# Text Classification: NLP

BBC Dataset

Enron Email Dataset

## BERT

# BERT

Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova
Google AI Language

## BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova Google AI Language

- BERT stands for Bidirectional Encoder Representations from Transformers
- The pre-trained BERT model can be fine tuned with just one additional output layer to create state-of-the-art models. No substantial task specific changes in architecture required
- BERT incorporates context from both directions as opposed to unidirectional models which can only attend to previous tokens in the self-attention layers of the transformer
- **Pretraining:** BERT uses “masked language model” (MLM) and “next sentence prediction” in pre-training.
    - Masked Language Model objective is to predict the original vocabulary id of the masked word based on the context
    - Next Sentence Prediction involves predicting the next sentence from two options
- Pre-training corpus: BooksCorpus (800M words) and English Wikipedia (2,500M words)
- **Finetuning:** BERT model is first initialized with the pre-trained parameters, and all the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters.
- BERT uses unified architecture across different tasks. There is minimal difference between the pre-trained architecture and the final downstream architecture.

## BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
## Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova
## Google AI Language

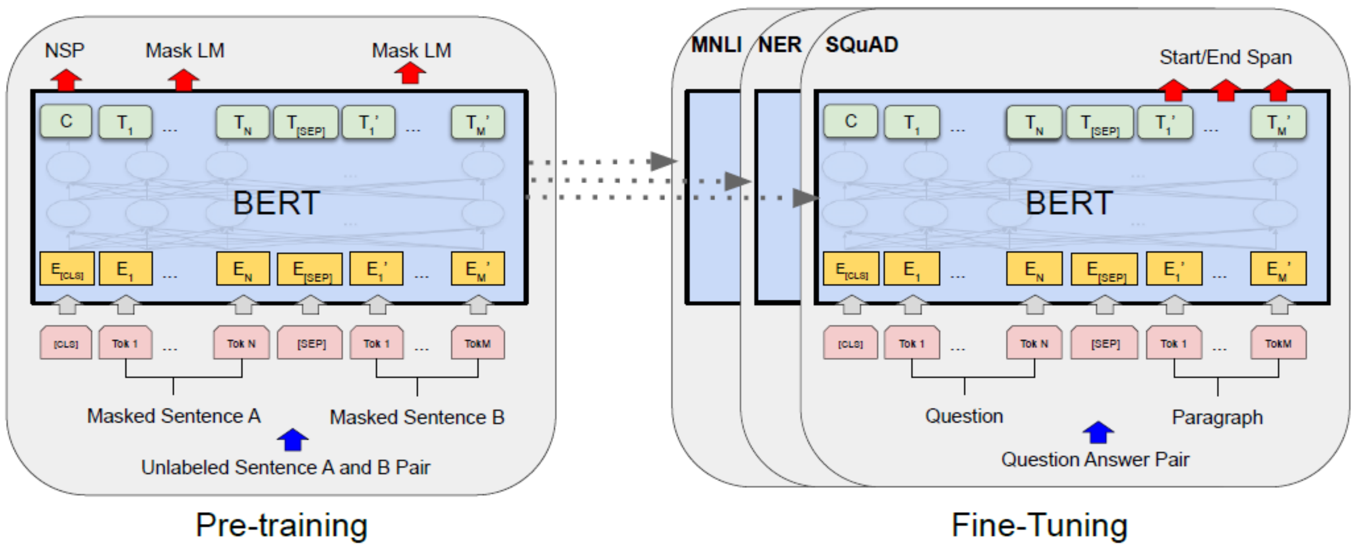


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

## BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova Google AI Language

- BERT’s model architecture is a multi-layer bidirectional Transformer encoder
- The paper deals with two model sizes:
- BERTBASE:
  - Number of layers of transformer block (L) =12
  - Hidden size (H) =768
  - Number of self attention heads (A) =12,
  - Total Parameters= 110M)
- BERTLARGE (L=24, H=1024, A=16, Total Parameters=340M)
  - Number of layers of transformer block (L) =24
  - Hidden size (H) =1024
  - Number of self attention heads (A) =16
  - Total Parameters= 340 M

## BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova Google AI Language

- BERT fine-tuning results on 11 NLP tasks are presented:
  - Glue
  - SQuAD v.1.1
  - SQuaD v.2.2
  - SWAG
- The paper demonstrates that bidirectional architectures pretrained on a large corpus can successfully tackle a broad set of NLP tasks.

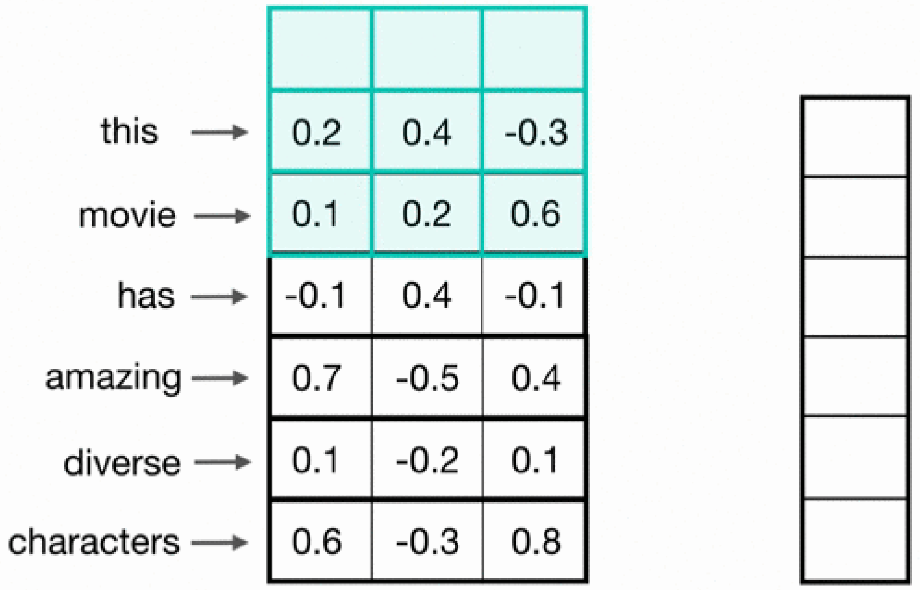
## CNN

## Convolutional Neural Networks for Sentence Classification

Yoon Kim
New York University
yhk255@nyu.edu

- This paper presents a series of experiments with convolutional neural networks (CNN) trained on top of pre-trained word vectors for sentence-level classification tasks.
- Simple CNN on static vectors achieves excellent results on multiple benchmarks.
- Learning task-specific vectors through fine-tuning offers further gains in performance.
- The model further propose a simple modification to the architecture to allow for the use of both task-specific and static vectors.
- The CNN models proposed improve upon the state of the art on 4 out of 7 tasks, which include sentiment analysis and question classification.

