Data Challenge Solvay

Results & Award Ceremony

Bisnode Group Analytics

Bisnode

The Challenge

Bisnode

Sentiment Analysis

- **85,700 reviews in the training set**
  - Balanced
  - Variable length
  - 85,409 not empty
- **29,140 reviews in the test set**
  - 19073 actually used for evaluation
  - Representative of the training set
- **Metric: accuracy**

Bisnode

General Remarks

Bisnode

Only import what you need

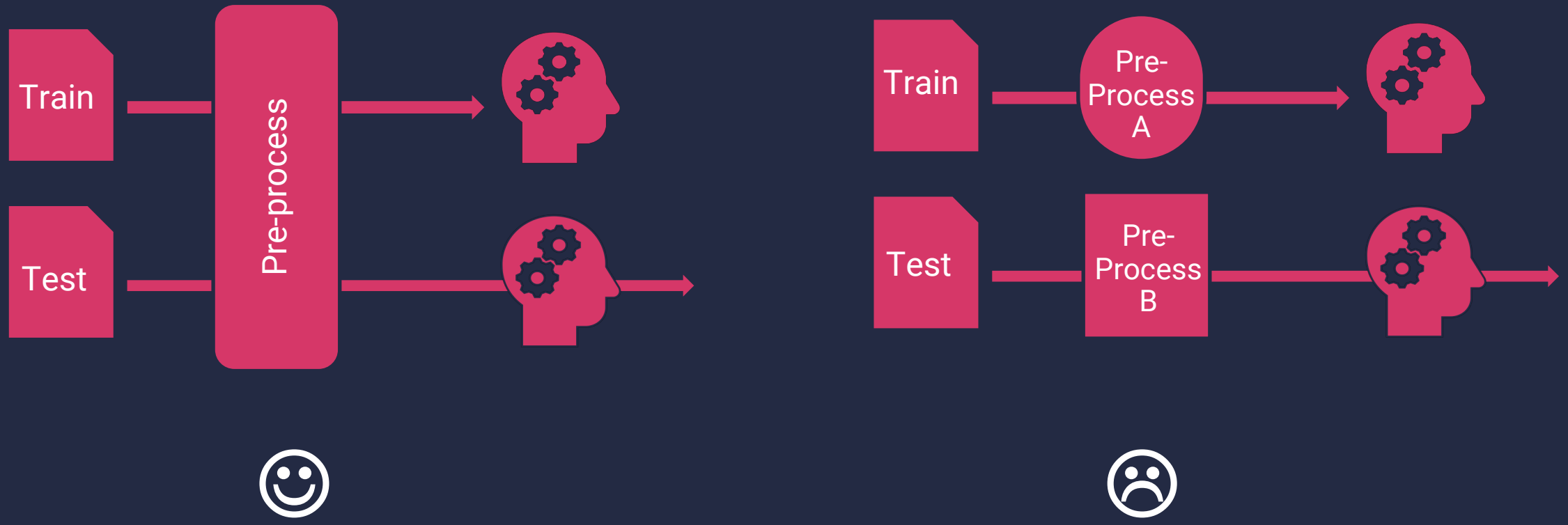
```
import nltk

nltk.download('brown')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_treebank_pos_tagger')
nltk.download('punkt')
nltk.download('tagsets')
nltk.download('vader_lexicon')
```

Needed ?

Bisnode

Same pre-processing for train/test sets



Bisnode

Beware of the output format

```
import json
from pprint import pprint
from random import random


# Load json into a Python list of dictionaries
test_set = json.loads(open("test_set_no_score_26-2-2019.json").read())
print type(test_set)
print len(test_set)
pprint(test_set[:2])

# Compute the score of each record
test_set_result = []
for record in test_set[:10]:
    record_result = {
        '_id': record['_id'],
        'score': int(round(random()))  # not the most reliable prediction. Make sure it is 0 or 1.
    }
    test_set_result.append(record_result)

# Write your results to a file
with open('group_name_results.json', 'w') as f:
    f.write(json.dumps(test_set_result))`
```

