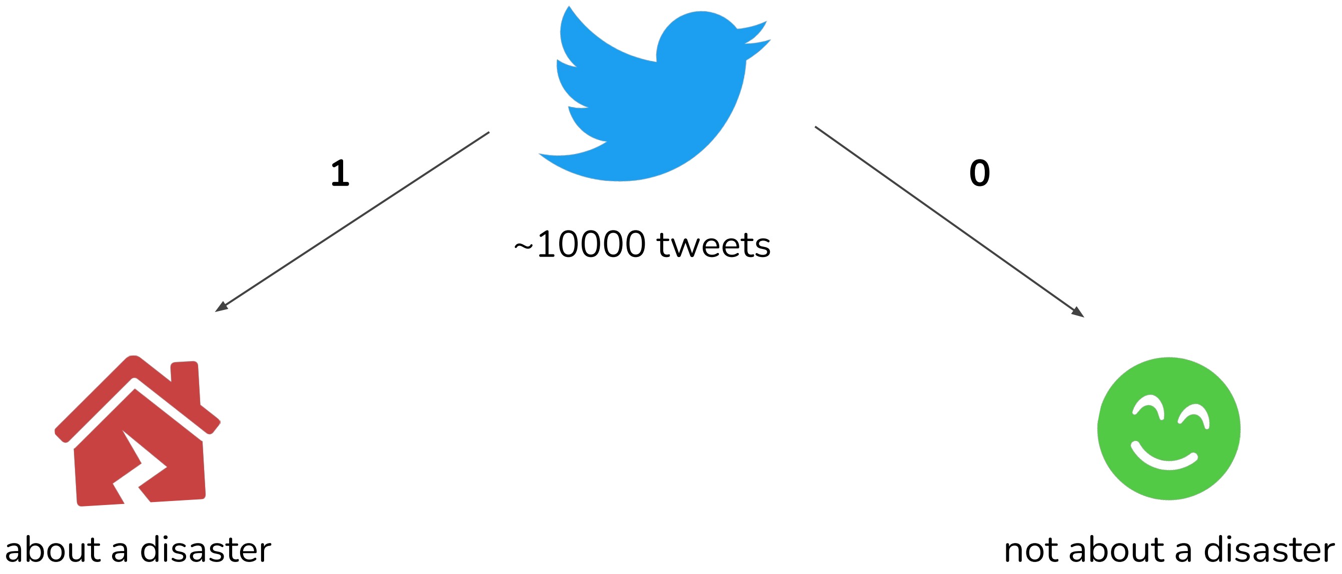
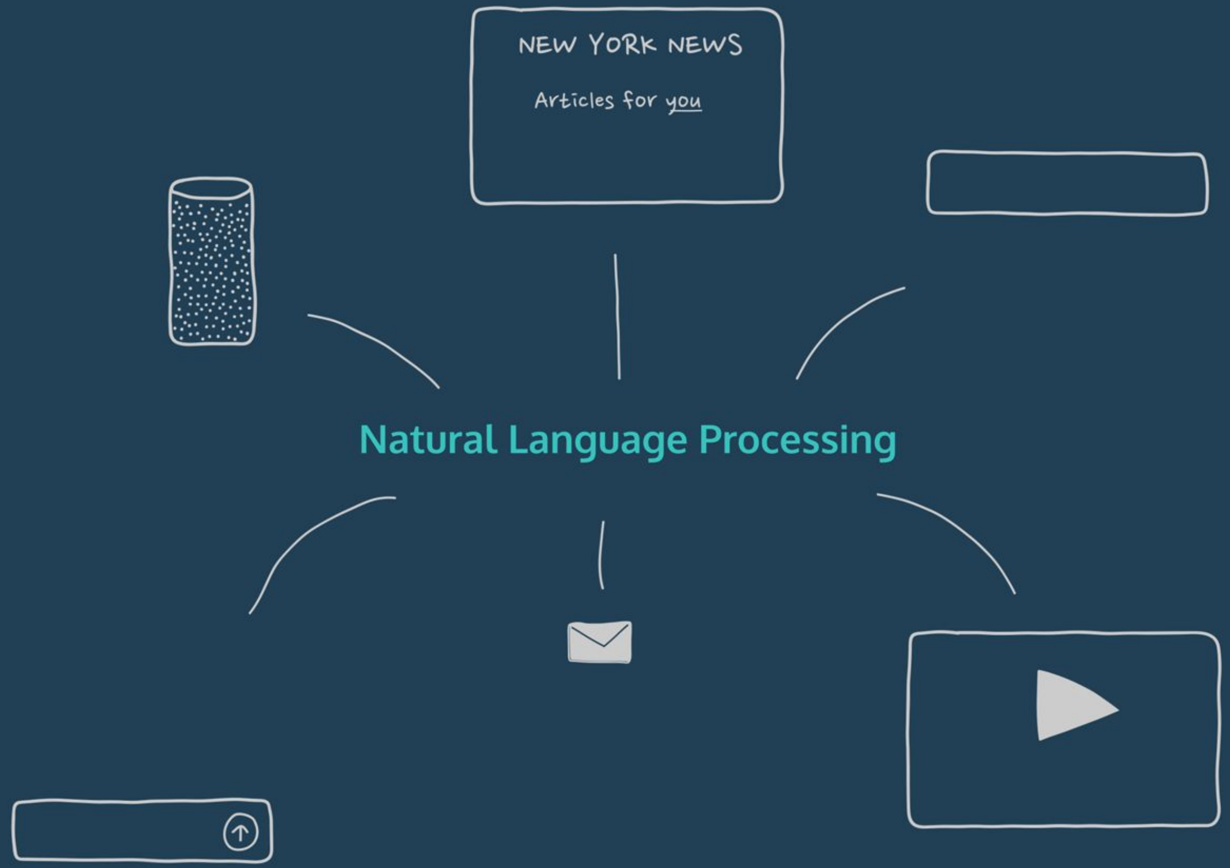
Disaster Tweet Classification