Convolutional Neural Networks for Sentence Classification
Yoon Kim
New York University
yhk255@nyu.edu



|  |  |  |  |
|---|---|---|---|
| this | 0.2 | 0.4 | -0.3 |
| movie | 0.1 | 0.2 | 0.6 |
| has | -0.1 | 0.4 | -0.1 |
| amazing | 0.7 | -0.5 | 0.4 |
| diverse | 0.1 | -0.2 | 0.1 |
| characters | 0.6 | -0.3 | 0.8 |

*Source: https://cezannec.github.io/CNN_Text_Classification*

Convolutional Neural Networks for Sentence Classification

Yoon Kim
New York University
yhk255@nyu.edu

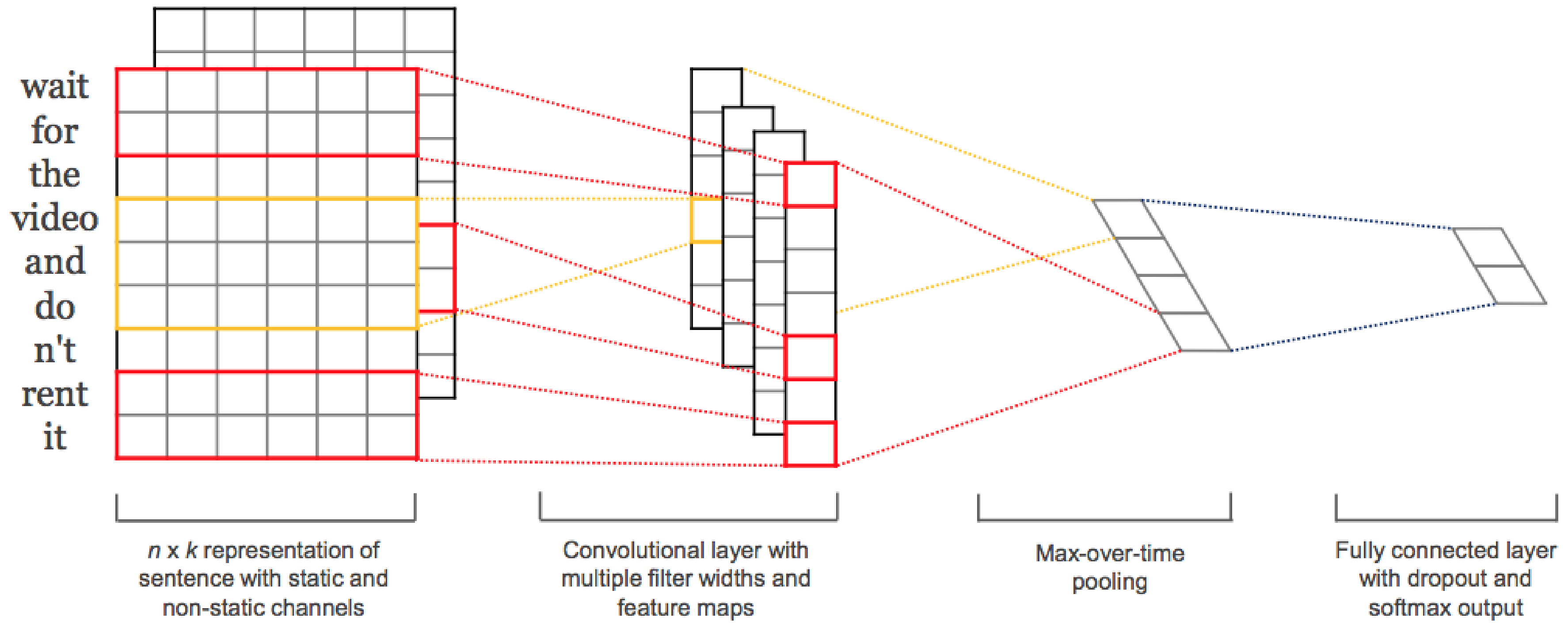

wait
for
the
video
and
do
n't
rent
it
*n* x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 1: Model architecture with two channels for an example sentence.

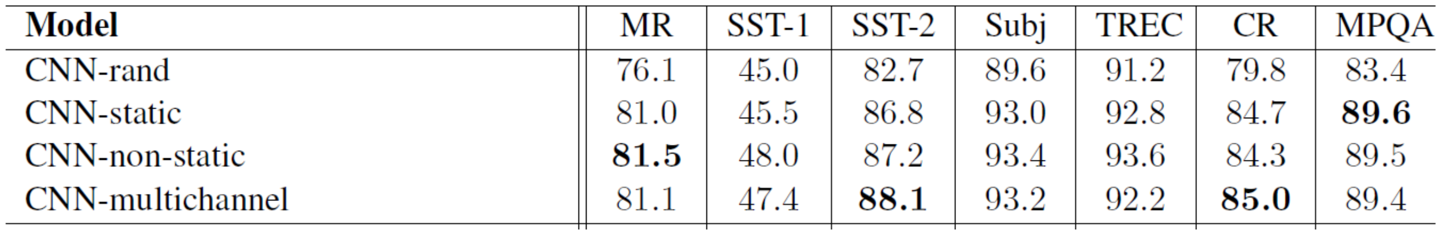
## Convolutional Neural Networks for Sentence Classification

