Università di Pisa

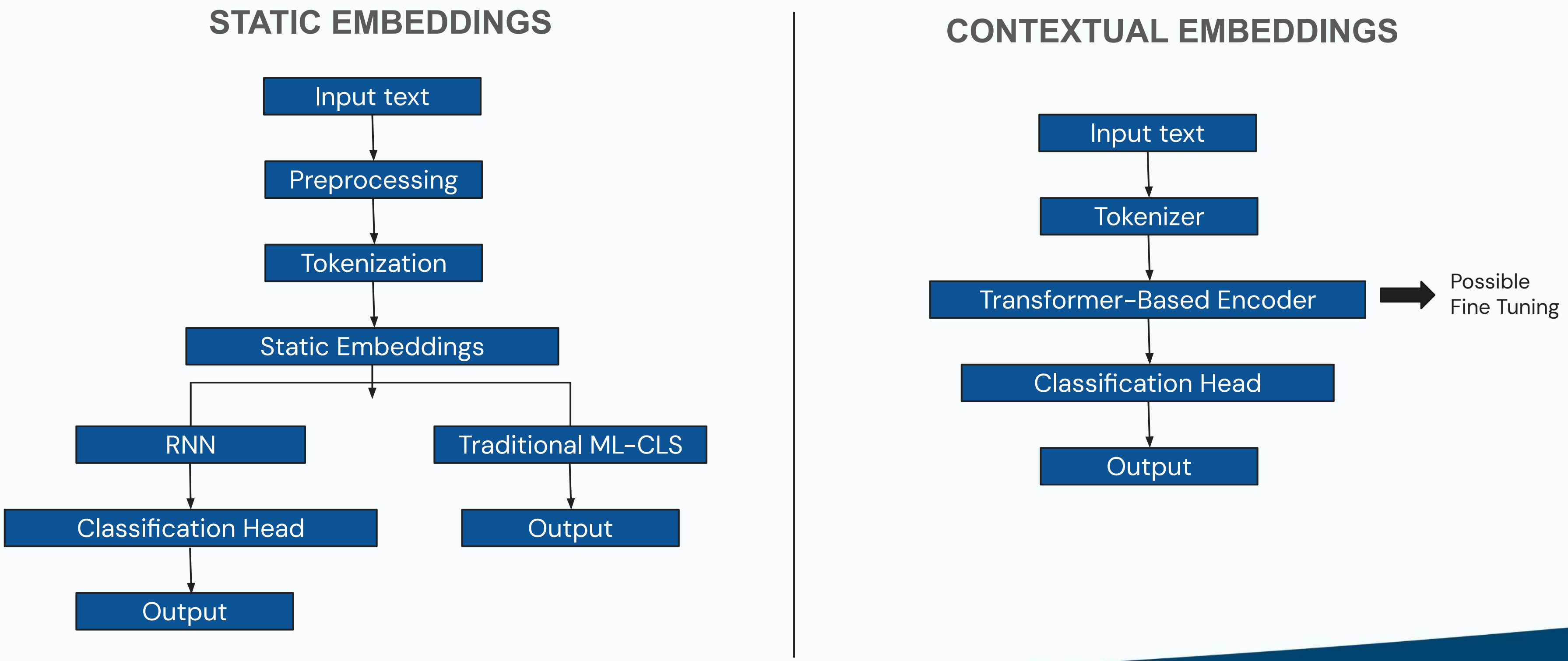
Sentiment Classification with Neural Text Representation on Amazon Reviews