Eduardo Barros Innarelli - 170161
Victor Ferreira Ferrari - 187890
Vinícius Couto Espindola - 188115

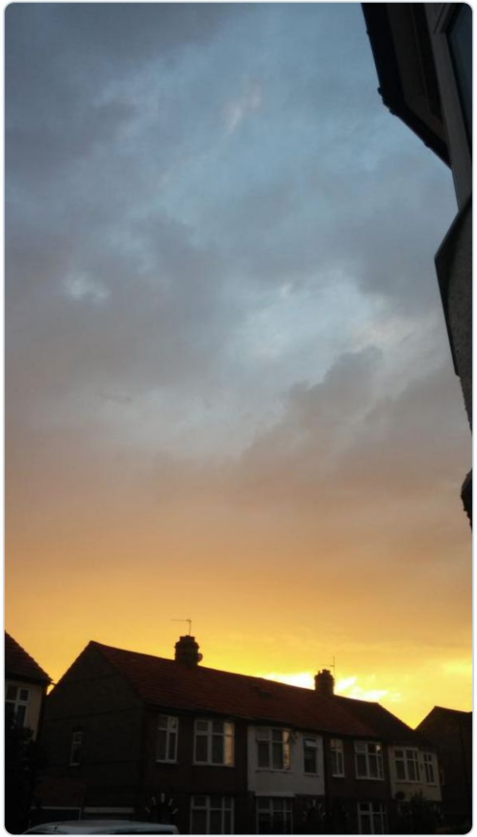
Real or Not? NLP With Disaster Tweets




NEW YORK NEWS
Articles for you
Natural Language Processing

Anna K
@AnyOtherAnnaK

On plus side LOOK AT THE SKY LAST NIGHT IT WAS ABLAZE



12:43 AM · Aug 6, 2015 · Twitter for Android

"Ablaze" is used **metaphorically**.

So, it is **not** a real disaster.