Yoon Kim
New York University
yhk255@nyu.edu

- Datasets tested on:
- MR: Movie reviews with one sentence per review. Classification involves detecting positive/ negative reviews
- SST-1: Stanford Sentiment Treebank—an extension of MR but with train/dev/test splits provided and fine-grained labels (very positive, positive, neutral, negative, very negative)
- SST-2: Same as SST-1 but with neutral reviews removed and binary labels.
- Subj: Subjectivity dataset where the task is to classify a sentence as being subjective or objective
- TREC: TREC question dataset—task involves classifying a question into 6 question types (whether the question is about person, location, numeric information, etc.)
- CR: Customer reviews of various products (cameras, MP3s etc.). Task is to predict positive/ negative reviews

## Convolutional Neural Networks for Sentence Classification Yoon Kim New York University yhk255@nyu.edu

- Several variants of the model are tested:
- CNN-rand: Baseline model where all words are randomly initialized and then modified during training.
- CNN-static: A model with pre-trained vectors from word2vec. All words— including the unknown ones that are randomly initialized—are kept static and only the other parameters of the model are learned.
- CNN-non-static: Same as above but the pretrained vectors are fine-tuned for each task.
- CNN-multichannel: A model with two sets of word vectors. Each set of vectors is treated as a 'channel' and each filter is applied to both channels, but gradients are backpropagated only through one of the channels. Hence the model is able to fine-tune one set of vectors while keeping the other static. Both channels are initialized with word2vec.



| **Model** | MR | SST-1 | SST-2 | Subj | TREC | CR | MPQA |
|---|---|---|---|---|---|---|---|
| CNN-rand | 76.1 | 45.0 | 82.7 | 89.6 | 91.2 | 79.8 | 83.4 |
| CNN-static | 81.0 | 45.5 | 86.8 | 93.0 | 92.8 | 84.7 | **89.6** |
| CNN-non-static | **81.5** | 48.0 | 87.2 | 93.4 | 93.6 | 84.3 | 89.5 |
| CNN-multichannel | 81.1 | 47.4 | **88.1** | 93.2 | 92.2 | **85.0** | 89.4 |

## Convolutional Neural Networks for Sentence Classification

Yoon Kim
New York University
yhk255@nyu.edu

- Despite little tuning of hyperparameters, a simple CNN with one layer of convolution performs remarkably well. Our results add to the well-established evidence that unsupervised pre-training of word vectors is an important ingredient in deep learning for NLP.

## Supervised and Semi-Supervised Text Categorization using LSTM for Region Embeddings

Rie Johnson RIEJOHNSON@GMAIL.COM RJ Research Consulting, Tarrytown NY, USA Tong Zhang TONGZHANG@BAIDU.COM Big Data Lab, Baidu Inc, Beijing, China

- This paper is on text classification. Explores ways to improve text classification based on One-Hot CNN.
- It takes the work on One-Hot CNN based text categorization model (Linear) and adds a non-linear feature generator consisting of 'text region embeddings + Pooling' to get better text categorization results.
- LSTM was used to embed and extend the one-hot neural layers to make a more sophisticated region embedding with variable sizes to derive better classifications.
- This paper explored this method with semi-supervised and supervised data.
- The best results were obtained by combining region embeddings in the form of LSTM and convolution layers trained on unlabeled data.

Supervised and Semi-Supervised Text Categorization using LSTM for Region Embeddings
Rie Johnson RIEJOHNSON@GMAIL.COM RJ Research Consulting, Tarrytown NY, USA Tong Zhang TONGZHANG@BAIDU.COM Big Data Lab, Baidu Inc, Beijing, China

## The Models used to compare

- In One-hot CNN – This had only one layer of fixed region of neurons. And one pooling layer.
- Here the disadvantage is the number of input neurons (ex: region size is 5) is fixed.
- Wv-LSTM is the traditional word vector based RNN neural network
- Oh-LSTM is the One-hot LSTM which uses the word occurrences without using the vector embeddings.
- Bi-directional one hot LSTM with pooling which uses one hot vectors for faster prediction

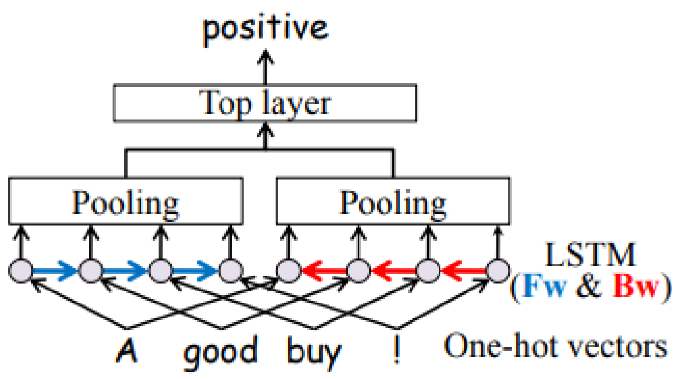


*Figure 4.* *oh-2LSTMp*: our one-hot bidirectional LSTM with pooling.

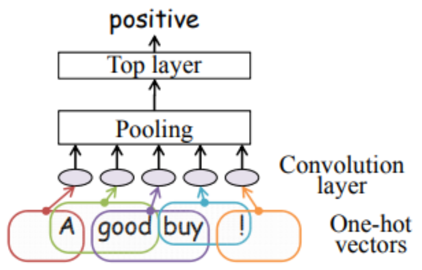


*Figure 1.* One-hot CNN (*oh-CNN*) [JZ15a]

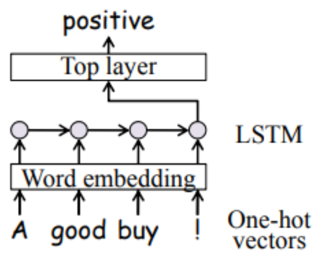


*Figure 2.* Word vector LSTM (*wv-LSTM*) as in [DL15].