19/05/2025

Mihnea Molnar, Gemma Ragadini, Andrea Lepori, Pan Zhang, Loris Giunta

- Group 4 -

Proposed approaches



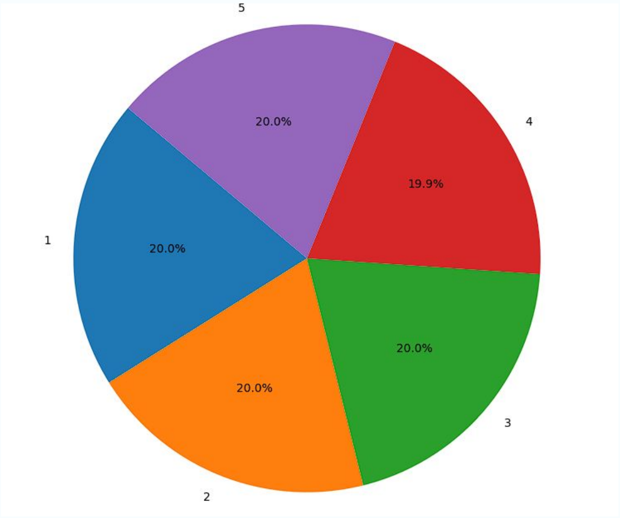
DATASET – ‘AMAZON REVIEW FULL SCORE DATASET’

ORIGINAL DATASET

	Rating	Title	Review
0	3	more like funchuck	Gave this to my dad for a gag gift after direc...
1	5	Inspiring	I hope a lot of people hear this cd. We need m...
2	5	The best soundtrack ever to anything.	I'm reading a lot of reviews saying that this ...
3	4	Chrono Cross OST	The music of Yasunori Misuda is without questi...
4	5	Too good to be true	Probably the greatest soundtrack in history! U...
5	5	There's a reason for the price	There's a reason this CD is so expensive, even...
6	1	Buyer beware	This is a self-published book, and if you want...
7	4	Errors, but great story	I was a dissapointed to see errors on the back...
8	1	The Worst!	A complete waste of time. Typographical errors...
9	1	Oh please	I guess you have to be a romance novel lover f...

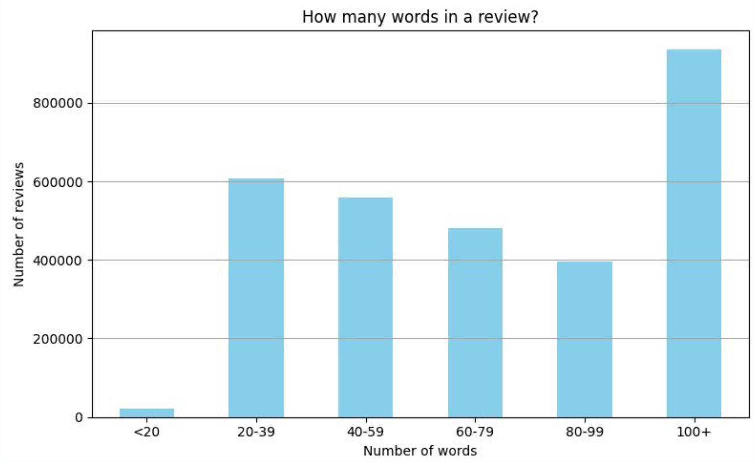
- 2999746 **ROWS**
- 3 **COLUMS**

BALANCED DATASET



DATASET – ‘AMAZON REVIEW FULL SCORE DATASET’

	Text	Rating
0	more like funchuck Gave this to my dad for a g...	3
1	Inspiring I hope a lot of people hear this cd....	5
2	The best soundtrack ever to anything. I'm read...	5
3	Chrono Cross OST The music of Yasunori Misuda ...	4



DATA CLEANING:

- Check rows with **null** values
- Check **duplications**
- **Merge** the columns **Title** and **Review** in one single column called **Text**

STATISTICS:

- On average, each text has a length of about **80** words
- The **standard deviation** is **43.26**, indicating a **moderate to high variability** in text length.