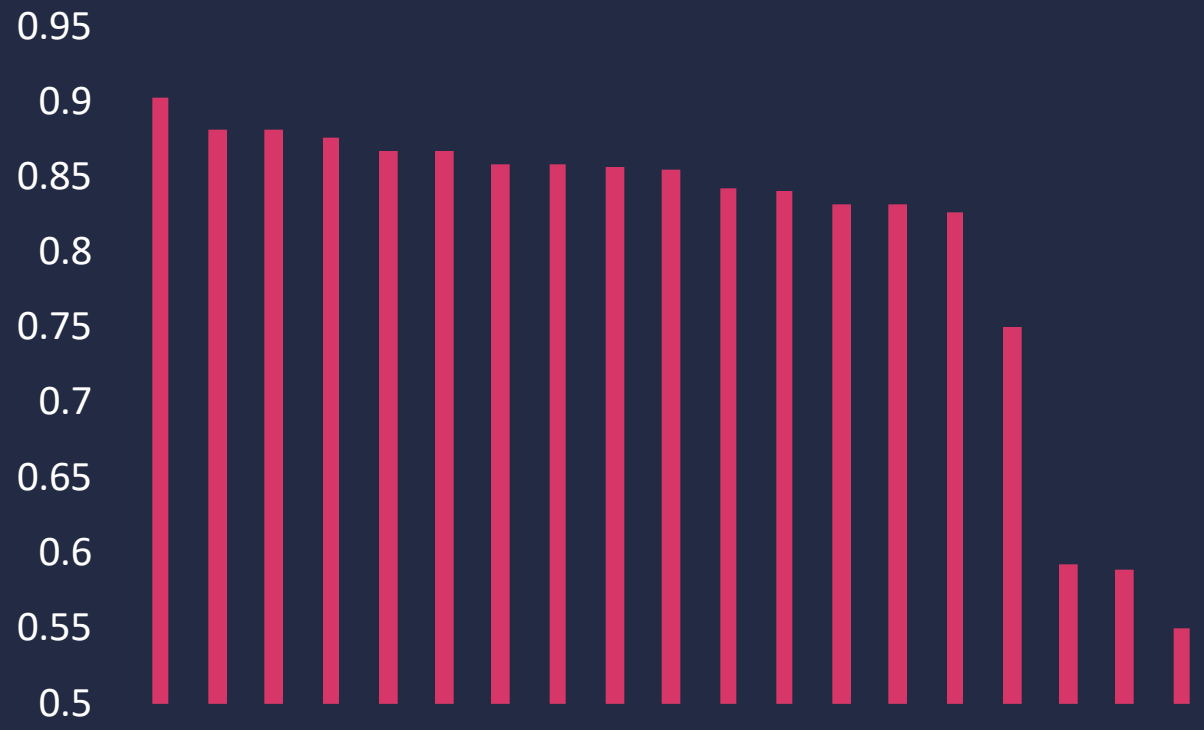
Bisnode

The Results

Bisnode

The Results

- **53** Participating teams
- **18** Valid results received
- **14** Above 80% accuracy

