Natural Language Processing with Disaster Tweets

Justinas Jučas - Delft University of Technology
William Yihao Zhang - Technical University Berlin
Gausse Mael Dongmo Kenfack - INSA de Lyon
Matthias Diederichsen - University of British Columbia

Table of Contents

Problem: Classifying Tweets

Anthony Matthews
@Anthy865

Property losses from California wildfire nearly double as week-old blaze rages http://t.co/E0UUsnpsq5

6:00 PM · Sep 20, 2015

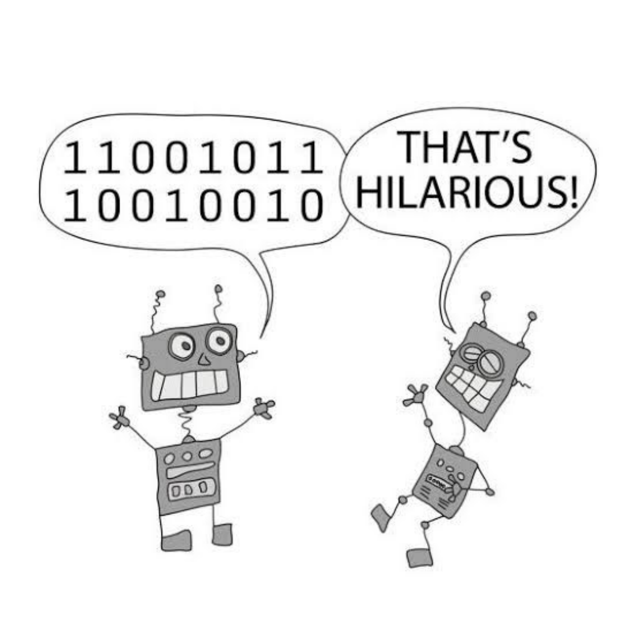
Jack
@justaguy33420

This bowl got me thinking... Damn I've been blazing for so damn long

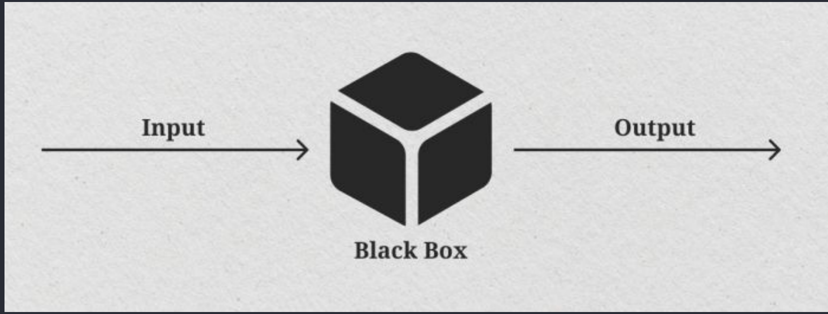
12:00 PM · Jun 1, 2021

How do we approach this?

Natural Language Processing

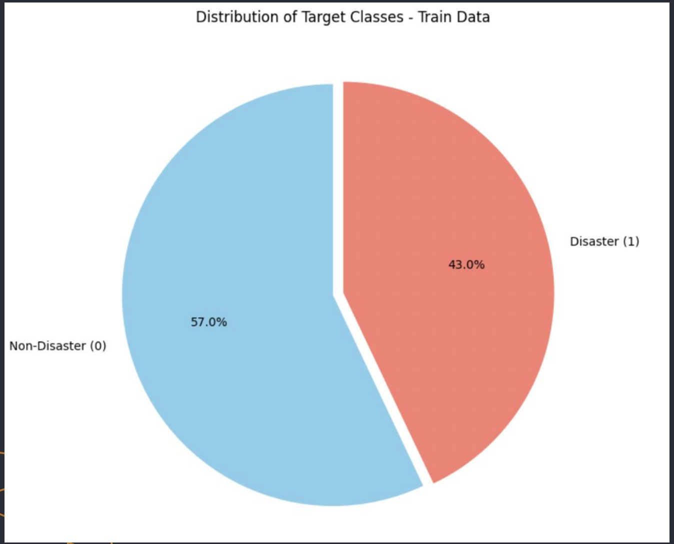


What we don't want:



Dataset

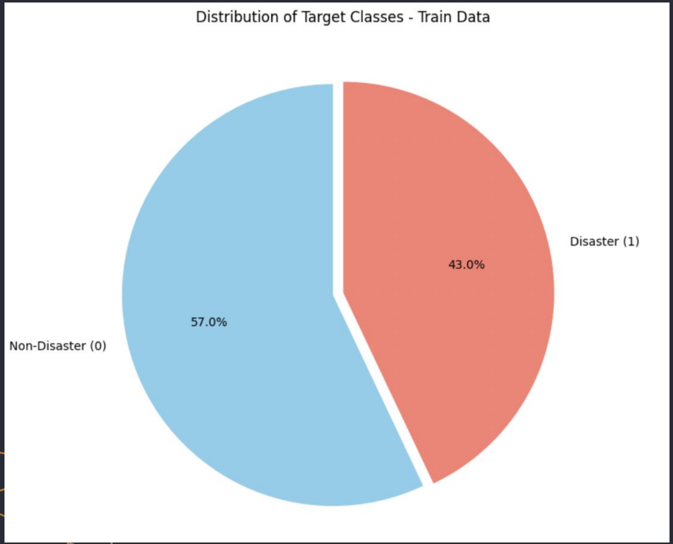
Train: ~7.6k tweets
Test: ~3.2k tweets



id	text	location	keyword	target
1	[...] LOOK AT THE SKY LAST NIGHT IT WAS ABLAZE [....]	"London, UK"	ablaze	0
2	Twelve feared killed in [...] crash	N/A	ambulance	1

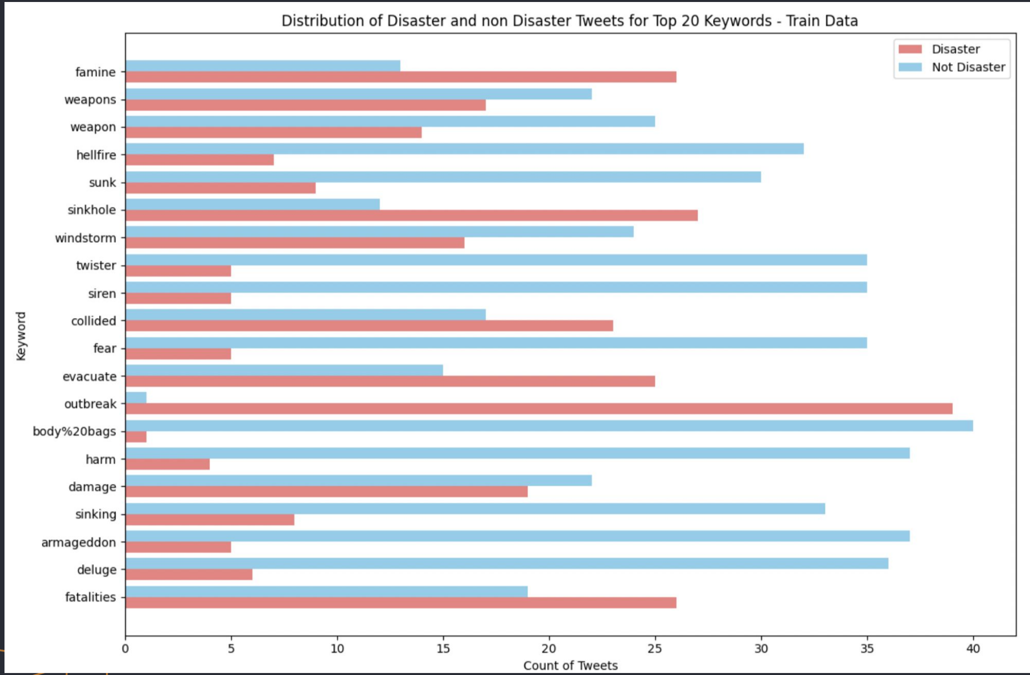
Dataset

Train: ~7.6k tweets
Test: ~3.2k tweets



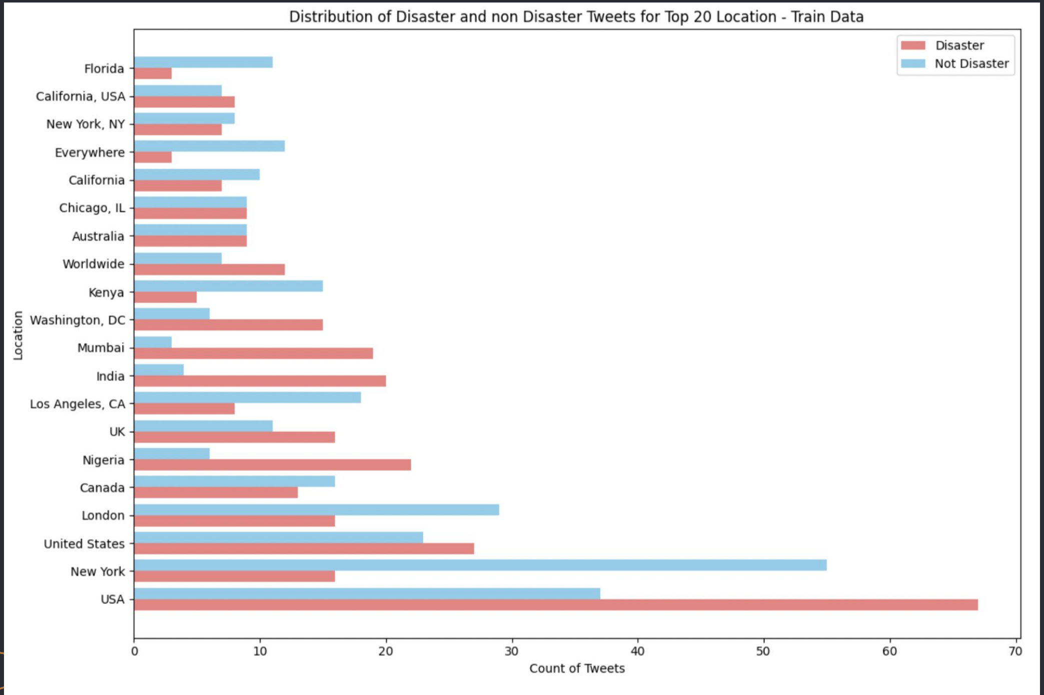
Null Values	Keyword	Location
Train	0.80%	33.27%
Test	0.79%	33.86%