SUBSET:

- It’s been created a subset of **500K** reviews

TEXT PREPROCESSING (I) – EMOTIONAL TOKENS

- TEXTUAL EMOJI:
  - :) , :), : -) → **HAPPY**
  - :(, : (, : - (→ **SAD**
  - **...**
- EMOJI UNICODE-BASED:
  - 🙂😊😀😄😃 → **HAPPY**
  - 😠😡🤬 → **ANGRY**
  - …
- PUNCTUATION:
  - **?** → **QUESTION_MARK**
  - **!!!** → **STRONG_EXCLAMATION**
  - **?!** → **SURPRISE**
  - …
- MAIUSC TEXT:
  - `THIS CD IS SO BAD` → **ALL_CAPS**

Inspired by:
Agarwal et al. (2011) – *Sentiment Analysis of Twitter Data*, Columbia University.
Krouska et al. (2016) – *The Effect of Preprocessing Techniques on Twitter Sentiment Analysis,* University of Piraeus

TEXT PREPROCESSING (II)

- REMOVE OF URL and @ MENTIONS
- REMOVE THE **STOPWORDS**
- REMOVE **PUNCTUATION**, **NUMBERS** AND **SPECIAL CHARACTERS**
- EXPAND THE CONTRACTED FORMS (I’ll → I Will, they’re → They are)
- **LOWERCASING**
- **LEMMATIZATION** WITH **SPACY** (For 500K instances, 1.57h, Google Colab)

NLTK

spaCy

RegEx
Regular Expression

```
important_words = {
    "not", "no", "nor", "never", "none", "nobody", "nothing", "neither", "nowhere",
    "don’t", "doesn’t", "didn’t", "can’t", "couldn’t", "won’t", "wouldn’t",
    "shouldn’t", "wasn’t", "weren’t", "isn’t", "aren’t", "hasn’t", "haven’t", "hadn’t",
    "cannot", "without", "hardly", "barely", "rarely", "scarcely",
    "dont", "doesnt", "didnt", "cant", "couldnt", "wont", "wouldnt",
    "shouldnt", "wasnt", "werent", "isnt", "arent", "hasnt", "havent", "hadnt"
}
stop_words = stop_words - important_words
```

TEXT PREPROCESSING (III)

NOT PREPROCESSED

Text	Rating
No Lonesome Dove Lonesome Dove is one of the best books I've ever read but Dead's Man Walk can't hold a candle to it. T...	3
Not I was hoping for Don't expect these to be magical.I am pretty easy on tools, but these aren't that tough.On one stuck b...	2
roof bag This bag held up good however it is not at all waterproof we had to dry all our clothes when we arrived atOur dest...	2
Suspensful Truly after reading the reviews written by previous readers about this book, i just had to read the book. I was int...	5
Received Broken When my son received this gift, the bowl was broken. I told him to call the company, but he hasn't done t...	2
If you are computer literate just a little bit, do not read As you may have already noticed from the title of this review this is ...	1

PREPROCESSED

Text	Rating
no lonesome dove lonesome dive one good book ever read dead man walk can not hold candle ranger portray bumble i...	3
not hoping not expect magicali pretty easy tool not toughon one stuck bolt first one use break without even give fight nd...	2
roof bag bag hold good however not waterproof dry clothe arrive atour destination	2
suspensful truly read review write previous reader book read book intrigue ultimately premise story bengal tiger lose no...	5
receive broken son receive gift bowl break tell call company not do yet not know resolution	2
computer literate little bit not read may already notice title review one bad technothriller one could choose reading wou...	1

yes antiamerican get complaint book stem overuse italic **exclamation_mark** every se