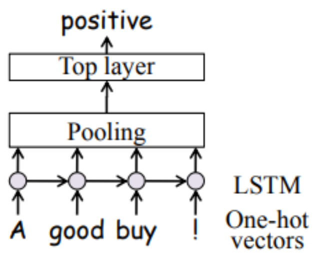


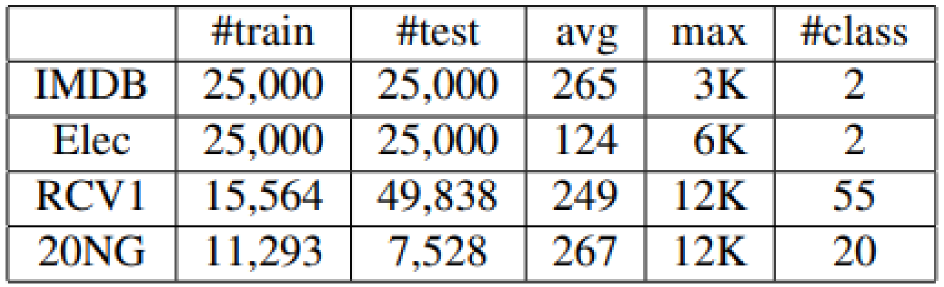
*Figure 3.* One-hot LSTM with pooling (*oh-LSTMp*).

Supervised and Semi-Supervised Text Categorization using LSTM for Region Embeddings
Rie Johnson RIEJOHNSON@GMAIL.COM RJ Research Consulting, Tarrytown NY, USA Tong Zhang TONGZHANG@BAIDU.COM Big Data Lab, Baidu Inc, Beijing, China

## Why Pooling instead of input/output gates?

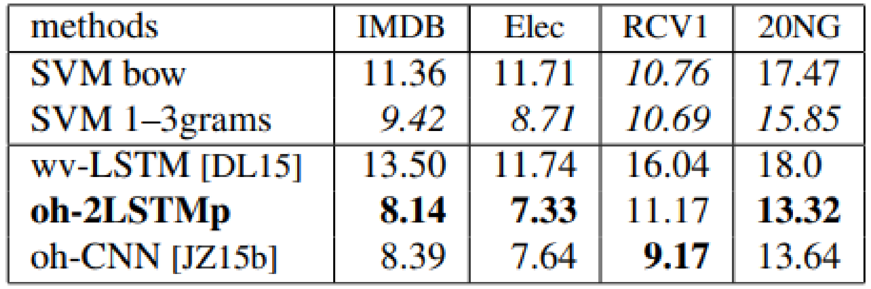
- pooling has the merit of enabling faster training via chopping. Since we set the goal of LSTM to embedding text regions instead of documents, it is no longer crucial to go through the document from the beginning to the end sequentially. At the time of training, we can chop each document into segments of a fixed length that is sufficiently long (e.g., 50 or 100) and process all the segments in a mini batch in parallel as if these segments were individual documents. (Note that this is done only in the LSTM layer and pooling is done over the entire document.)

The datasets used to compare:



| | #train | #test | avg | max | #class |
|---|---|---|---|---|---|
| IMDB | 25,000 | 25,000 | 265 | 3K | 2 |
| Elec | 25,000 | 25,000 | 124 | 6K | 2 |
| RCV1 | 15,564 | 49,838 | 249 | 12K | 55 |
| 20NG | 11,293 | 7,528 | 267 | 12K | 20 |

Percentage of error in predicting the correct class.



| methods | IMDB | Elec | RCV1 | 20NG |
|---|---|---|---|---|
| SVM bow | 11.36 | 11.71 | *10.76* | 17.47 |
| SVM 1–3grams | *9.42* | *8.71* | *10.69* | *15.85* |
| wv-LSTM [DL15] | 13.50 | 11.74 | 16.04 | 18.0 |
| **oh-2LSTMp** | **8.14** | **7.33** | 11.17 | **13.32** |
| oh-CNN [JZ15b] | 8.39 | 7.64 | **9.17** | 13.64 |

The paper explores using the traditional word vector LSTM and the improved one-hot bidirectional LSTM with pooling and they got the lowest percentage of error in 3 of the 4 datasets used with better performance.

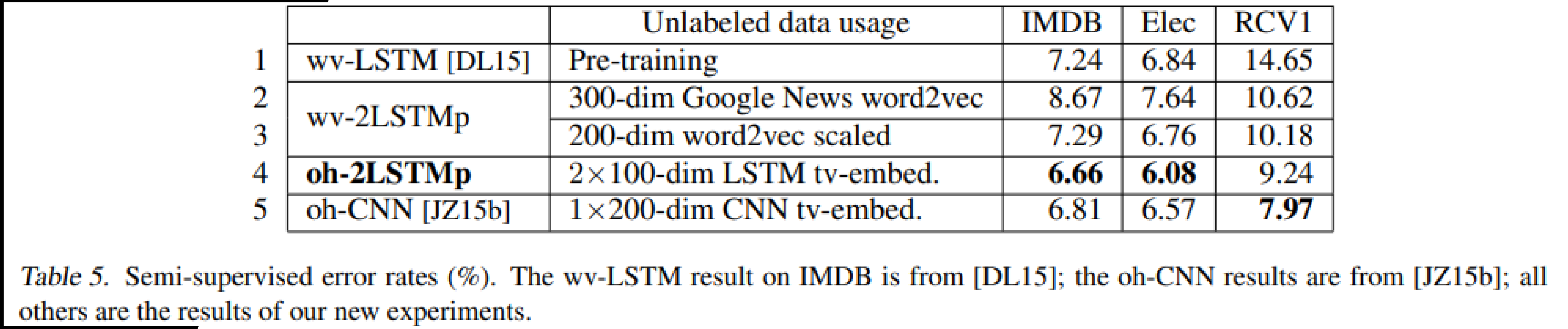
Supervised and Semi-Supervised Text Categorization using LSTM for Region Embeddings
Rie Johnson RIEJOHNSON@GMAIL.COM RJ Research Consulting, Tarrytown NY, USA Tong Zhang TONGZHANG@BAIDU.COM Big Data Lab, Baidu Inc, Beijing, China

## Conclusion

- We found this paper interesting as we were not able to get good accuracy using LSTM for predicting the categories of news articles in supervised learning.
- LSTM also had the problem of document lengths. The optimal length that worked was about 100 words. Above which training took time and the accuracy didn't improve.
- This paper gave a way to divide the documents in 50 – 100 words each. Then apply the LSTM bi-directional way on each division.