Data Exploration



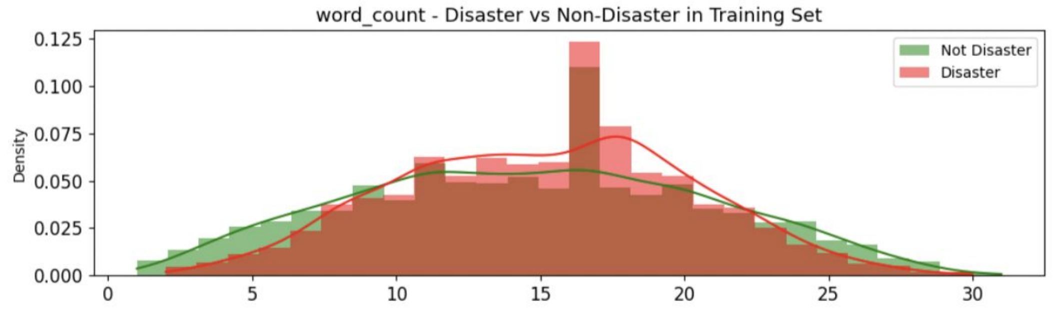
Certain keywords heavily skew towards non disaster or disaster tweets

Data Exploration



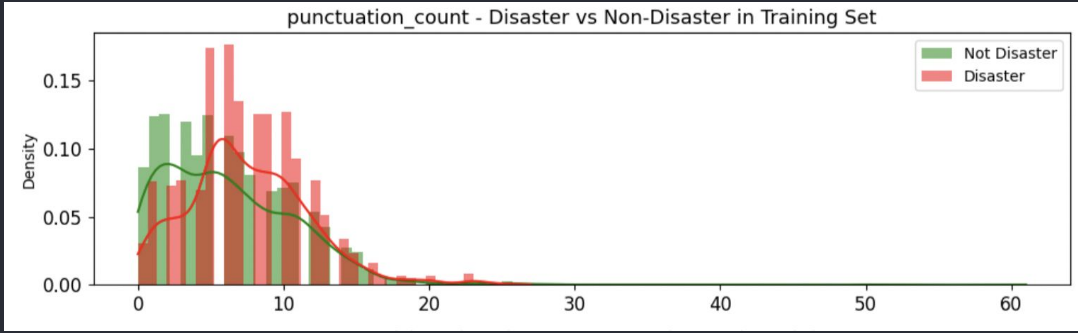
More balanced proportion for the tweet's Location

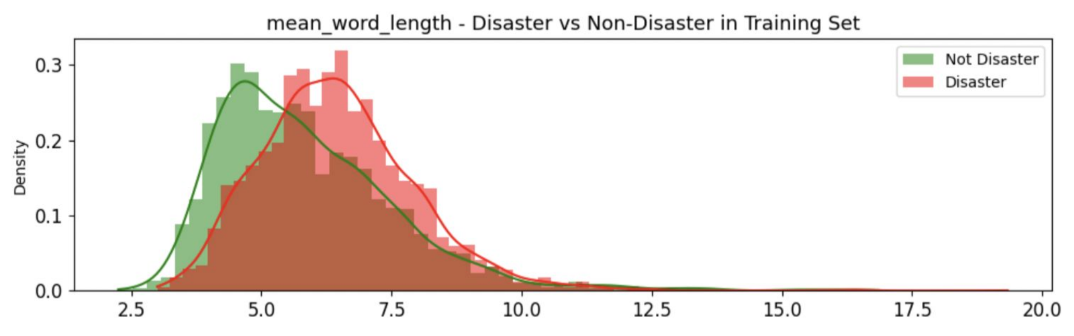
Data Exploration



Distribution of target class for engineered features

Data Exploration



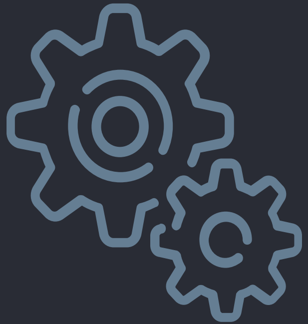


Observation:

Some engineered features have unique distributions

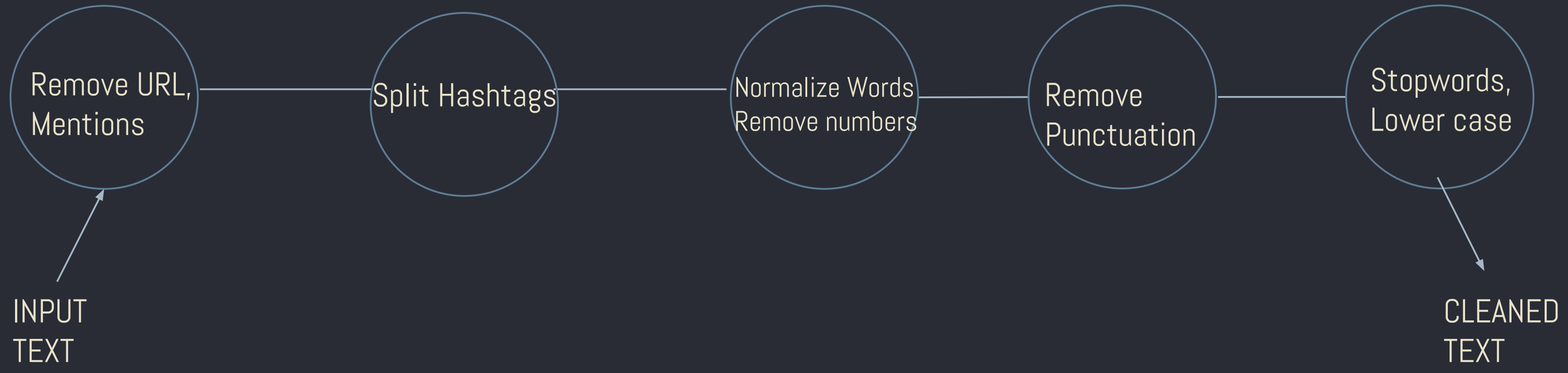
Data Preprocessing

Approach 1:
Custom preprocessing through regex

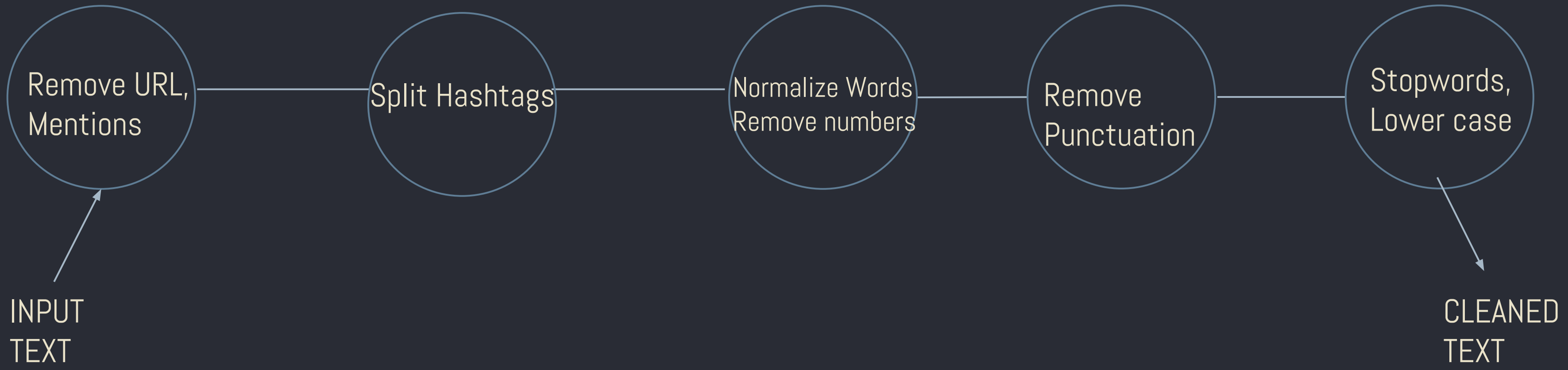


Approach 2:
Utilize open-source library SpaCy

Data Preprocessing

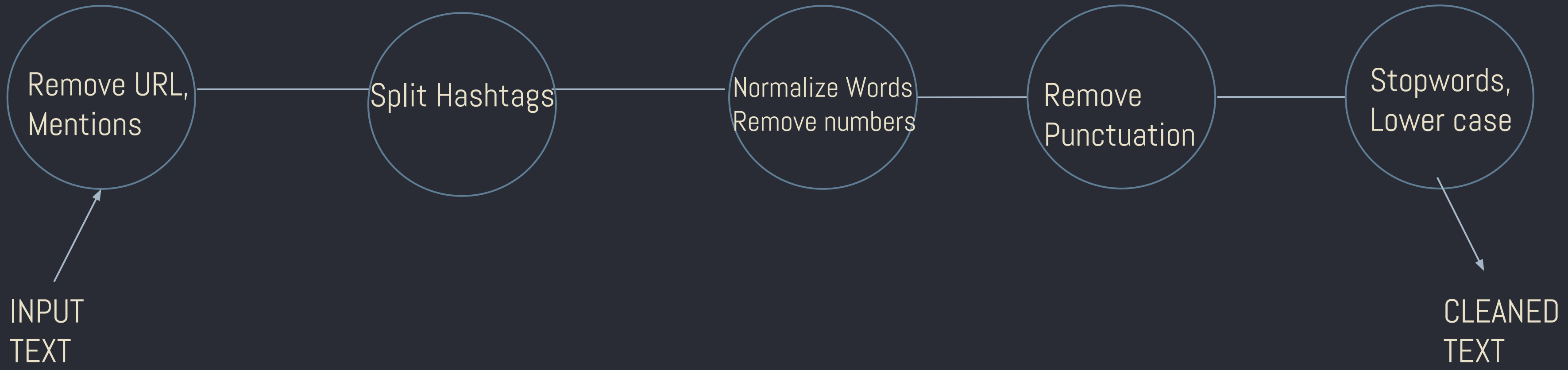


Data Preprocessing



#saudiarabia 13 confirmed dead as suicide bomber attacks Saudi Arabian mosque
-
The l... http://t.co/LwAnE9vupg - http://t.co/CpQguFZB28

Data Preprocessing

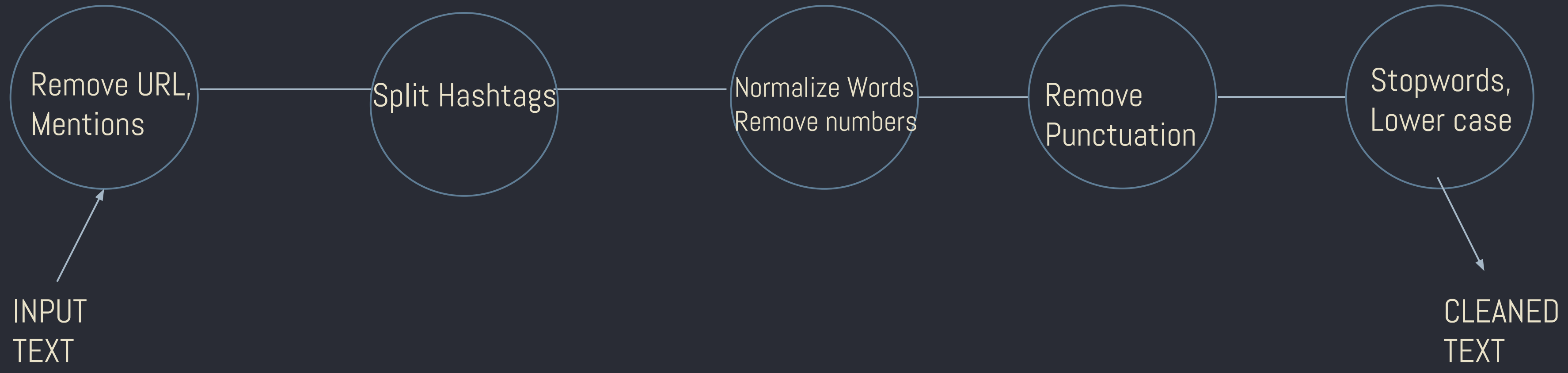


#saudiarabia 13 confirmed dead as suicide bomber attacks Saudi Arabian mosque

-

The l... ~~http://t.co/LwAnE9vupg~~ - ~~http://t.co/CpQguFZB28~~

Data Preprocessing

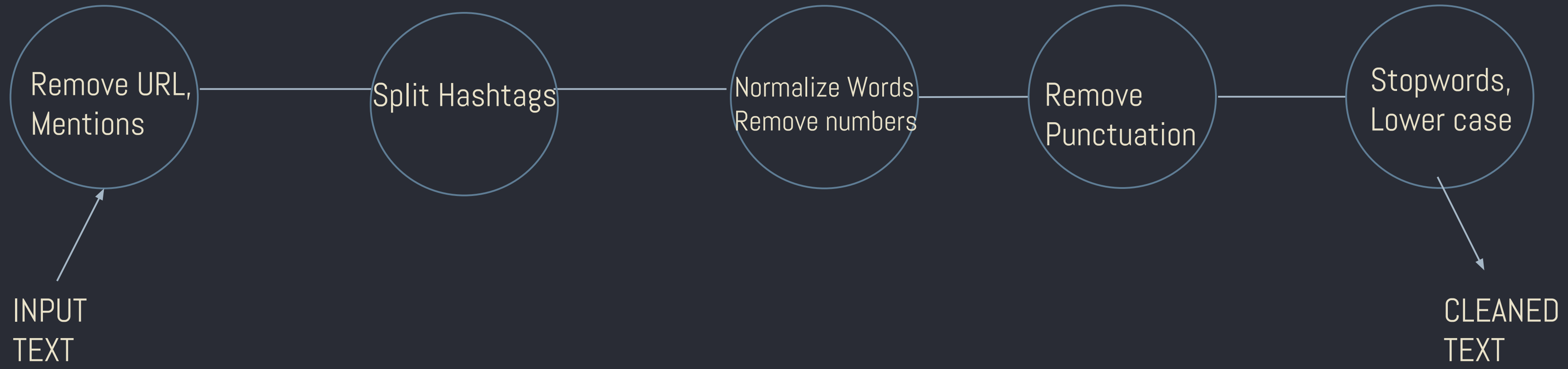


#saudiarabia 13 confirmed dead as suicide bomber attacks Saudi Arabian mosque

-

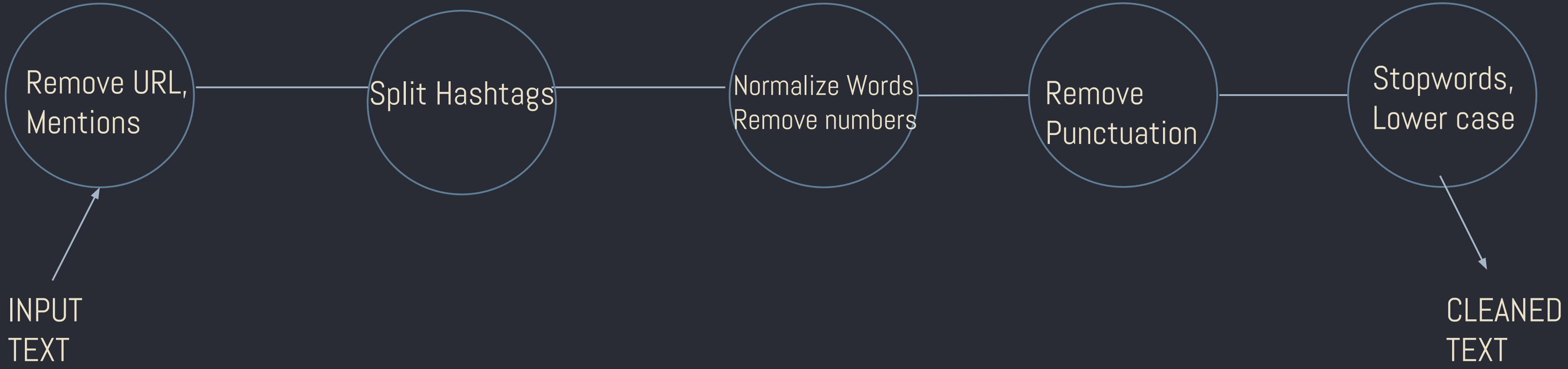
The I... -

Data Preprocessing



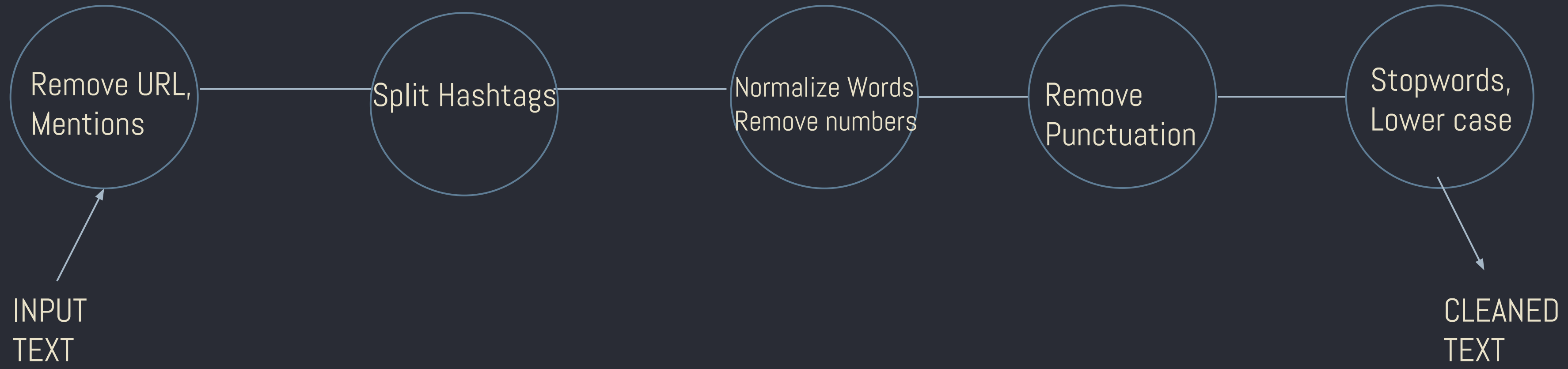
Saudi arabia ~~13~~ confirmed dead as suicide bomber attacks Saudi Arabian mosque - The I... -

Data Preprocessing



Saudi arabia confirmed dead as suicide bomber attacks Saudi Arabian mosque - The I... -

Data Preprocessing



Saudi arabia confirmed dead ~~as~~ suicide bomber attacks Saudi Arabian mosque
~~The I~~

Approach I: LSTM

LSTM Model – GloVe Embeddings

GloVe = Global Vectors for Word Representations

Twitter GloVe:
- Introduce 2014 by Stanford NLP research group
- Trained on 2 Billion tweets containing 27 billion tokens
- Vocabulary of 1.2 Tokens
- Embeddings available: 25d, 50d, 100d and 200d

In our Dataset:
- 92.86% coverage of unique tokens

LSTM Model – Word2Vec Embeddings

word2vec-google-news-300:
- Created in 2013 at Google
- Trained on 100 billion words from Google News.
- 300-dimensional vectors
- Predicts context words from a target word (Skip-Gram) or predicst the target word from context words (Continuous Bag of Words).