love dvds **all_caps** wild wild west dvds great **all_caps** sure co

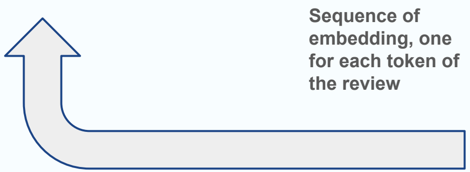
STATIC EMBEDDING APPROACH – OVERVIEW

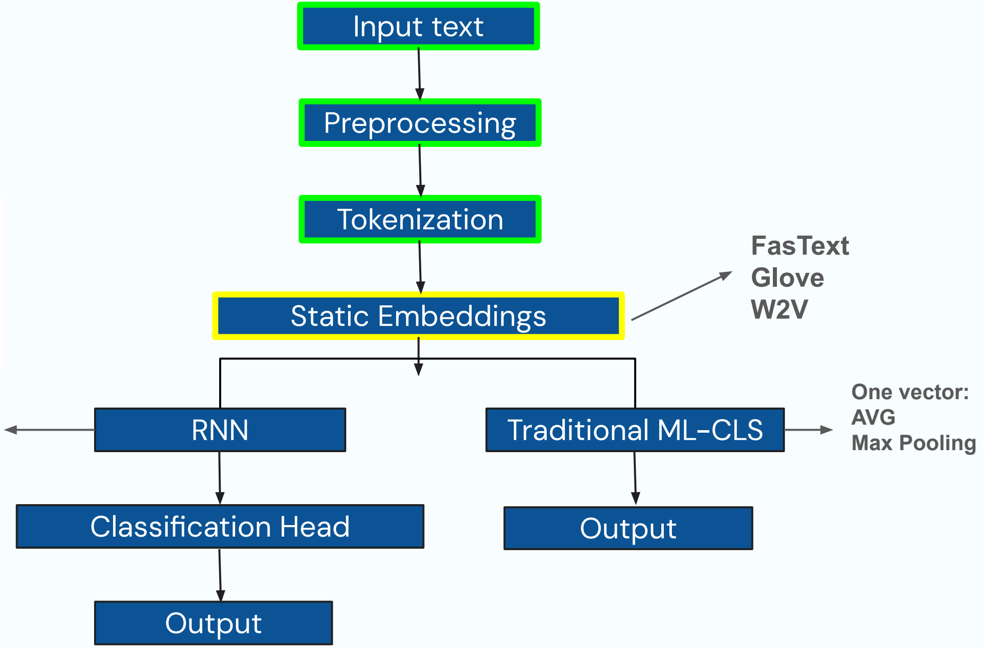
RESULTS FROM THE PAPER THAT INSPIRED OUR WORK

Tang, H., Zhang, N., Yu, X., Mao, T., & Wang, L. (2022). *Enhancing Sentiment Analysis with Word2Vec and LSTM: A Comparative Study*. Qianjiang College, Hangzhou Normal University.

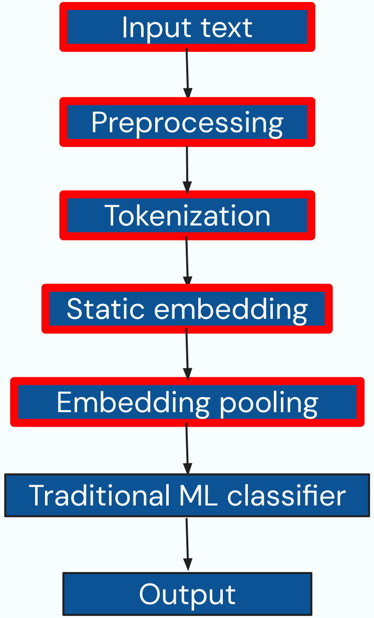
Model	Accuracy	Precision	Recall	F1
CNN	0.737	0.702	0.768	0.733
BiLSTM	0.745	0.737	0.779	0.742
Bi-LSTMCNN	0.751	0.729	0.783	0.750
Word2vec+SVM	0.762	0.731	0.798	0.767
Word2vec+LSTM	0.789	0.742	0.824	0.782

Dataset of 20K comments on the COVID-19 topic.



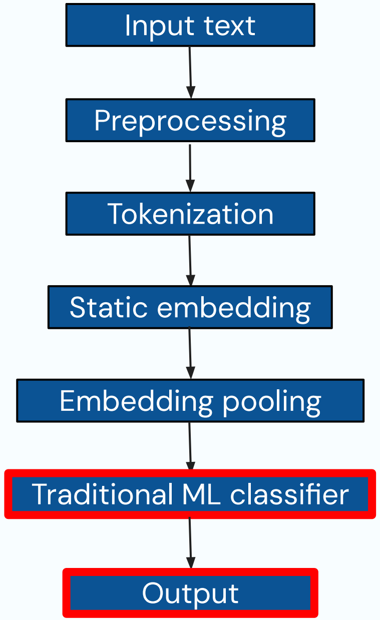


Static embedding + “traditional ML” approach



- **GloVe** word embeddings **(pretrained)**
- **Single embedding per review**
- Two pooling approaches:
  - **Average word embeddings** per review
  - **Element-wise maximum** across all word embeddings in a review.
- Evaluate the **impact of different level of text preprocessing on the performances:**
  - a **minimal** preprocessing
  - a **heavier** preprocessing