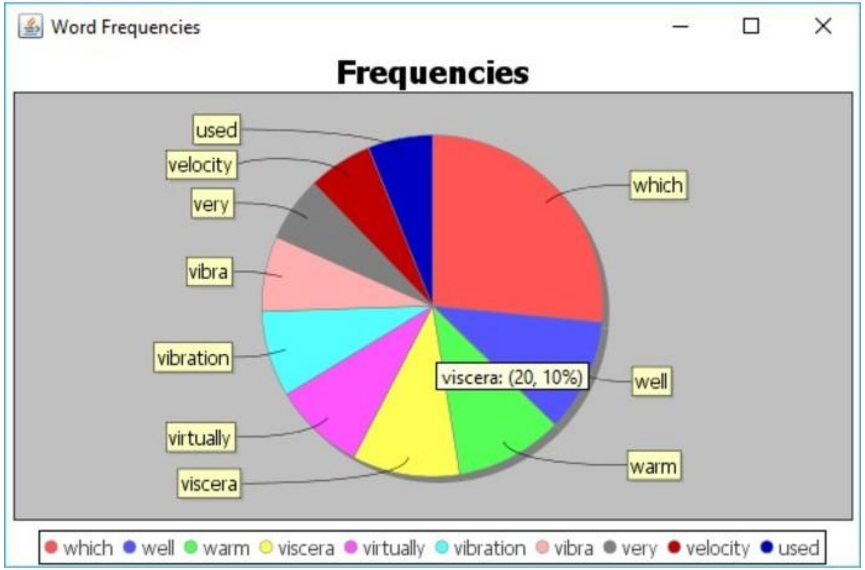
We classify it as **not real.**

Pre-Processing

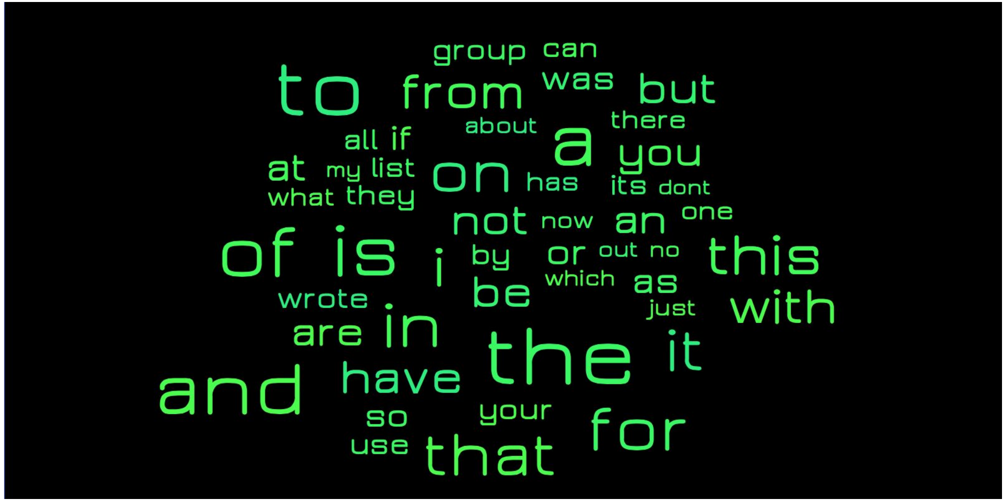
Key Points

- Remove Superfluous Data | Keep What’s Relevant
- Compress Remaining Data

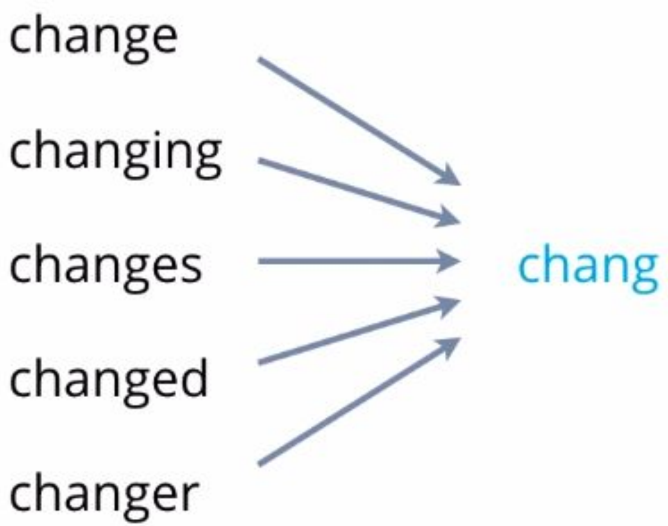
Gum would be perfection ! $\}$ Tokens

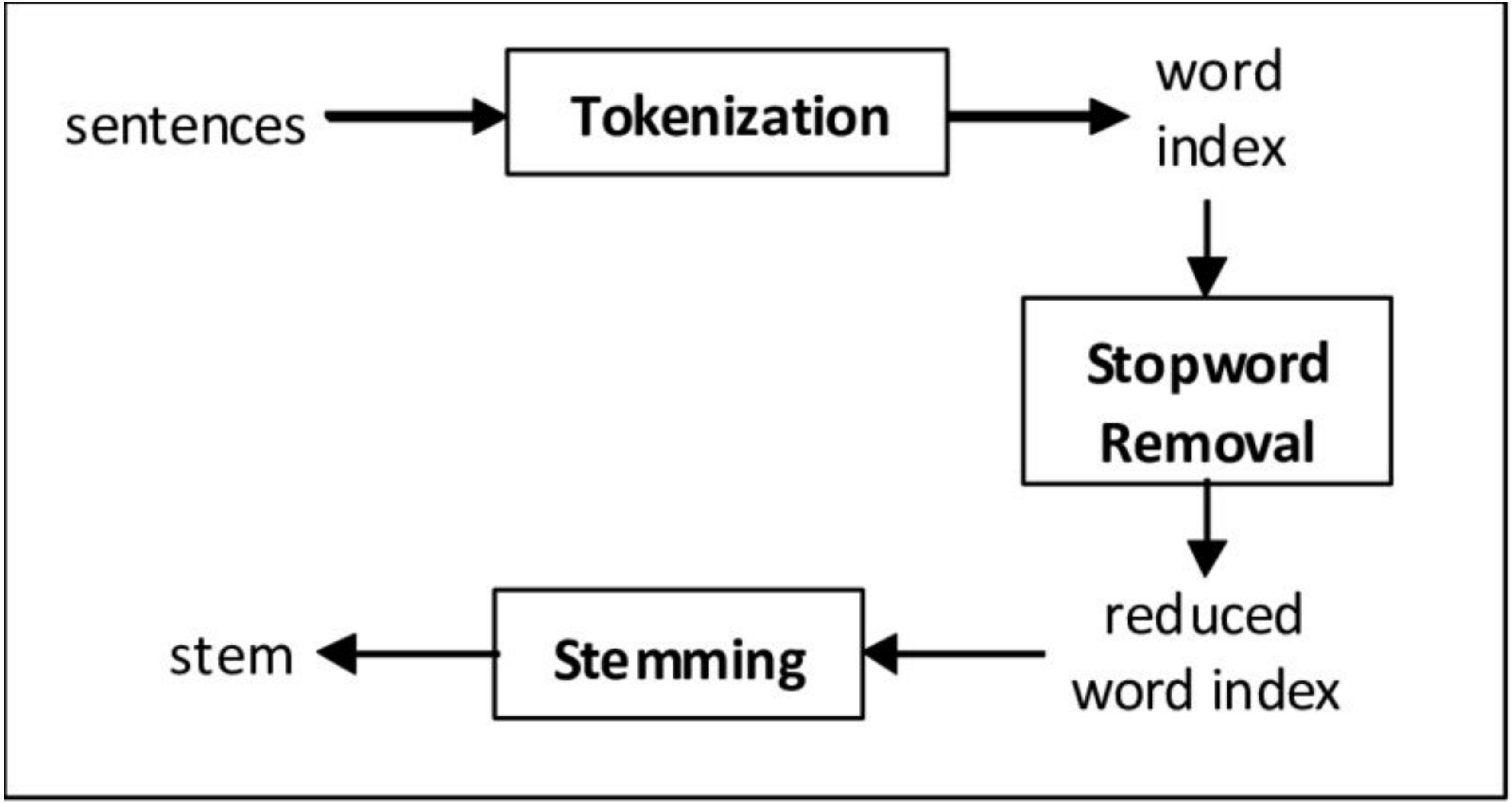



STOP


group can
to from was but
there
about
all if a
at my list you
on has its dont
what they
not now an one
of is i by or out no this
which as
be
wrote just with
are in
the it
and have
so your
for
use that

Stemming




sentences
Tokenization
word index
Stopword Removal
reduced word index
Stemming
stem

Encoding

Key Points

- How to Avoid False Word Relations?

- How to Retain information Regarding Meaning?

Label Encoding VS One Hot Encoding

- False Numeric Gradient Relation
- Words Completely Independent
  Better!