  Then apply the max pooling on the results of all the documents combined.

  The paper also concluded that unidimensional CNN with 1000 input range and two CNN layers and max pooling can outperform on classification.

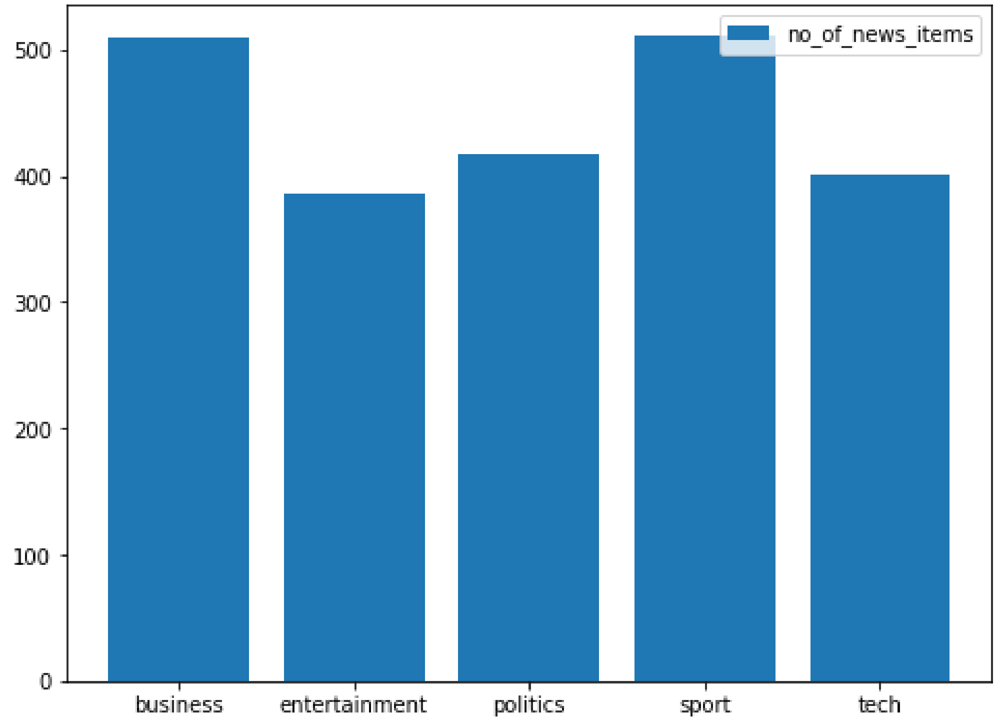


| | | Unlabeled data usage | IMDB | Elec | RCV1 |
|---|---|---|---|---|---|
| 1 | wv-LSTM [DL15] | Pre-training | 7.24 | 6.84 | 14.65 |
| 2 | wv-2LSTMp | 300-dim Google News word2vec | 8.67 | 7.64 | 10.62 |
| 3 | | 200-dim word2vec scaled | 7.29 | 6.76 | 10.18 |
| 4 | **oh-2LSTMp** | 2×100-dim LSTM tv-embed. | **6.66** | **6.08** | 9.24 |
| 5 | oh-CNN [JZ15b] | 1×200-dim CNN tv-embed. | 6.81 | 6.57 | **7.97** |

*Table 5.* Semi-supervised error rates (%). The wv-LSTM result on IMDB is from [DL15]; the oh-CNN results are from [JZ15b]; all others are the results of our new experiments.