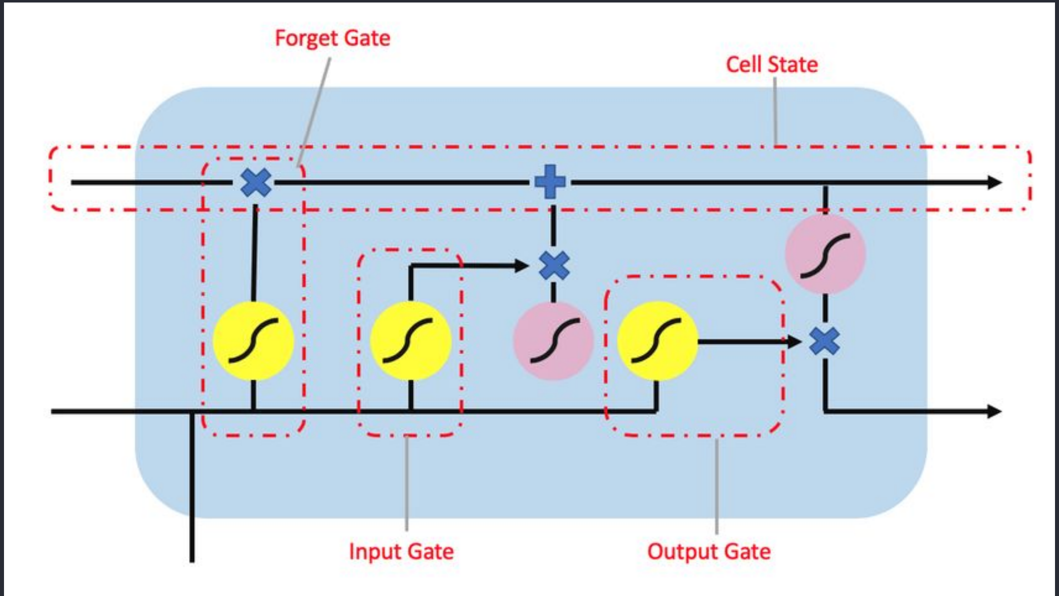
In our Dataset:
- 74.7% coverage of unique tokens

LSTM - Motivation

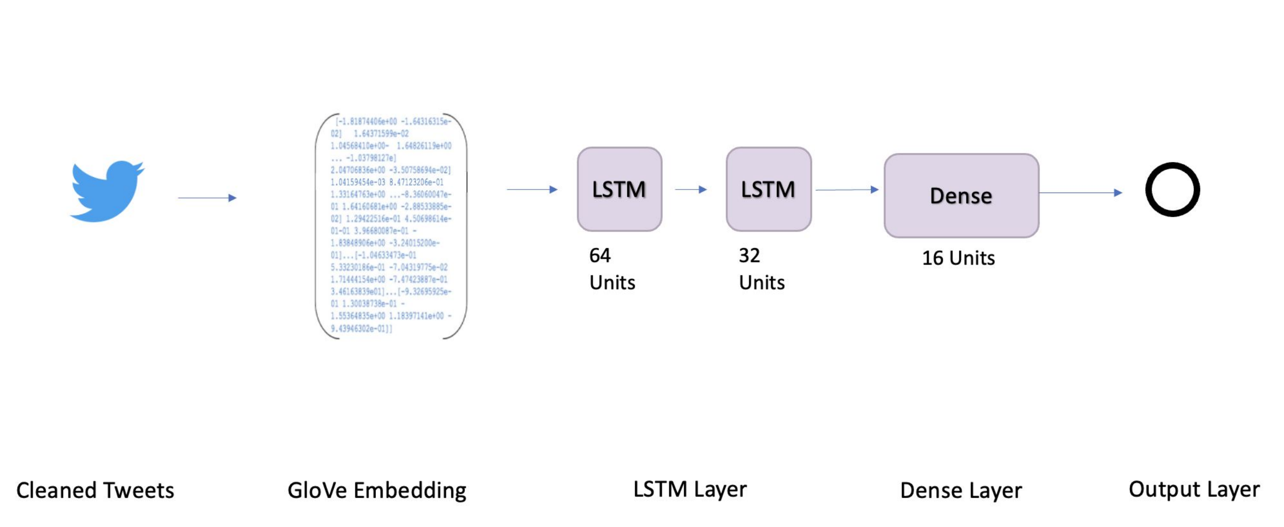
LSTMs are good at:

1. Handling sequential data and storing context
2. Dealing with Ambiguity
3. Sentiment Analysis

LSTM - Quick visualization

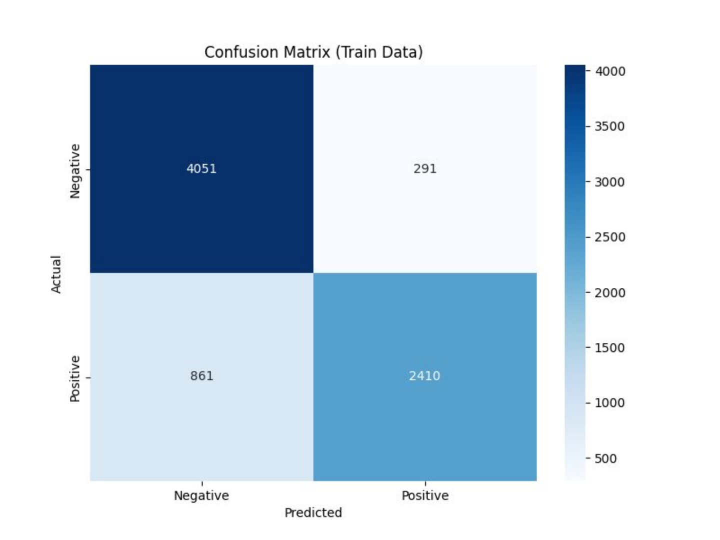


LSTM Model –GloVe Embeddings

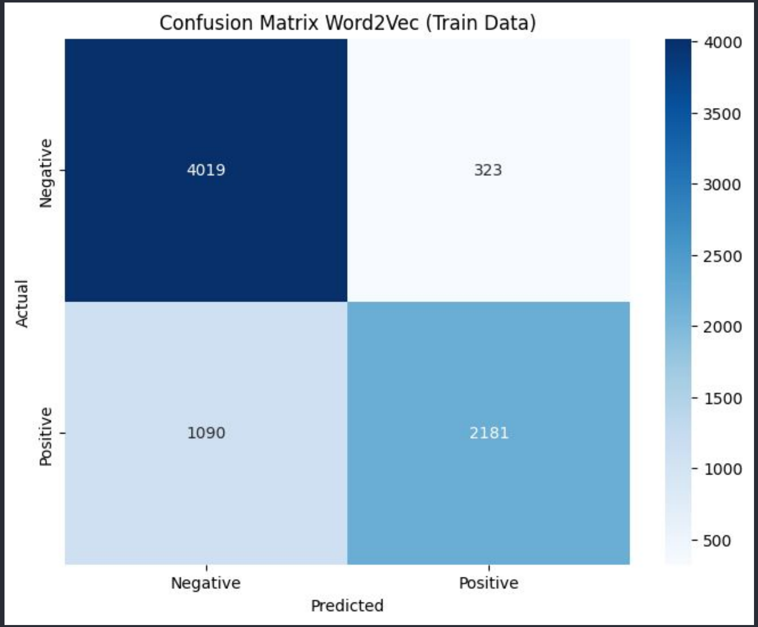


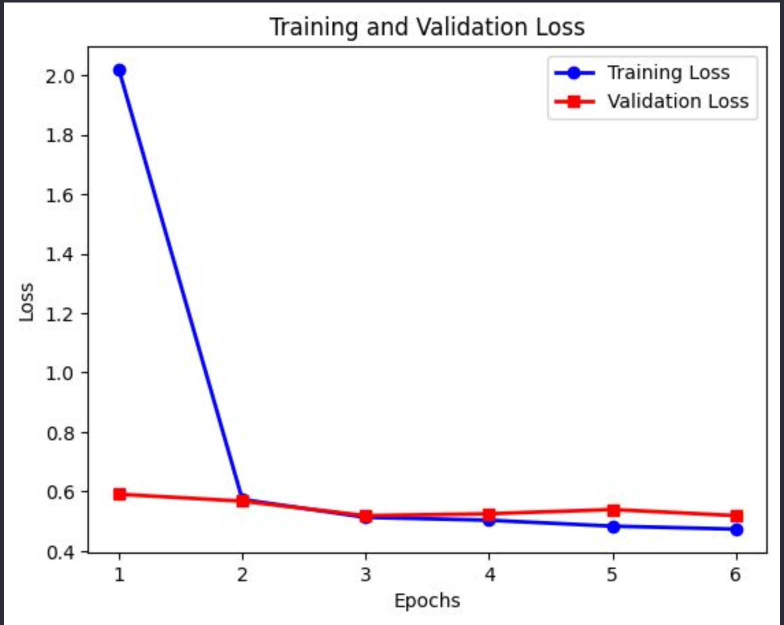
Test Accuracy: 80.69%

LSTM - Glove Embeddings Results



LSTM - Word2Vec Results

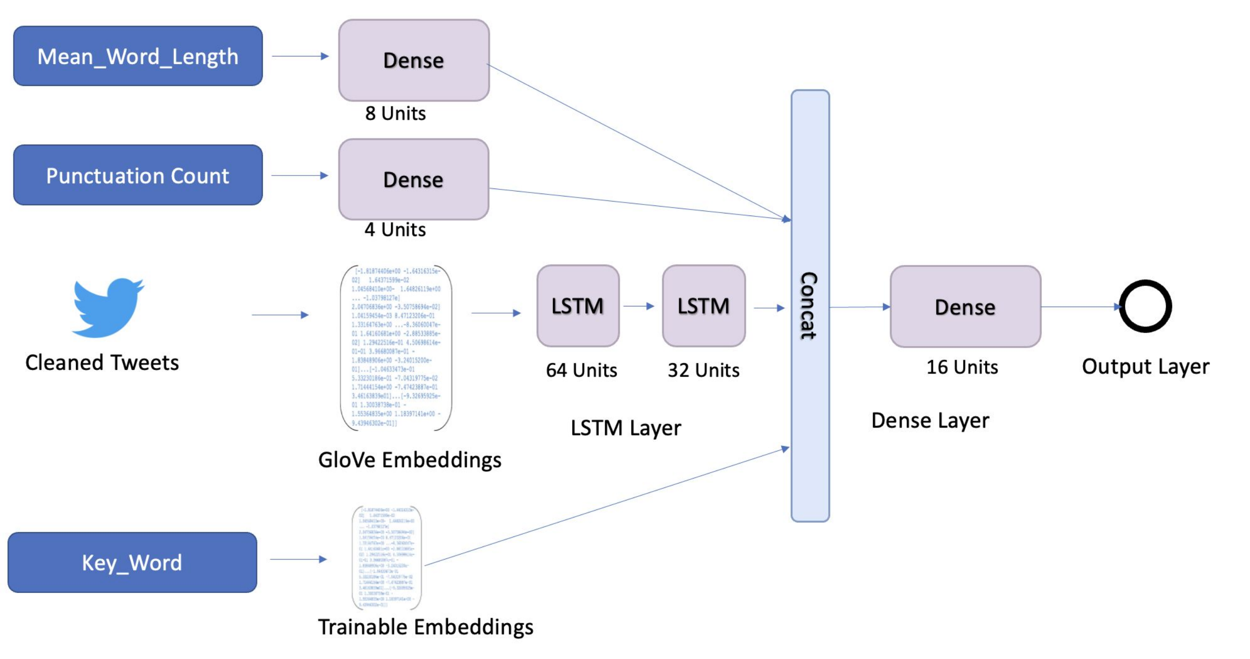




Final test accuracy:

Before modification: 77.43%　　*After* modification: 80.12%

LSTM Model - Modifications



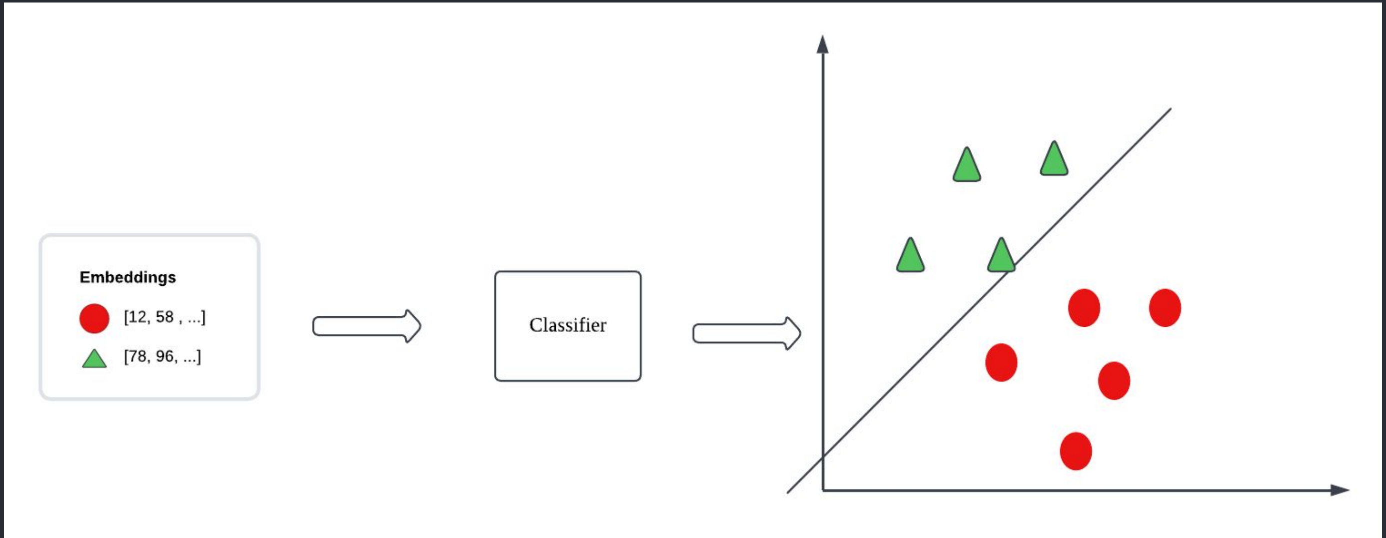
Idea:
Multi Input Model
With engineered Features

Test Accuracy: 80.44%

Approach II: **MLP with Embeddings**

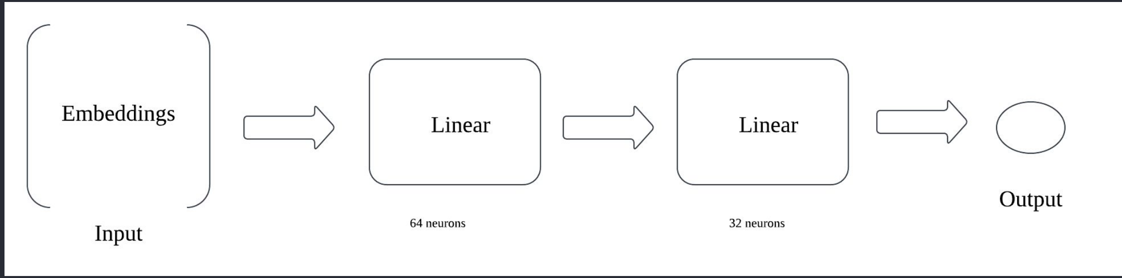
MLP with Embeddings

Let's try a simple classifier with the text embeddings as input



MLP with Embeddings

MLP architecture




```
MLPModel(
  (fc1): Linear(in_features=300, out_features=64, bias=True)
  (dropout): Dropout(p=0.3, inplace=False)
  (fc2): Linear(in_features=64, out_features=32, bias=True)
  (fc3): Linear(in_features=32, out_features=1, bias=True)
)
```

MLP with Embeddings

MLP architecture

- Embeddings from Spacy model “en_core_web_lg”
- Experimenting with available data
- Test Accuracy : 80.90 %

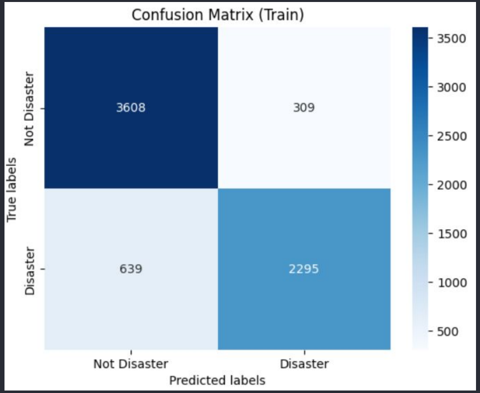
en_core_web_lg

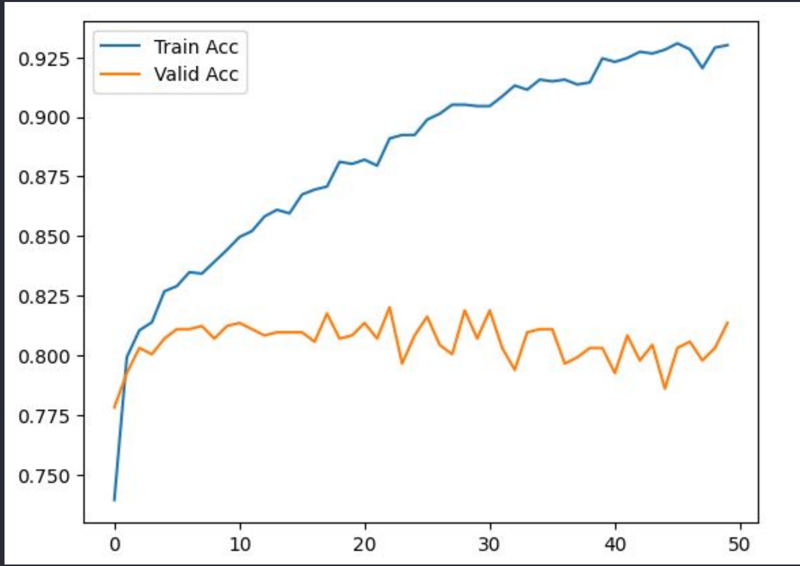
English pipeline optimized for CPU. Components: tok2vec, tagger, parser, senter, ner, attribute_ruler, lemr