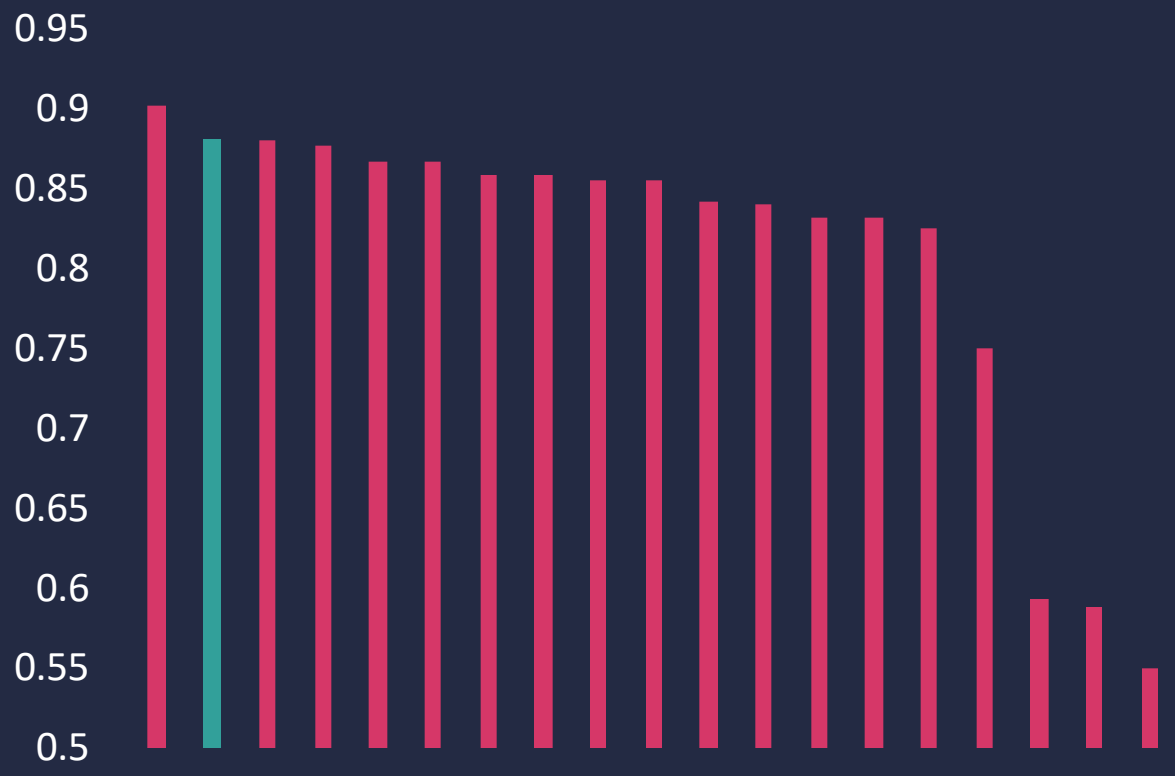


Our Approach

Bisnode

Bisnode – 88.10%

- **Cleaning**
  - Unidecode, remove rare words (<5)
- **Google embeddings**
  - 300 dimensions
  - 0-vector for "UNK" & "padding" vectors
  - Trainable
- **Bidirectional GRU**
  - Dropout for regularization
  - 1000 words per review