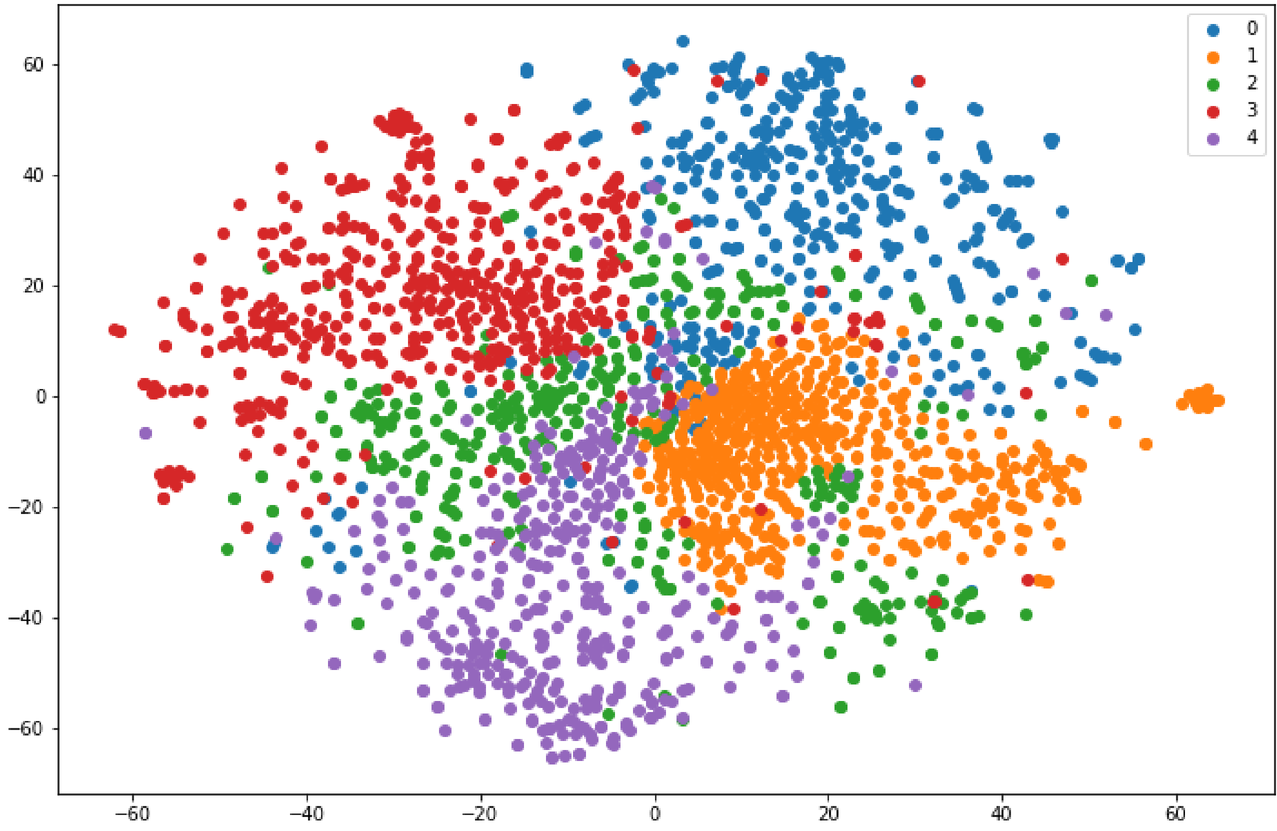
## BBC News Dataset

- 2225 news articles
- Roughly evenly distributed across:
  - Business
  - Entertainment
  - Politics
  - Sports
  - Technology
- Objective: To build a model to classify news articles



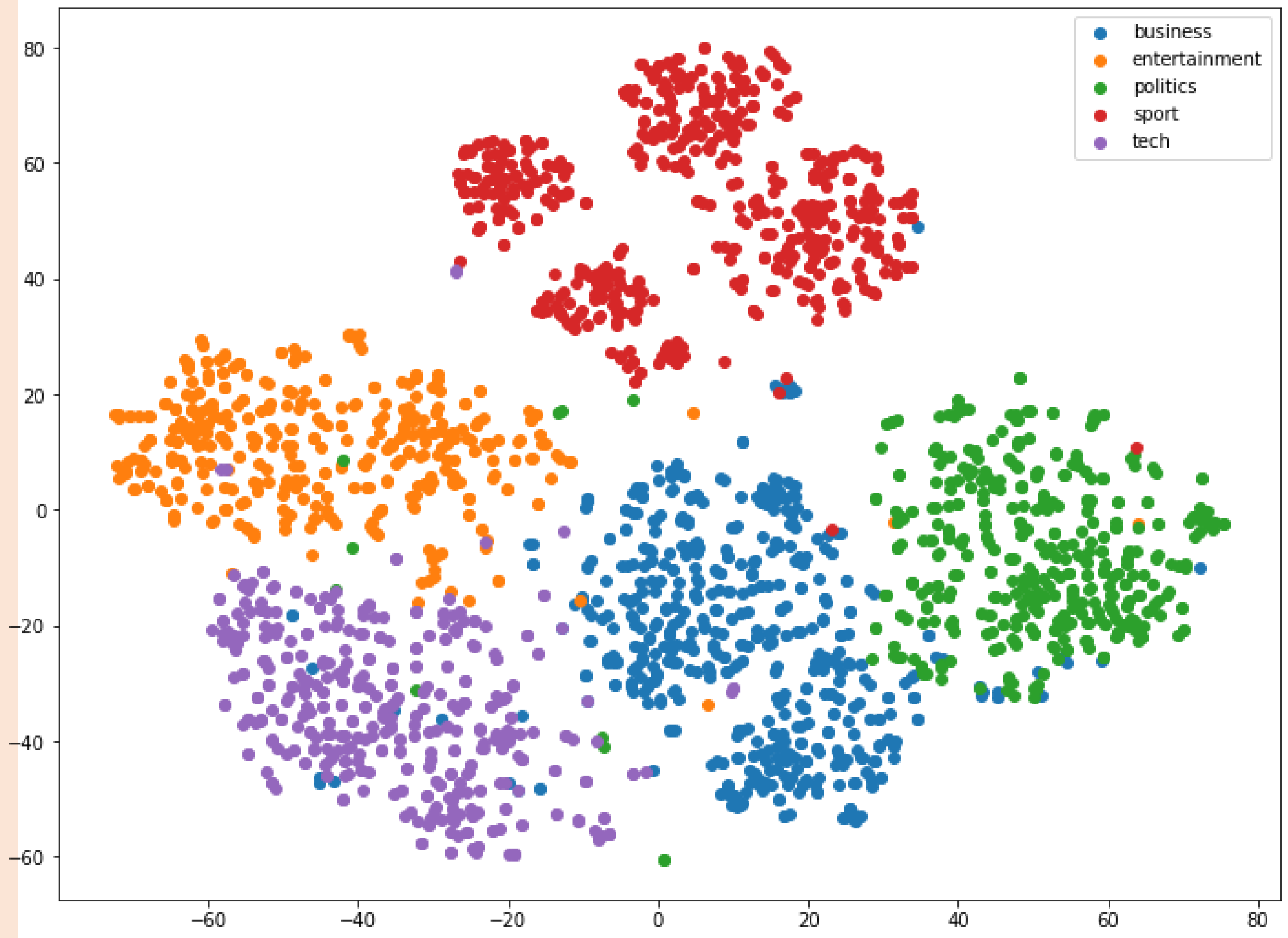
# BBC News: BoW

- Text was treated as a bag of words
    - Each article was converted as a vector of 3007 dimensions, using CountVectorizer
- Various classifiers used
- Test accuracy achieved:
    - `KNN 0.69`
    - `SVC 0.96`
    - `Random Forest 0.86`
    - `Gradient Descent 0.96`
    - `Logistic Regression 0.97`