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

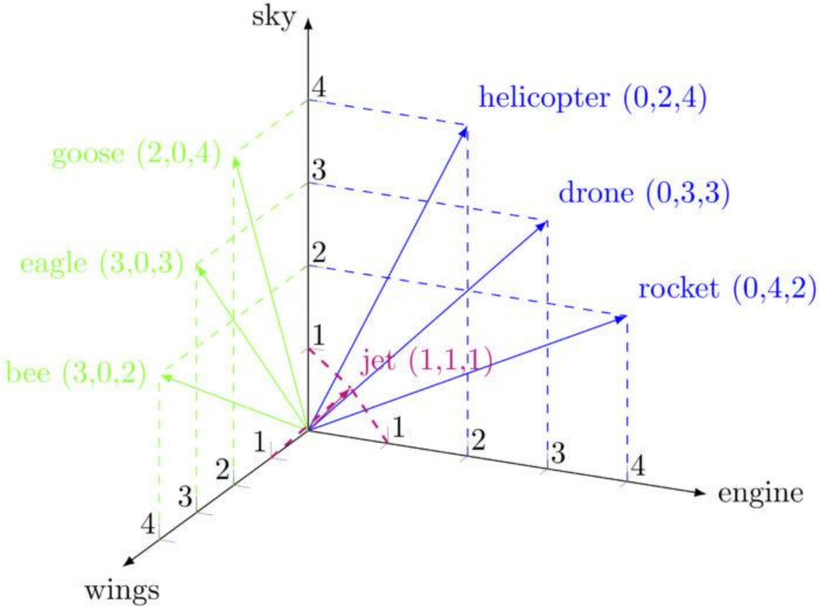
→

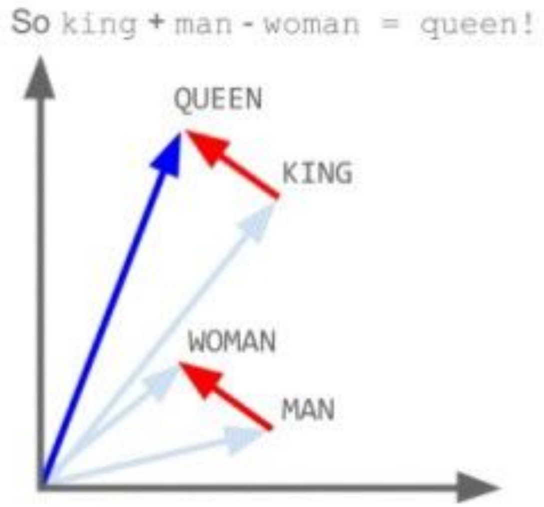
One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

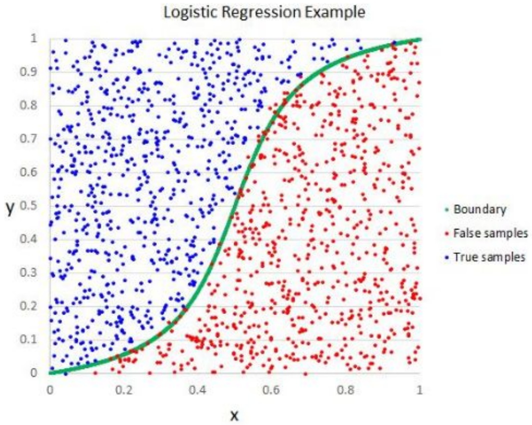
Word Embedding

- Similar Numbers to Similar Words!
  Much Better!

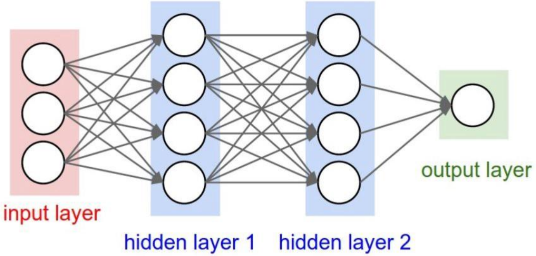




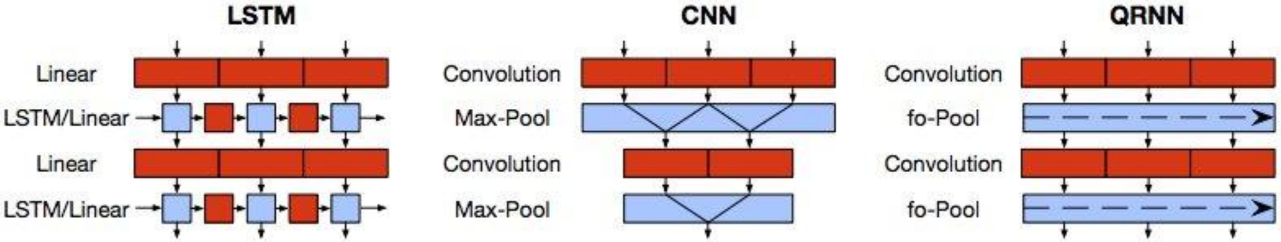
Models



Logistic Regression Example
y
x
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1
Boundary
False samples
True samples



input layer
hidden layer 1
hidden layer 2
output layer



LSTM
Linear
LSTM/Linear
Linear
LSTM/Linear
CNN
Convolution
Max-Pool
Convolution
Max-Pool
QRNN
Convolution
fo-Pool
Convolution
fo-Pool