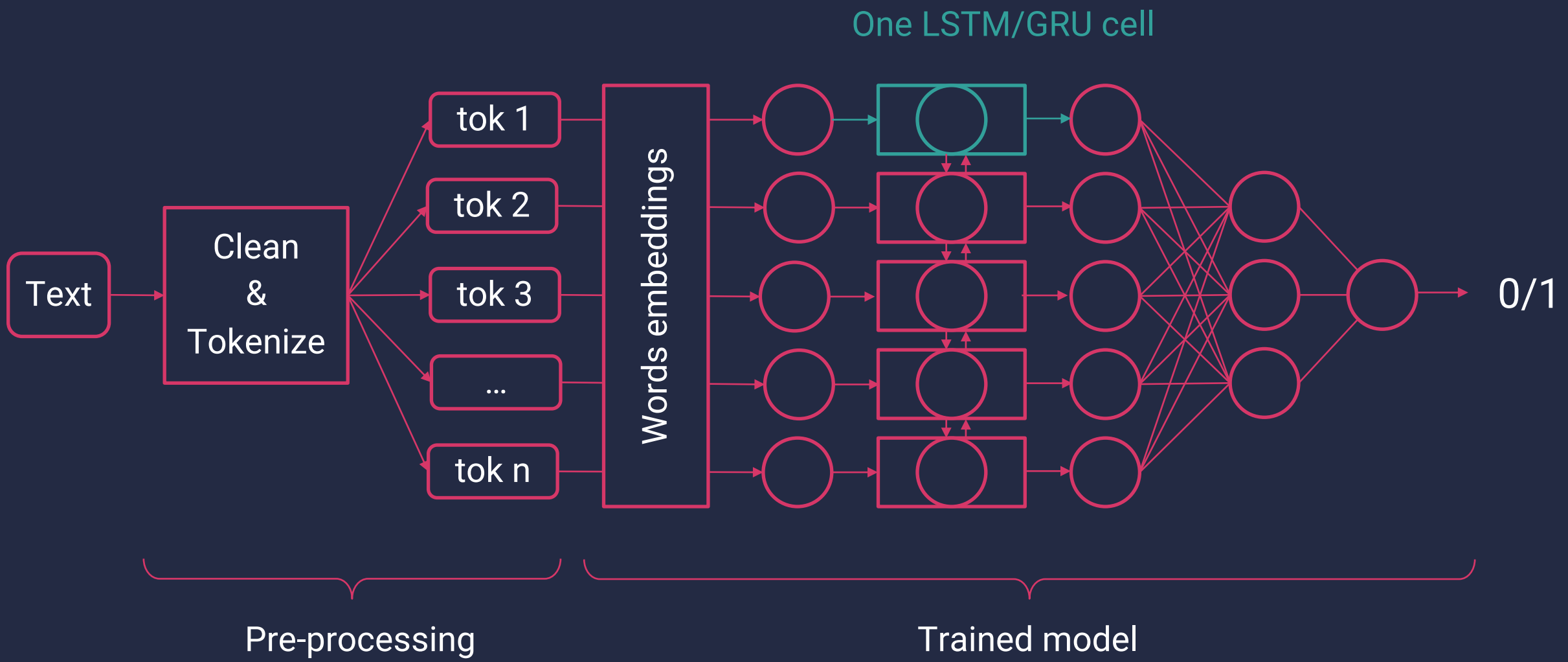


Main code

```
input = Input(shape=(max_review_length,))
x = Embedding(vocab_size, embedding_size, input_length=max_review_length,
    weights=[embedding_matrix], trainable=True)(input)
x = SpatialDropout1D(0.5)(x)
x = Bidirectional(GRU(100, return_sequences=True))(x)
x = Conv1D(64, kernel_size=2, padding='valid', kernel_initializer='he_uniform')(x)
avg_pool = GlobalAveragePooling1D()(x)
max_pool = GlobalMaxPooling1D()(x)
conc = concatenate([avg_pool, max_pool])
output = Dense(1, activation="sigmoid")(conc)
model = Model(inputs=input, outputs=output)
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

https://www.tensorflow.org/tutorials/representation/word2vec
https://ahmedbesbes.com/overview-and-comparison-of-tranditional-and-deep-learning-models-in-text-classification.html
https://keras.io/layers/recurrent/