Static embedding + “traditional ML” approach



- Classifiers used:
  - **SVM**
  - **Logistic Regression**
  - **MLP**
  - **Random forest**
  - **XGBoost**
  - **KNN**
- Tools:
  - simple classifiers in **scikit-learn**

📖 *These baseline techniques are consistent with those discussed in:* **Wankhade et al. (2022)**, *A Survey on Sentiment Analysis Methods, Applications, and Challenges, Springer.*

scikit learn

🧪Traditional ML – Experiments

- 100K subsample of the dataset
- **Train/test split**: 80/20
- **Binary classification task**
- **Experiment (A)**: **minimal** and **coarse preprocessing** (lowercasing, remove punctuation)
- **Experiment (B)**: **heavier and more accurate text processing** (as illustrated before)
- **Fair comparison:** same setting for both experiments (same pretrained embedding, same dim)
- **Evaluation metrics**: **Accuracy**, Precision, Recall, F1 and taking **time** into consideration too

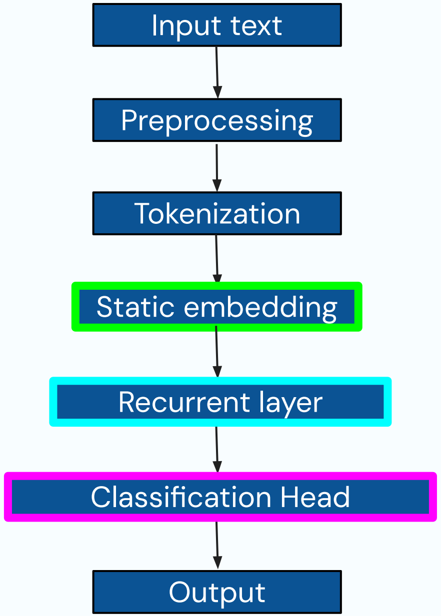
🧪Traditional ML – Results – Experiment (A)

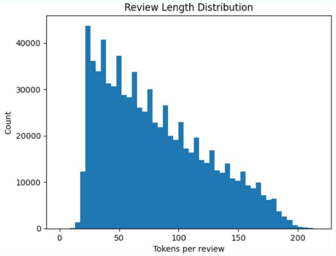
model	**Accuracy** (%) **Avg Pooling**	Training time Avg Pooling	**Accuracy** (%) **Max Pooling**	Training time Max Pooling
SVM	0.79	9 min	0.71	13 min
Logistic Regression	0.78	5 s	0.68	5 s
MLP	0.80	5 min	0.70	10 min
Random forest	0.76	2 min	0.72	1 min
XGBoost	0.77	10 s	0.78	8 s
KNN	0.70	10 s (test time)	0.61	12 s (test time)

🧪Traditional ML – Results – Experiment (B)

model	**Accuracy** (%) **Avg Pooling**	Training time Avg Pooling	**Accuracy** (%) **Max Pooling**	Training time Max Pooling
SVM	0.80	9 min	0.72	14 min
Logistic Regression	0.79	2 s	0.69	2 s
MLP	0.79	2 min	0.69	8 min
Random forest	0.76	2 min	0.73	1 min
XGBoost	0.79	10 s	0.78	8s
KNN	0.71	12 s (test time)	0.62	12 s (test time)

