten_22 (Flatten)	(None, 144)	0	concatenate_22[0][0]
dropout_22 (Dropout)	(None, 144)	0	flatten_22[0][0]
dense_22 (Dense)	(None, 3)	435	dropout_22[0][0]
Total params: 46,073,679			
Trainable params: 46,073,679			
Non-trainable params: 0			



# Recurrent Neural Network



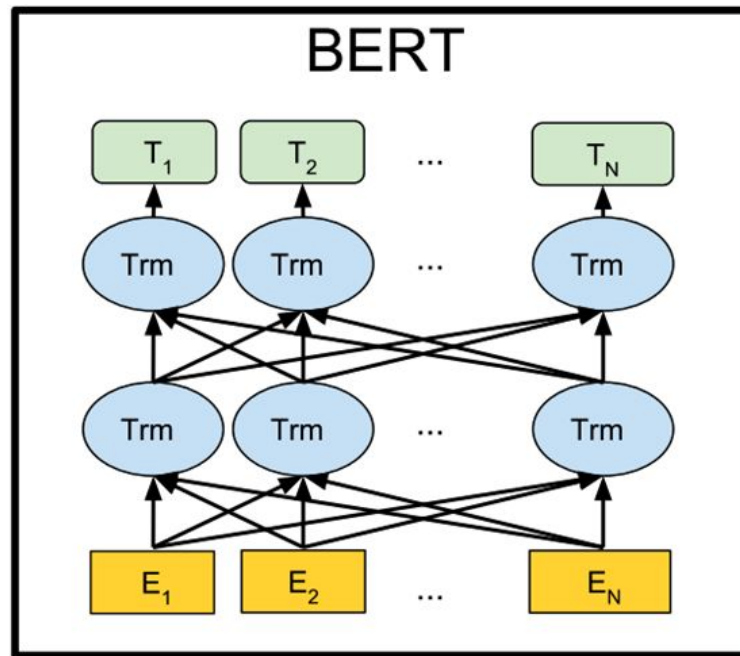
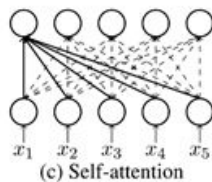
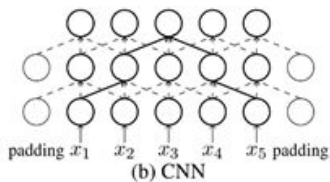
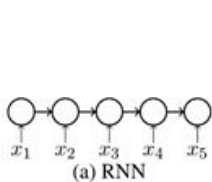
Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	(None, 90)	0	
embedding_1 (Embedding)	(None, 90, 300)	45954300	input_3[0][0]
spatial_dropout1d_3 (SpatialDro	(None, 90, 300)	0	embedding_1[2][0]
bidirectional_5 (Bidirectional)	(None, 90, 128)	186880	spatial_dropout1d_3[0][0]
bidirectional_6 (Bidirectional)	(None, 90, 128)	74112	bidirectional_5[0][0]
global_average_pooling1d_3 (Glo	(None, 128)	0	bidirectional_6[0][0]
global_max_pooling1d_3 (GlobalM	(None, 128)	0	bidirectional_6[0][0]
concatenate_3 (Concatenate)	(None, 256)	0	global_average_pooling1d_3[0][0] global_max_pooling1d_3[0][0]
dense_5 (Dense)	(None, 16)	4112	concatenate_3[0][0]
dropout_3 (Dropout)	(None, 16)	0	dense_5[0][0]
dense_6 (Dense)	(None, 3)	51	dropout_3[0][0]
Total params: 46,219,455			
Trainable params: 265,155			
Non-trainable params: 45,954,300			



# BERT: Bidirectional Encoder Representations from Transformer

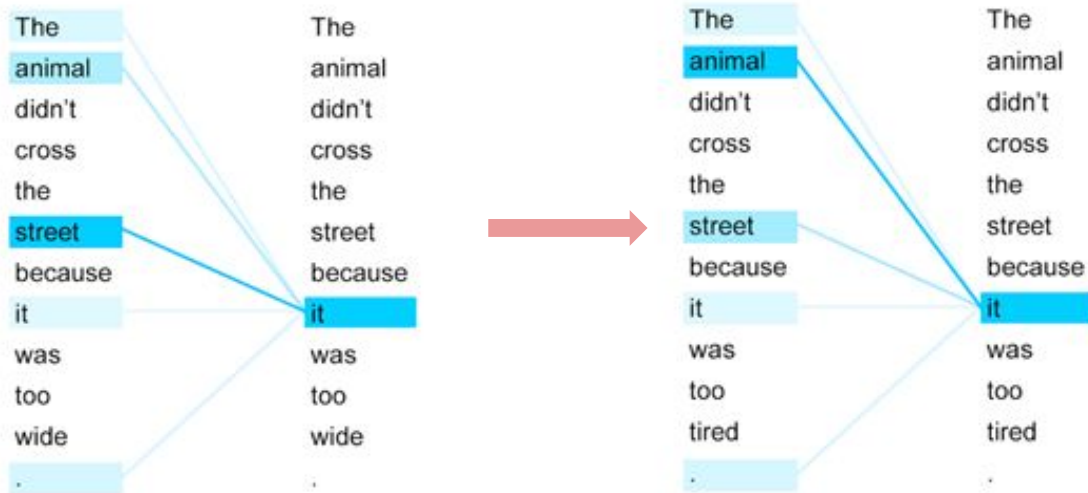


- Invented by Google (2017)
- Based on *transformer*
- Outperform RNN & CNN
- Advantage above traditional NN: any position can attend to all position





# BERT: Bidirectional Encoder Representations from Transformer



5

**Result**

Keyword:

# Result of **Baseline** model



	Accuracy	F1_Score (Marco)	Train_Time
Random_Forest_90	40.242%	0.3964	248s
Random_Forest_1000	40.244%	0.3958	2615s
SVM	40.756%	0.2697	17362s

# Result of **Advanced** model



	Accuracy	F1_Score (Marco)	F1_Score (‘Negative’)	F1_Score (‘Neutral’)	F1_Score (‘Positive’)
CNN	66%	0.66	0.70	0.62	0.62
RNN	66%	0.66	0.70	0.63	0.66
DistillBERT	68%	0.68	0.76	0.66	0.63

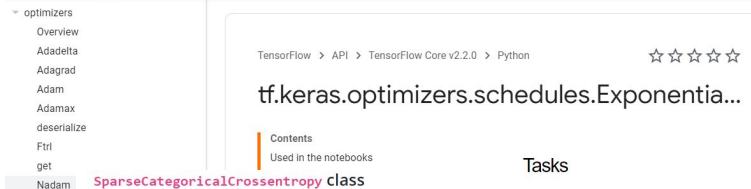
# Conclusion & Reflection

Keyword: Pre-Trained Language Model,  
Know your data, Task-Specification,

# Key Points

- Know your data
  - Bot data
  - Missing values
  - Repeated reviews
- Task Specification - fine-grain vs coarse-grain
- Pre-Trained Language Model
  - Accessible SOTA research for all

# Unlimited Classroom - Give & Take



**SparseCategoricalCrossentropy class**

```
tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=False, reduction="auto", name="sparse_categorical_crossentropy")
```

Multi-Class Classification

One-of-many classification. Each sample can belong to one of a set of classes. The target is a vector of scores. The target is a single class and negative classes. This task is treated as a single classification problem.







Computes the crossentropy loss between the labels and predictions.

Use this crossentropy loss function when there are two or more label classes. Each sample can belong to one or more classes. If you want to provide labels using one-hot representation, use `tf.keras.losses.CategoricalCrossentropy`. There should be `# classes` floating point values per feature for `y_true` and a single floating point value per feature for `y_pred`.

In the snippet below, there is a single floating point value per example for `y_true` and a single floating point value per example for `y_pred`. The shape of `y_true` is `[batch_size, num_classes]`.

Standalone usage:

```
>>> y_true = [1, 2]
>>> y_pred = [[0.05, 0.95, 0], [0.1, 0.8, 0.1]]
>>> # Using 'auto'/'sum over batch size' reduction type.
>>> sccce = tf.keras.losses.SparseCategoricalCrossentropy()
>>> sccce(y_true, y_pred).numpy()
1.177
```

Multi-Class	
C = 3	Samples
	
	
	
Labels (t)	
	[0 0 1] [1 0 0] [0 1 0]

kaggle



## Sentiment CNN, RNN-Atte, RNN-CNN, RNN-HierCNN

Python notebook using data from [multiple data sources](#) · 897 views · 1y ago



### CNN Model

### Spell checking

In [45]:

```
def build_vocab(
    voc
    tim
    out
```

We can check which words are absent from the embeddings and use the Levenshtein distance.

Finding the words that are often used in texts but not in embeddings and fixing them leads to great gains in accuracy. Using the Levenshtein's distance we can augment the dictionary 'vocab\_mapper'.



## Fastai with Transformers (BERT, RoBERTa, ...)

Python notebook using data from [Sentiment Analysis on Movie Reviews](#) · 11,765 views · 2mo ago

[Tutorial, nlp, multiclass classification](#) · +2 more

Fastai with HuggingFace Transformers (BERT, RoBERTa, XLNet, XLM, DistilBERT)




N.B. This implementation is a supplement of the Medium article "Fastai with Transformers (BERT, RoBERTa, XLNet, XLM, DistilBERT)".

Also, remember the upvote button is next to the fork button, and it's free too! 🙌

... but in the embeddings

# Unlimited Classroom - Give & Take





Search

Home

Compete

**Data**


Notebooks


Discuss


Courses


More


Recently Viewed

 WebMD Drug Reviews ...

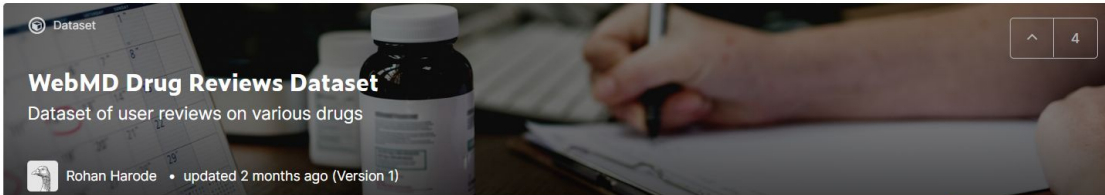
 NLP - EDA, Bag of Wor...

 Glove Pre-Trained Wor...

 Basic EDA, Cleaning an...


 GloVe: Global Vectors f...

Dataset



WebMD Drug Reviews Dataset

Dataset of user reviews on various drugs

 Rohan Harode • updated 2 months ago (Version 1)

Data

Tasks

**Kernels**

Discussion

Activity


Metadata

Download (169 MB)

New Notebook

Your Recent Notebooks

Data\_Cleaning

 Private

running for 11 minutes · [Edit](#) · [Stop](#)

GPU quota: 30h remaining

Public

Your Work

Shared With You

Favorites

Sort by

Hotness

Outputs

Languages

Tags

Search notebooks

No notebooks to show

32





# Thanks!

Any questions?



# Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by [SlidesCarnival](#)
- Photographs by [Unsplash](#)