Model Structure



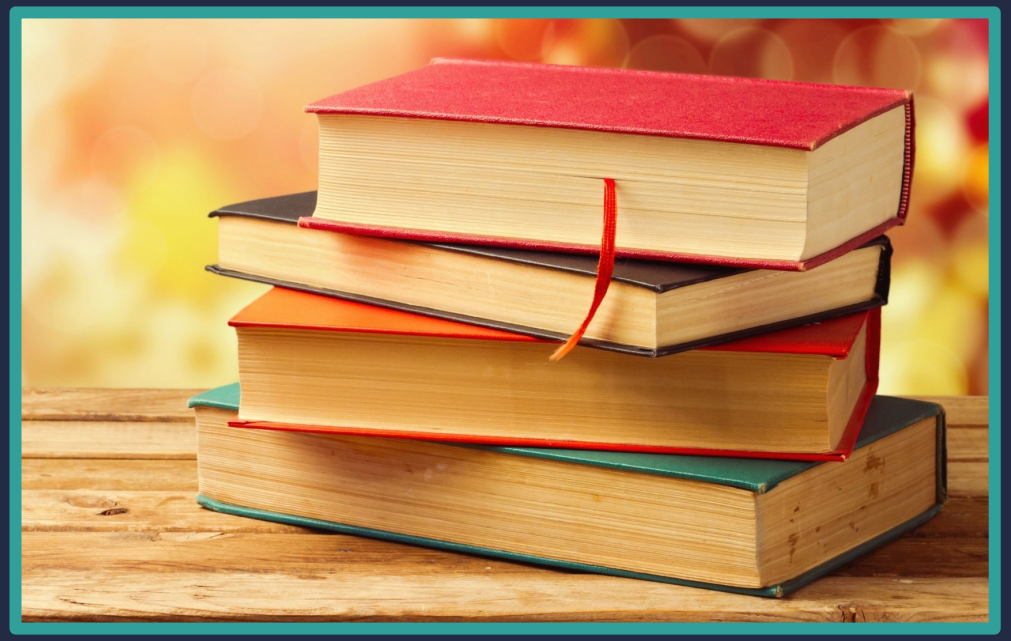
The Winners

Bisnode

BR – 87.67%

- **Remove frequent words (>0.7)**
- **TF-IDF**
  - Up to 3-grams
- **Linear SVC**

“Simple and efficient” ☺



MHB_Bogosort – 88.07%

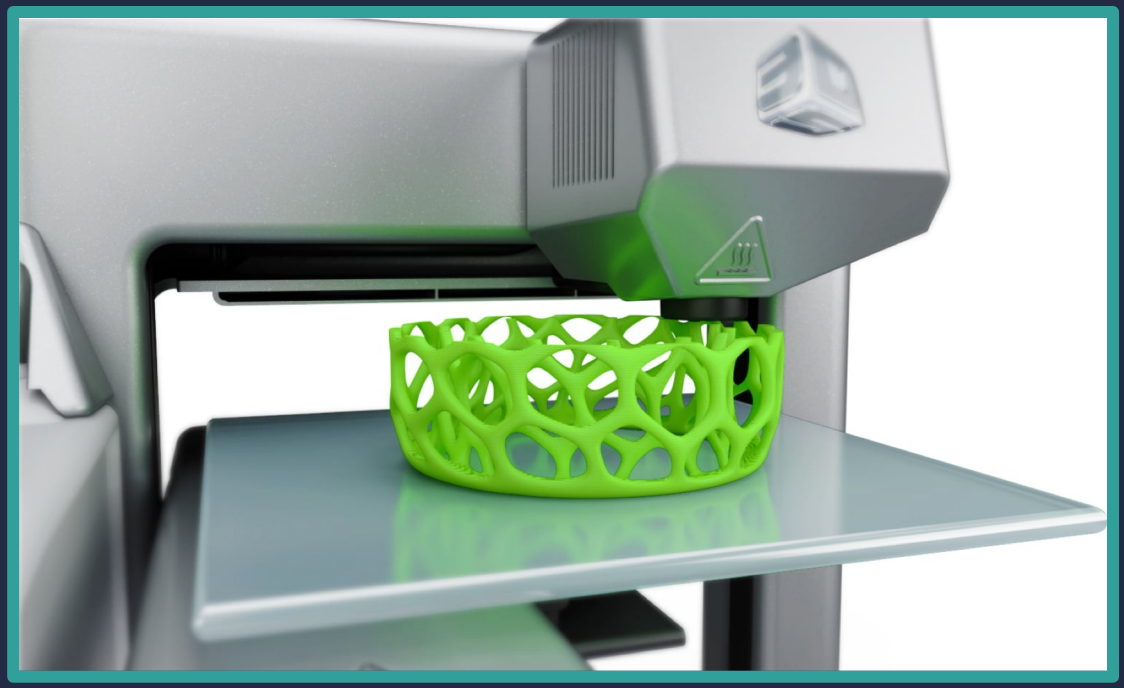
- **Cleaning**
  - Contractions equivalence
- **GloVe embeddings**
  - 300 + 2 dimensions
- **Bidirectional LSTM with dropout**
  - 300 words per review

“Deep Learning Masters” ☺

FastAi – 90.22%

- **FastAi: Open source deep learning library**
- **Tokenization, frequency filtering**
- **Pre-trained LSTM**
  - Regularization(dropout)
  - Fine-tuned with training set

“Smart is lazy” ☺



https://docs.fast.ai/text.html

Full Results

1	FastAi	0.902270225
-	Bisnode	0.881015221
2	MHB	0.880931159
3	BR	0.876736748
4	jm	0.867456614
5	TheAnachronicsLXI	0.867351754