LANGUAGE	EN English
TYPE	CORE Vocabulary, syntax, entities, vectors
GENRE	WEB written text (blogs, news, comments)
SIZE	LG 382 MB
COMPONENTS	tok2vec, tagger, parser, senter, attribute_ruler, lemmatizer, ner
PIPELINE	tok2vec, tagger, parser, attribute_ruler, lemmatizer, ner
VECTORS	685k keys, 343k unique vectors (300 dimensions)

MLP with Embeddings

- The Training process is fast
- The Model starts overfiting very quickly
- Not enough data

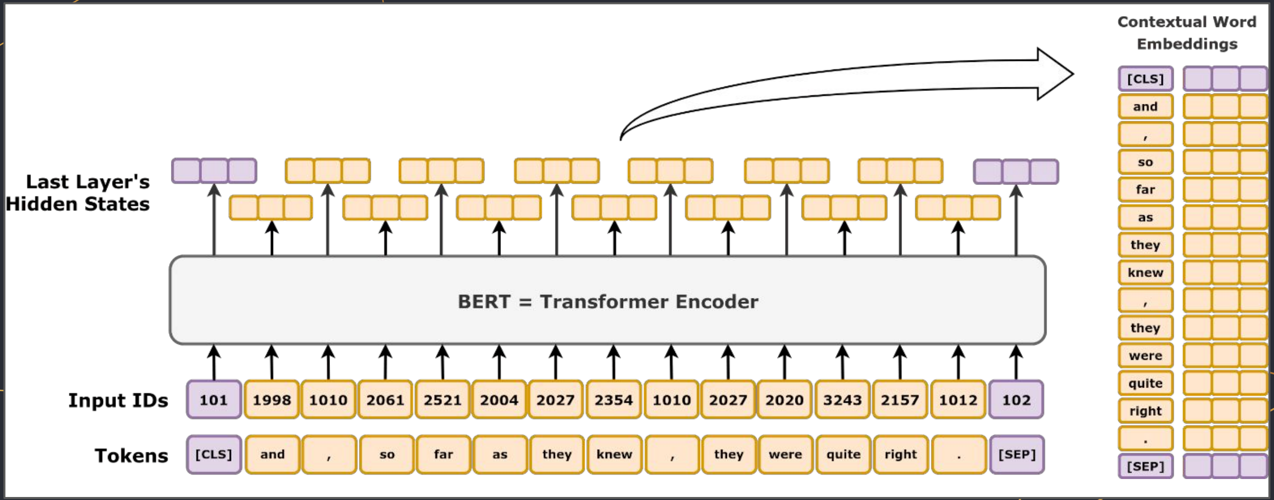




Approach III: **distil-BERT**

distil-BERT

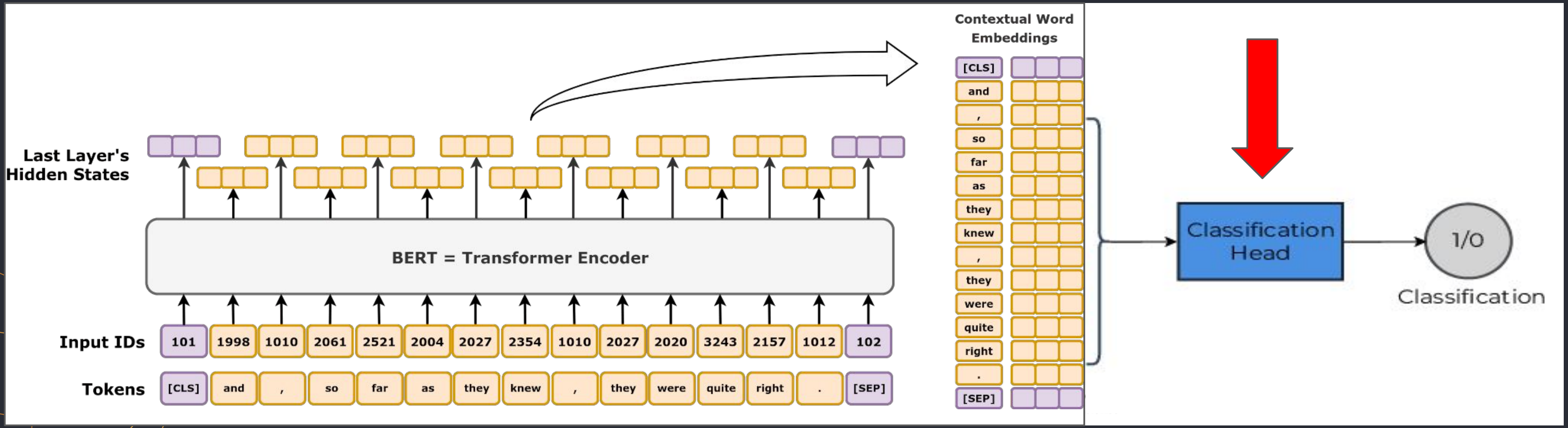
- **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
- Smaller, faster version of BERT
- 66M parameters
- Considered as baseline approach for NLP experiments