# BBC News: TfIdf

- Each article converted into vectors of 14,415 dimensions using SKLearn TfidfVectorizer
- Various classifiers used
- Accuracy achieved:
  - `KNN 0.96`
  - `SVC 0.98`
  - `Random Forest 0.81`
  - `Gradient Descent 0.98`
  - `Logistic Regression 0.98`



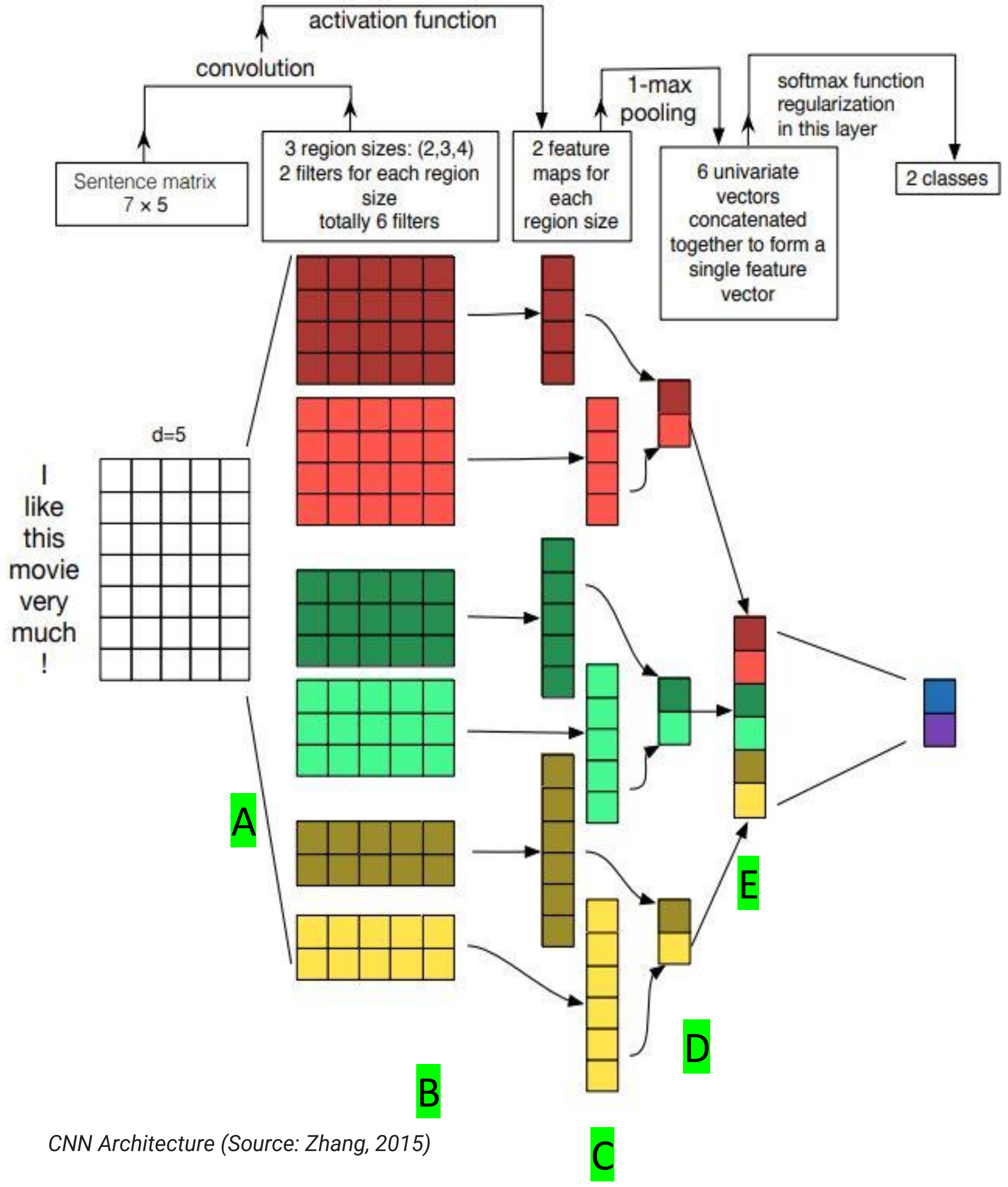
## BBC News: **CNN**

- Text converted into embeddings using FastText
  - Each article padded to maximum length of 4862 words
- 3 convolution layers of kernel sizes 3, 4, 5
- 100 filters each
- Output of each filter maxpooled to a single scalar. All such scalars concatenated to vector 300 dimensions
- Linear layer with in_channels of 300 and out_channels of 5
- Accuracy: **96.7% at 10 epochs**

| **Description** | **Values** |
|---|---|
| input word vectors | fastText |
| embedding size | 300 |
| filter sizes | (3, 4, 5) |
| num filters | (100, 100, 100) |
| activation | ReLU |
| pooling | 1-max pooling |
| dropout rate | 0.5 |

# BBC News: **CNN**

- In the current instance:
- A: 4862 x 300 array
- B: 100 nos of 3 x 300 kernels, 100 nos of 4 x 300 kernels and 100 nos of 5 x 300 kernels
- C: 100 nos of 4,860-dimension vectors, 100 nos of 4,859-dimension vectors, 100 nos of 4,858 vectos
- D: 300-dimension vector
- E: 5-dimension vector