6	NonTraditionnal	0.858910502
7	MALEX	0.858700781
8	infox	0.855869554
9	JustForFun	0.855187962
10	felixsimon	0.842447439
11	IronMan	0.840717244
12	Bxl+MST	0.832538143
13	Airbus	0.831699261
14	Laumi	0.826089236
15	Vranckx	0.750747129
16	TimOlivier	0.593089708
17	HappyKanguru	0.588475856
18	TheEmpathicReaders	0.550201856

The 5 best teams will be invited in our offices to discuss their methodology !

1. FastAi
2. MHB_Bogosort
3. BR
4. Jm
5. TheAnachronicsLXI

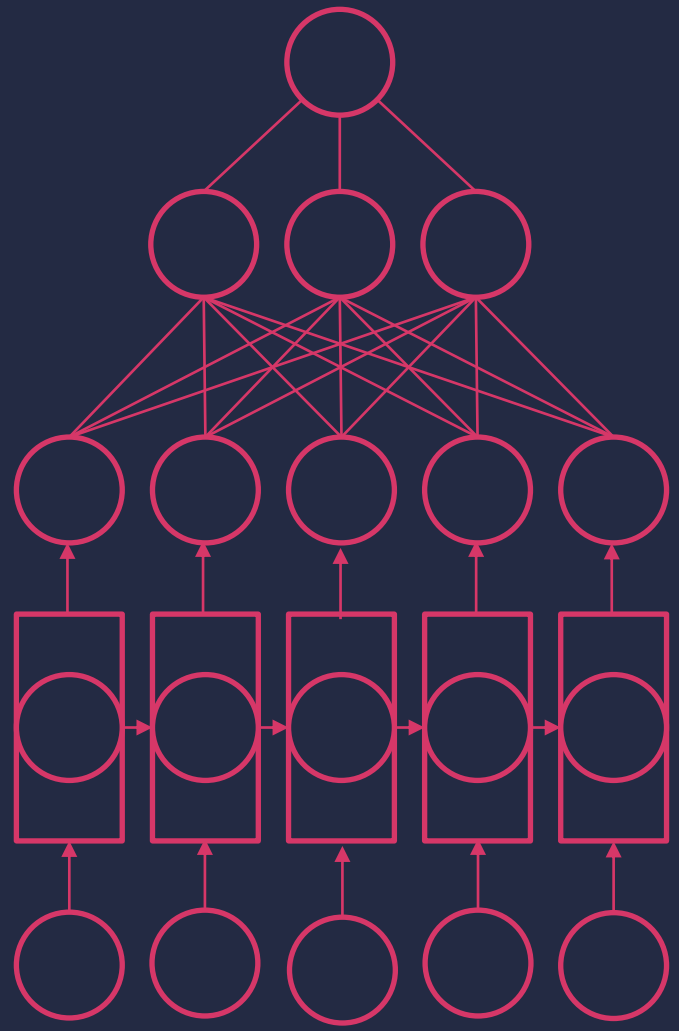
Take home message

Bisnode

Take home message

- **Deep Learning is powerful**
  - Semantic (embeddings)
  - Sentence structure (RNN)
  - Lots of tutorials & frameworks available

- **... But**
  - Trials and errors
  - Requires lots of data
  - Slow training
  - Sometimes, simpler models perform well too



Congratulations & Thank you !

Bisnode


Bisnode