Idea I: classification-head training

Default classification head: 600k parameters

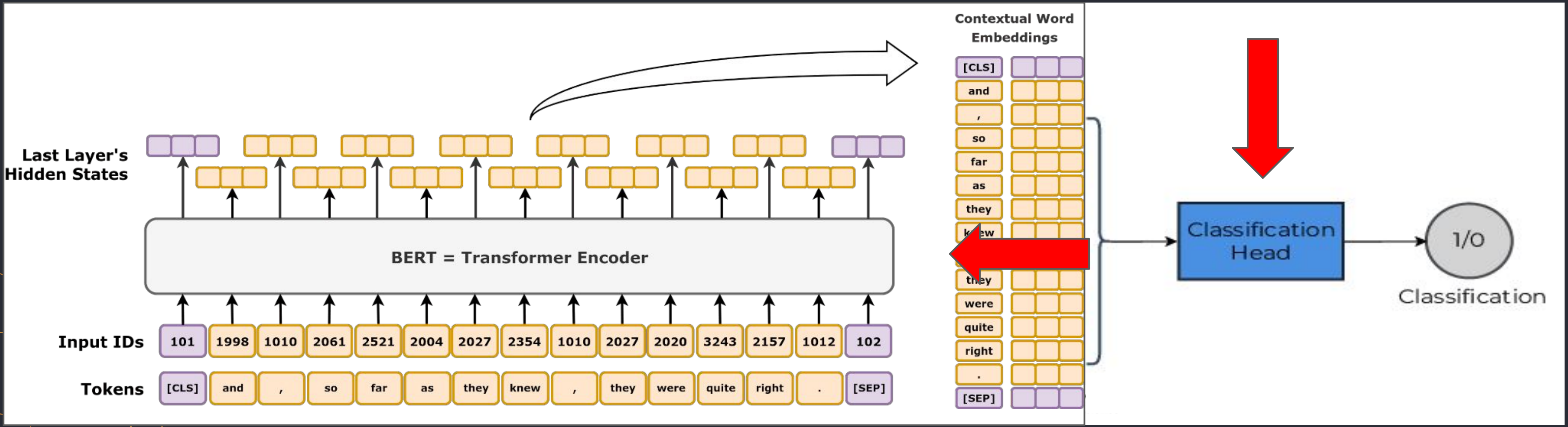
Final accuracy: **81.00%**



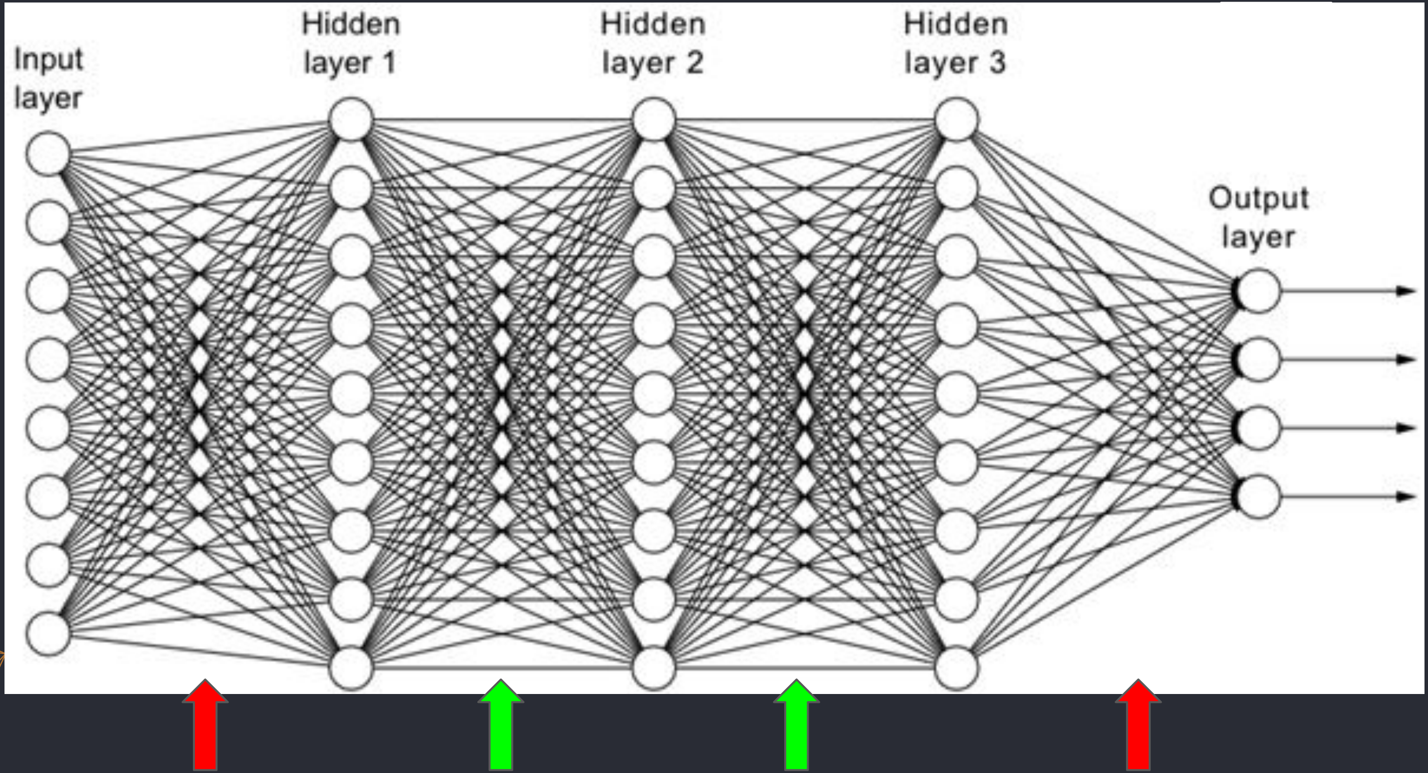
Idea II: fine-tuning transformers' weights

Retrain the whole model

Final accuracy: **80.91%**



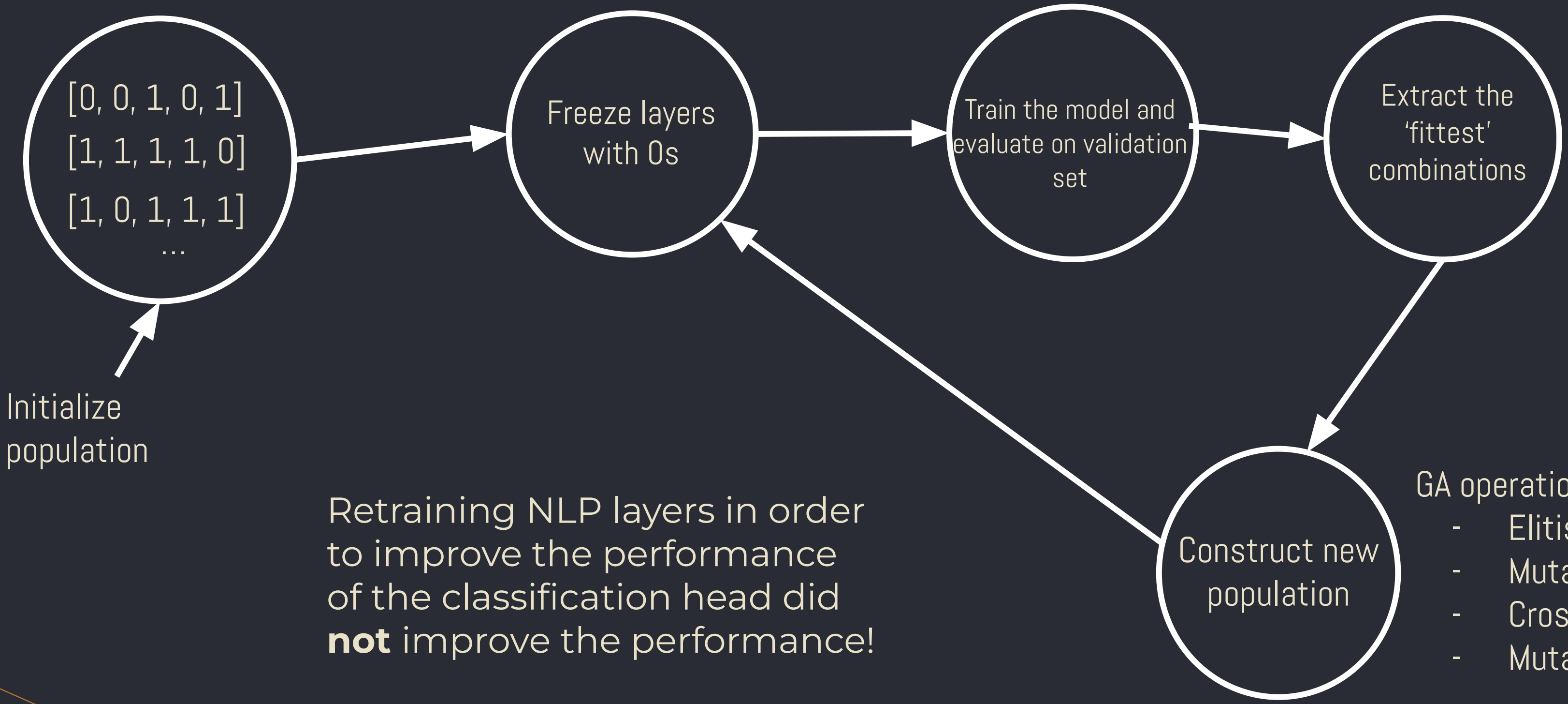
Idea III: finding components to (re)train



$2^{102} > 1000000000000000000000000000000$

But there is **exponentially** many different ways to train the model... What can we do?

Genetic Algorithm!



GA operations:
- Elitism
- Mutation
- Crossover
- Mutated Elitism

Retraining NLP layers in order to improve the performance of the classification head did **not** improve the performance!

Final accuracy: **80.94%**

Can we decrease the size of the model?

YES!

Neural Network Quantization

- Normally, all weights are encoded in float32 values!
- Models encoded in lower bit values take less storage and have higher inference!

Standard Quantization methods:

- Static quantization
- K-Means quantization
- GPTQ
- AQLM

class transformers.GPTQConfig <source>

```
( bits: int, tokenizer: typing.Any = None, dataset: typing.Union[typing.List[str], str, NoneType] =
None, group_size: int = 128, damp_percent: float = 0.1, desc_act: bool = False, sym: bool = True,
true_sequential: bool = True, use_cuda_fp16: bool = False, model_seqlen: typing.Optional[int] = None,
block_name_to_quantize: typing.Optional[str] = None, module_name_preceding_first_block:
typing.Optional[typing.List[str]] = None, batch_size: int = 1, pad_token_id: typing.Optional[int] =
None, use_exllama: typing.Optional[bool] = None, max_input_length: typing.Optional[int] = None,
exllama_config: typing.Optional[typing.Dict[str, typing.Any]] = None, cache_block_outputs: bool =
True, modules_in_block_to_quantize: typing.Optional[typing.List[typing.List[str]]] = None, **kwargs )
```

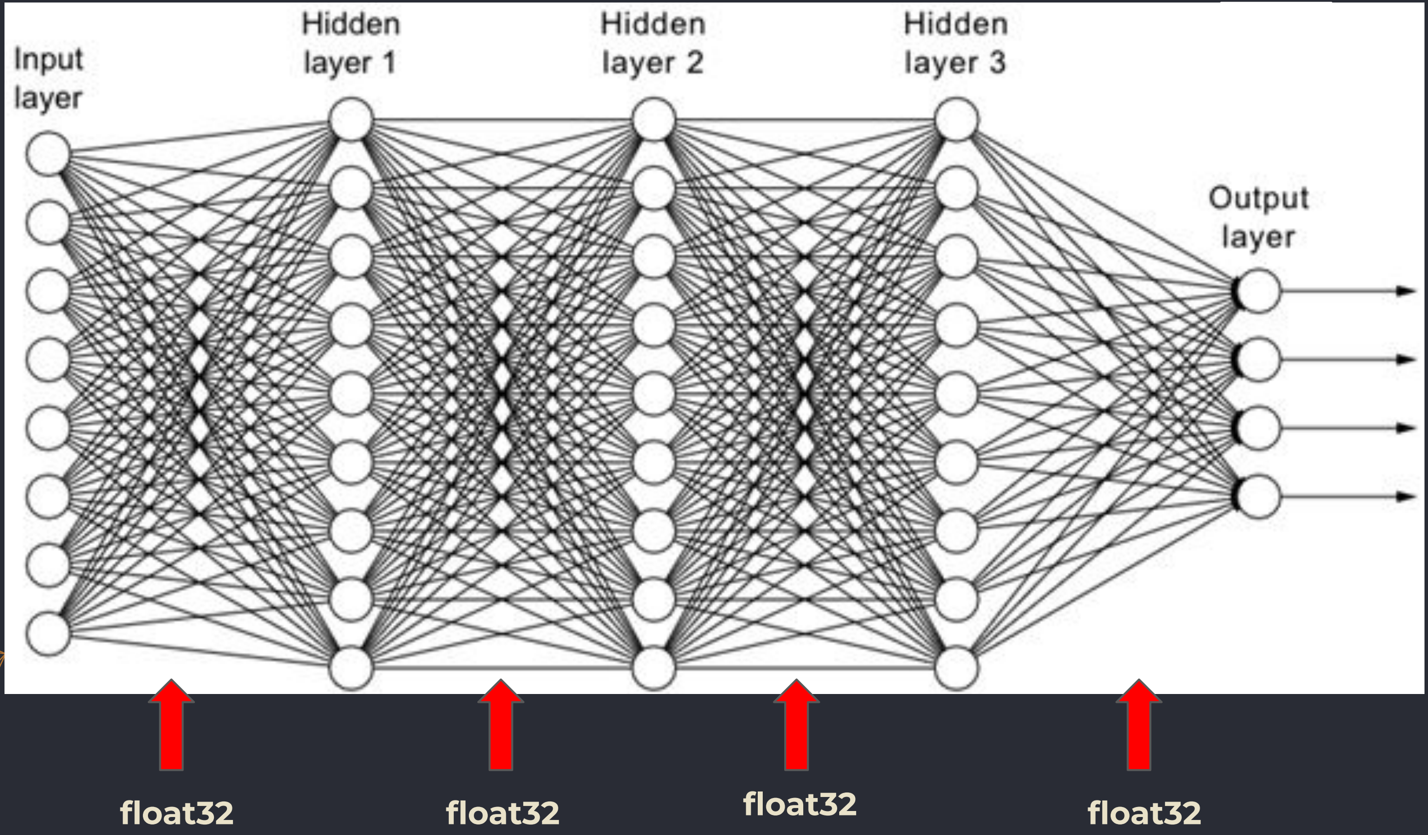
Boring!