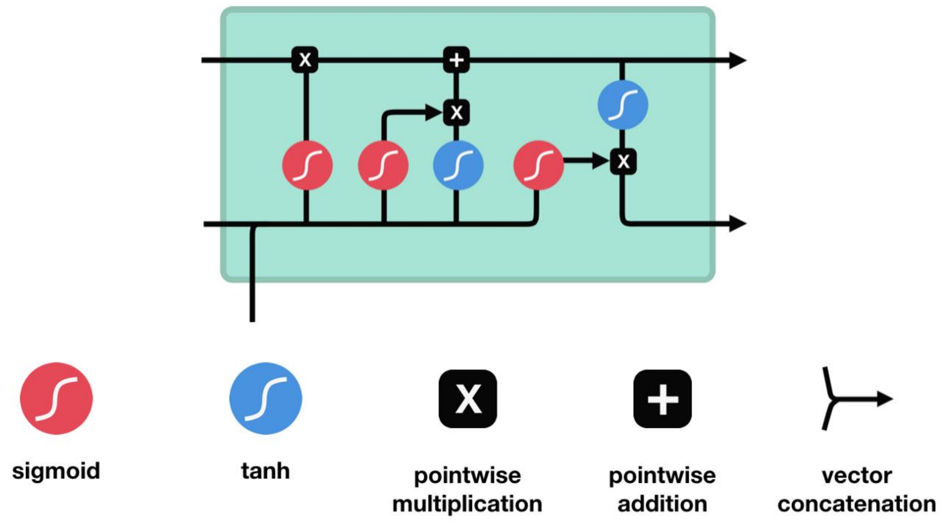

activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!
A
B
C
D
E

*CNN Architecture (Source: Zhang, 2015)*

# BBC News: LSTM

```
LSTM(
(embedding): Embedding(50108, 300)
(lstm): LSTM(300, 128, num_layers=2, batch_first=True)
(fc): Linear (in_features=128, out_features=5, bias=True)
(dropout): Dropout(p=0.5, in place=False)
        )
```

Each article truncated to 100 words
Word embedded using FastText
Accuracy of 89.7% achieved

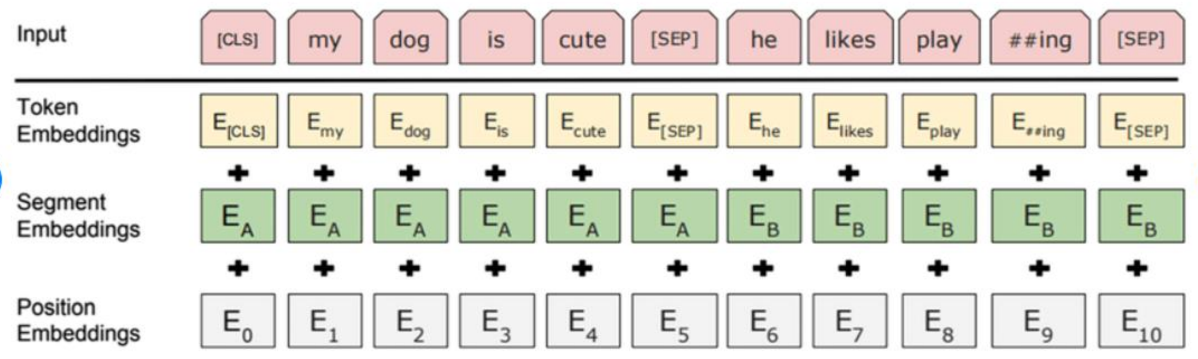


## BBC News: Bert

Embedded using BertTokenizer

BertForSequenceClassification used to classify the articles

Test accuracy of 95%



Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ $E_{my}$ $E_{dog}$ $E_{is}$ $E_{cute}$ $E_{[SEP]}$ $E_{he}$ $E_{likes}$ $E_{play}$ $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
$E_A$ $E_A$ $E_A$ $E_A$ $E_A$ $E_A$ $E_B$ $E_B$ $E_B$ $E_B$ $E_B$
Position Embeddings
$E_0$ $E_1$ $E_2$ $E_3$ $E_4$ $E_5$ $E_6$ $E_7$ $E_8$ $E_9$ $E_{10}$

Embedding (BERT tokenizer), the input embeddings are the sum of the token embeddings, the segmentation embeddings, and the position embeddings

## Enron Dataset:

- 517400 emails from Enron
- Fields extracted:
    - From address
    - To address
    - CC addresses
    - BCC addresses
    - Name of the person
    - Folder
    - Subject
    - Date
    - Body

- Objective:
    - Classify into folders for each person
- Challenges:
    - Each person has varying folders, while some have a few others have large numbers. In total, over 1000 folders
    - Large dataset, leading to crashing of the colab sessions

## Enron: KMeans

- 60,000 emails were selected at random
- Subject and body combined and TfidfVectorizer used to embed
- From, To, CC, BCC, Name were treated as additional features
- In total, 1065 features
    - 1060 (tfidf embeddings of text)
    - 5 additional features (from, to, cc, bcc, and name)
- Various algorithms used to classify
    - KNN 64.0%
    - Random Forest 72.7%
    - Gradient Descent 81.0%
    - Logistic Regression 88.3%
    - SVC 80.3%

- Challenges:
    - SVC takes too long to train. So eventually took a very small sample
    - Logistic Regression did not converge even at 500 iterations, so eventually settled for 100 iterations. It did not converge

## Enron: Tfidf

- 60,000 emails were selected at random
- Subject and body combined and TfidfVectorizer used to embed
- From, To, CC, BCC, Name were treated as  additional features
- In total, 1065 features
  - 1060 (tfidf embeddings of text)
  - 5 additional features (from, to, cc, bcc, and name)
- Various algorithms used to classify
  - KNN 64.0%
  - Random Forest 72.7%
  - Gradient Descent 81.0%
  - Logistic Regression 88.3%
  - SVC 80.3%

- Challenges:
  - SVC takes too long to train. So eventually took a very small sample
  - Logistic Regression did not converge even at 500 iterations, so eventually settled for 100 iterations. It did not converge

# Enron: LSTM

- Text: 30 words from ‘Subject’ combined with 70 words from the ‘Body’
- FastText used to embed the Text
- LSTM applied on the text
- Linear layers applied on output from LSTM combined with 5 additional features (From, To, CC, BCC, Name)
- Results: Test Accuracy of 71.0% from 3 epochs
- Simple LSTM without the 5 additional features returned an accuracy of

## Enron: Bert

- Text embedded using BertTokenizer
- BertForSequenceClassification to classify
- Accuracy:
- How to incorporate additional features?