Implementation: Python

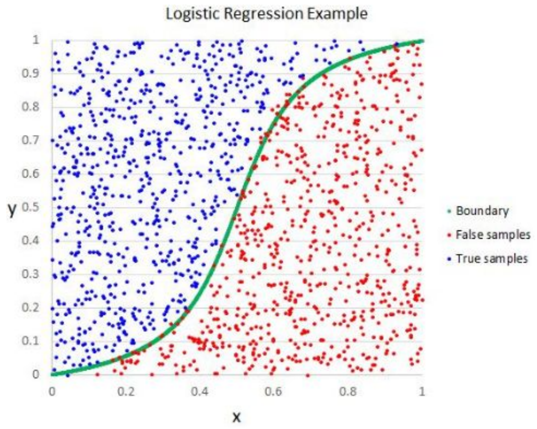
- Pre-Processing: NLTK/Keras/BERT

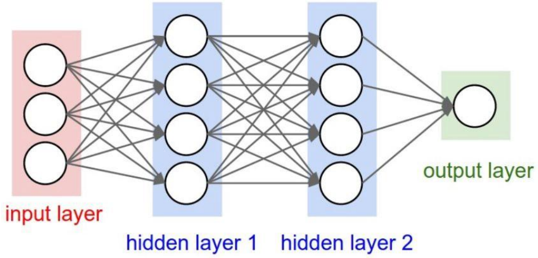
- Regression: Scikit-Learn

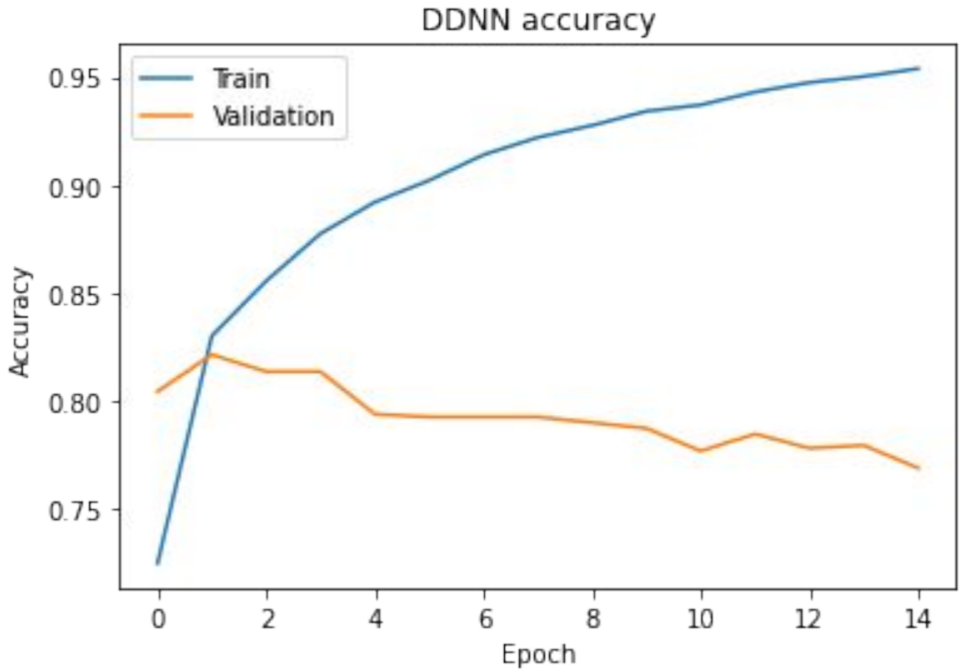
- Neural Networks: Tensorflow (Keras)

Linear Regression
Dense Neural Networks

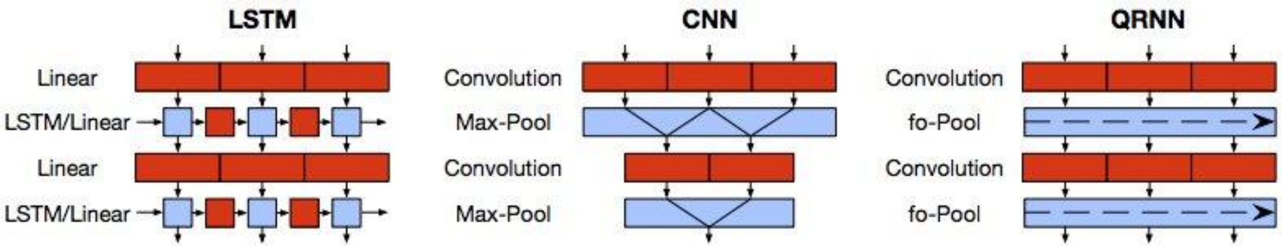
- Simple to implement
- A good starting point
- <Results>
- Not very used in NLP