Idea: find optimal combination of uniformly-quantized layers!

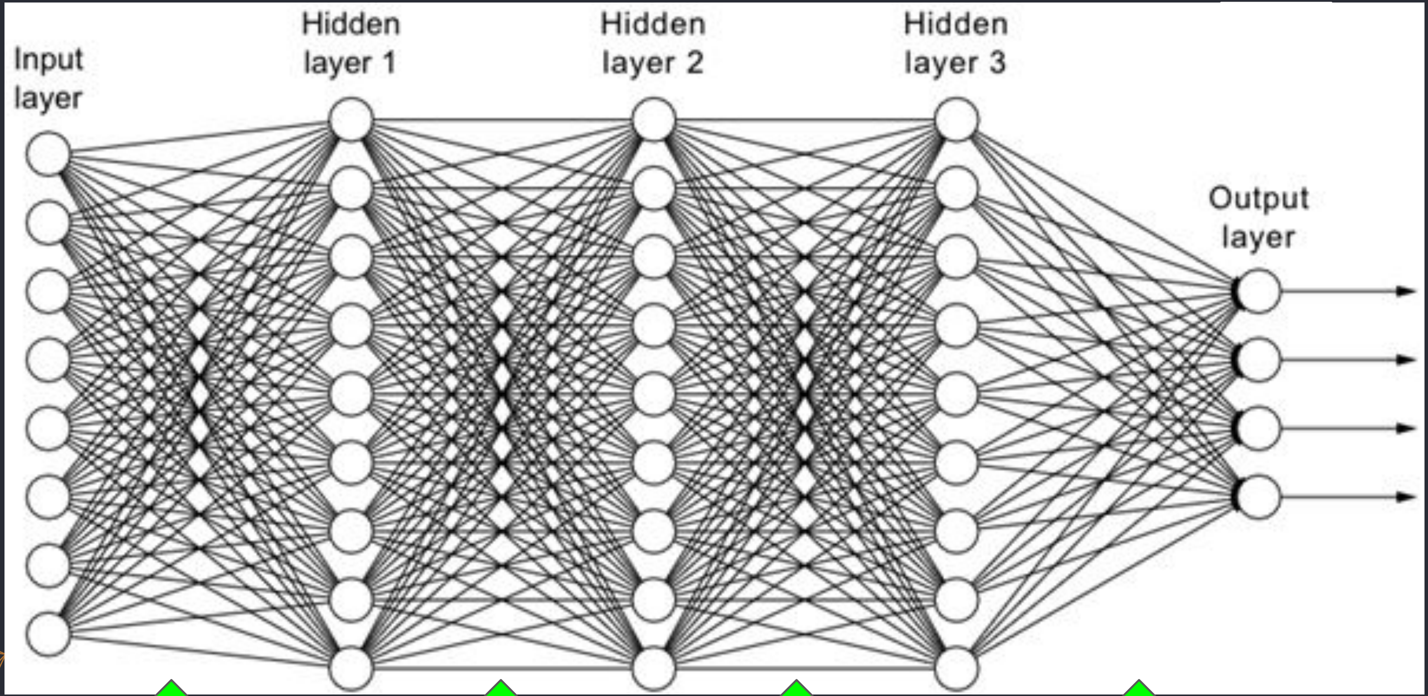
$$s = \frac{max - min}{2^b - 1}$$

$$q = \mathrm{round}(x/s) + z$$

Idea III: finding optimal layers to (re)train

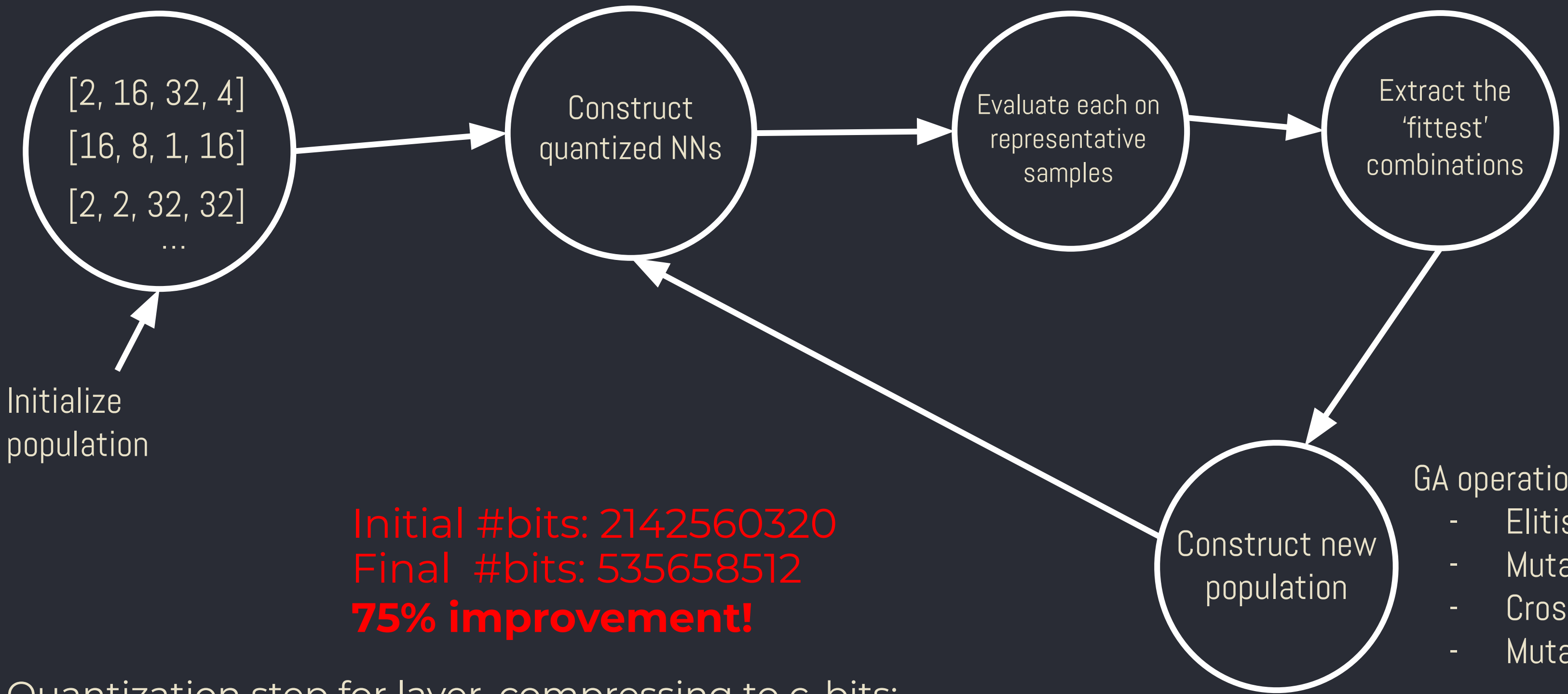


Idea III: finding optimal layers to (re)train



int2 int32 int4 int8

$$f(\mathrm{X}) = \lambda_1 l_{\text{validation loss}}(\mathrm{X}) + \lambda_2 q_{\text{quantization score}} + \lambda_3(\max(0, c_{\text{min accuracy}} - f_{\text{validation mean accuracy}}(\mathrm{X})$$



Initial #bits: 2142560320
Final #bits: 535658512
75% improvement!

GA operations:
- Elitism
- Mutation
- Crossover
- Mutated Elitism

Quantization step for layer, compressing to c_i bits:
1. Find lowest absolute weight value in the layer w_{min}
2. Scale all weights by $1/w_{min}$ and cast to int.
3. Find the amount of bits needed to represent maximum absolute value of the resulting weights b_{max}
4. Bit-shift all weights by $\max(b_{max} - c_i, 0)$ to the right.

Final Results

Approach	Test Accuracy
LSTM+Glove	80.69 %
Multi-Input LSTM	80.44 %
MLP+Spacy	80.90%
Distil-BERT	81.00%

- All approaches resulted in similar accuracies
- Distil-BERT likely performed better due to its higher complexity
- Distil-BERT could be compressed to 25% of its size

Summary

- Data Exploration and Processing
- 3 Approaches
- Future Research Directions
- Potential Applications

Workload

25% each :)

-

Q&A