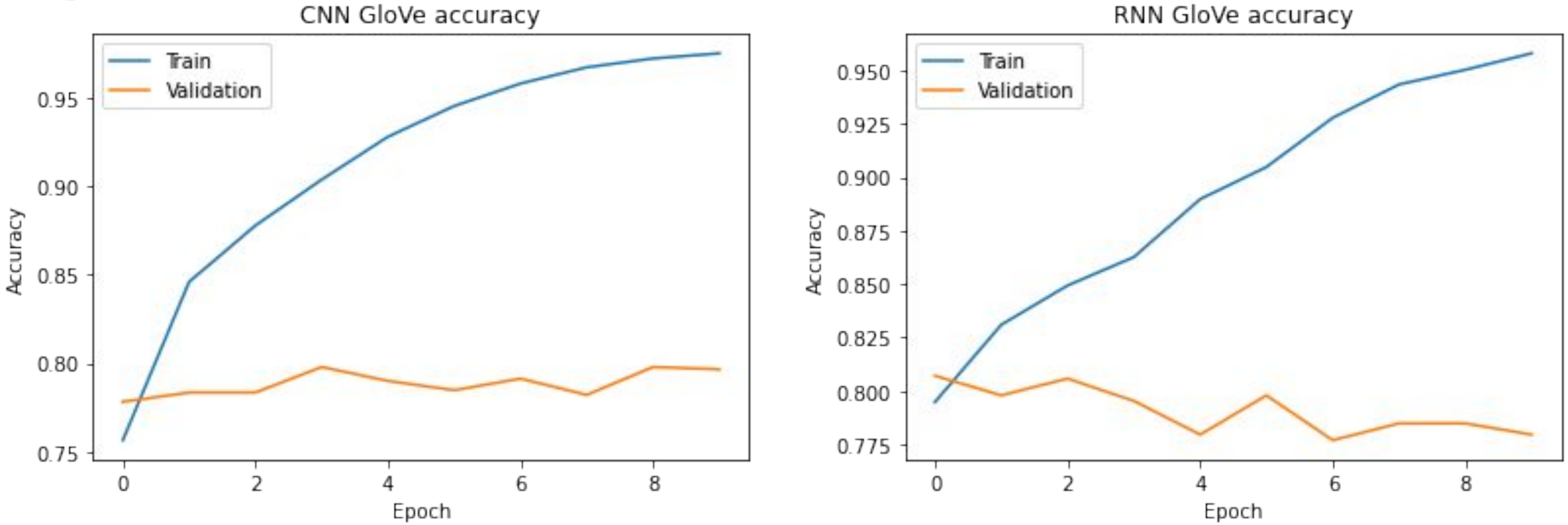

DDNN accuracy
Train
Validation
Accuracy
0.95
0.90
0.85
0.80
0.75
Epoch
0
2
4
6
8
10
12
14

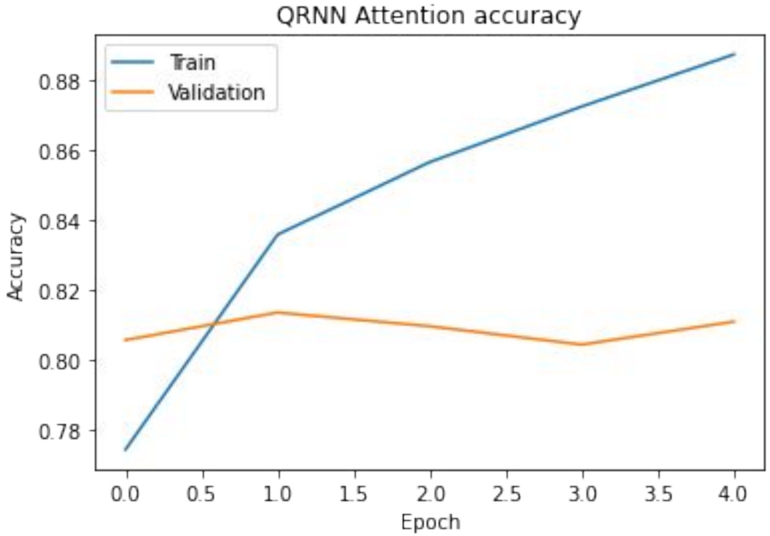
Recurrent & Convolutional Networks



- Getting the Context:
  - Recurrent considers the order
  - Convolutions associate words

- Worse results than the baseline


CNN GloVe accuracy
Train
Validation
Accuracy
Epoch
0.75
0.80
0.85
0.90
0.95
0
2
4
6
8
RNN GloVe accuracy
Train
Validation
Accuracy
Epoch
0.775
0.800
0.825
0.850
0.875
0.900
0.925
0.950
0
2
4
6
8



QRNN Attention accuracy
Train
Validation
Accuracy
0.78
0.80
0.82
0.84
0.86
0.88
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch

BERT

Bidirectional Encoder Representation from Transformers

- State-of-the-Art results on 11 NLP problems
- Why so special?

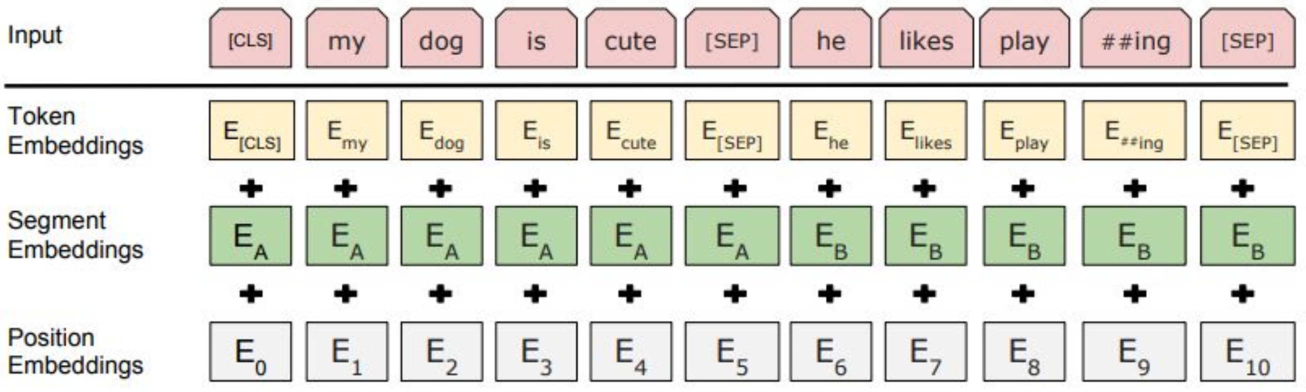
Intense Preprocessing & Pre-training

The “Swiss army knife of BERT”

- Words are hard: Ambiguous, Polysemous, Synonymous…
- Good encoding and pre-training are necessary
- Huge Pre-trained Dataset: > 2,500,000,000 Words

Input Embedding

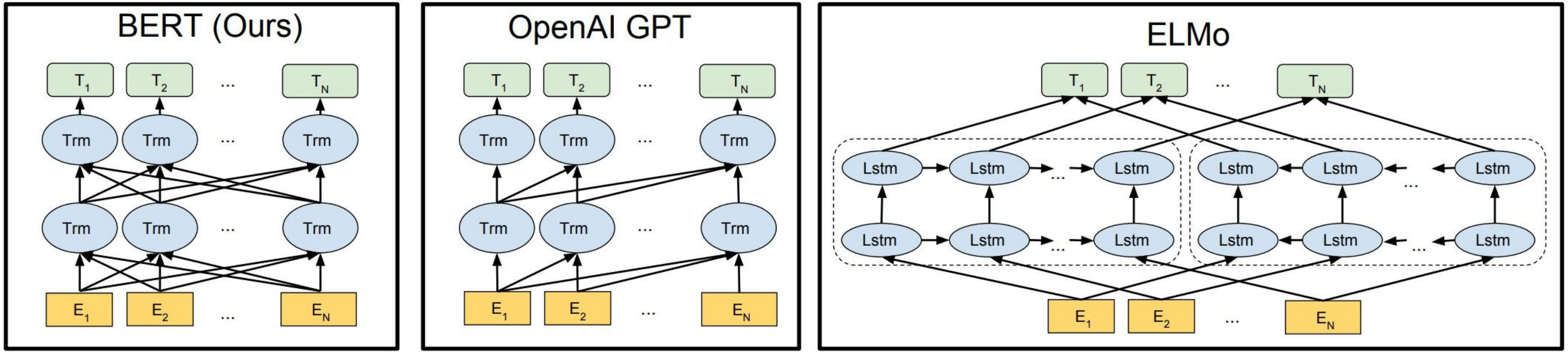
Not Just Meaning, but Also Position



- Three simultaneous embeddings
- Sum = Final Token Representation

Words Contextualized

Masked Language Model Pretraining



- Guess a word based in it’s sentence
- Deeply Bidirectional

Sentences Contextualized

Next Sentence Prediction Pretraining

[CLS] Joel is giving a talk. [SEP] The audience is enthralled. [SEP]

99% is_next_sentence
1% is_not_next_sentence

[CLS] Joel is giving a talk. [SEP] The audience is falling asleep. [SEP]

1% is_next_sentence
99% is_not_next_sentence

Fine Tuning

Transfer Learning Strategy

- BERT is enormous = overfitting in smaller Datasets
- How to use it on other problems then? **Fine Tuning**
- Further BERT's pretraining on our Dataset
- Append dense and sigmoid layers for binary classification

Final results

Method	Training Acc.	Validation Acc.
Logistic Regression	91,9%	79,8%
Dense Neural Networks	83,0%	82,1%
Recurrent Neural Networks	74,2%	79,4%
RNN (Pooling)	75,5%	79,8%
RNN (Pooling & GloVe)	84,9%	80,6%
Convolutional Neural Networks	90,4%	79,8%
Self-Attention CNN	86,4%	79,8%
Quasi-Recurrent Neural Networks	83,5%	81,2%
Self-Attention QRNN	83,6%	81,4%
BERT	83,0%	84,1%

BERT achieved the highest validation accuracy.

Test set accuracy

83,02%

Conclusions

- Different models with similar results (intrinsic difficulty of the problem? Dirty dataset?)
- **BERT** achieved better results, but it's training was very slow (~4h per epoch)...

Future works

- Limit pre-processing combinations
- Continue testing BERT and concurrent models (like RoBERTa)

Thanks!