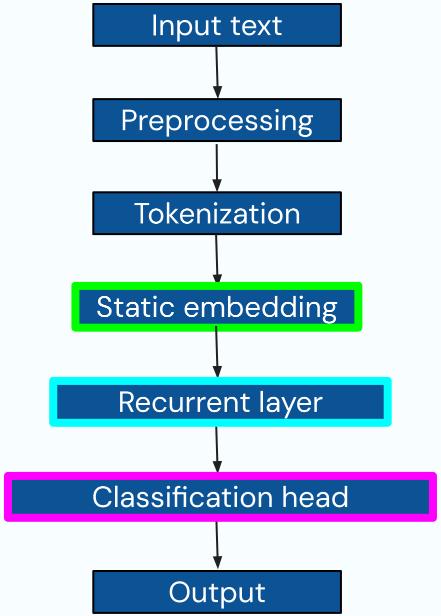
Static embedding + RNN ... and its many variants





- Adjust ***max sequence length*** dynamically based on the token per review distribution
- review as a **sequence of static embedding vector**:
  - ***Embedding layer*** from Keras (**randomly initialized** and **trained from scratch**)
  - Pretrained embedding (**GloVe, fasText,...)**: frozen or *trainable=True* to finetune
- Tools:
  - **Keras**, that allows rapid prototyping and automatic use of GPU

Static embedding + RNN … and its many variants



Many architectures have been tested varying both

- the **Recurrent layer**:
  - **LSTM/BiLSTM**
  - **GRU**
  - **Simple Vanilla RNN**
  - **Stacked BiLSTM**
- and the **Classification head**:
  - **MLP (dense layers)**
  - **CNN + dense layer**

Trained with:

- **Binary cross-entropy loss**
- **Adam optimizer**
- **Epochs: ~5** with **early stopping**

Models architecture inspired by:
Tang et al. (2022) – *Enhancing Sentiment Analysis with Word2Vec and LSTM: A Comparative Study*

🧪Deep Learning approach (RNN) – Experiment

Experiment setup:

- **500 K subsample** of the dataset, **balanced** in the target class
- **Train/test split**: 80/20
- Evaluated with Accuracy, Precision, Recall and F1 score on **internal test set**
- Optimized **binary cross-entropy loss,** used **adam optimizer**
- **Early stopping**, monitoring loss on VL set, with **patience**
- In models where it is not explicitly reported, **minimal text preprocessing** has been performed, i.e. lowercasing and removing line breaks
- Where **'+PreProc'** is indicated, **heavier pre-processing** has been performed that includes also: removing punctuation, removal of stopwords, lemmatization, etc … as illustrated before

🧪Deep Learning approach (RNN) – Results (I)

model	Accuracy-%	Precision-%	Recall-%	F1 Score-%	TR time
BiLSTM	0.935	0.939	0.931	0.935	4 min 30 s
BiLSTM + CNN	0.932	0.923	0.942	0.933	3 min
GRU	0.932	0.931	0.932	0.932	1 min 15 s
LSTM	0.930	0.937	0.923	0.930	1 min
Vanilla RNN	0.905	0.903	0.907	0.905	1 min 30 s
Stacked BiLSTM	0.931	0.928	0.934	0.931	4 min 30 s

🧪Deep Learning approach (RNN) – Results (II)

model	Accuracy–%	Precision-%	Recall-%	F1 Score-%	TR time
Glove+ BiLSTM	0.931	0.941	0.920	0.930	4 min 15 s
FasText + BiLSTM	0.930	0.927	0.935	0.931	3 min 45 s
+PreProc + BiLSTM	0.905	0.895	0.918	0.906	2 min
+PreProc + BiLSTM + CNN	0.902	0.910	0.893	0.901	1 min

🧪Deep Learning approach (RNN) – Results (III)

model	Accuracy–%	Precision-%	Recall-%	F1 Score-%	TR time
+PreProc + GRU	0.901	0.892	0.911	0.902	40 s
+PreProc + LSTM	0.901	0.900	0.902	0.901	38 s
+PreProc + Vanilla RNN	0.884	0.891	0.875	0.883	33 s
+PreProc + Stacked BiLSTM	0.903	0.908	0.896	0.902	1 min

🧪Deep Learning approach (RNN) – Results (IV)

model	Accuracy–%	Precision-%	Recall-%	F1 Score-%	TR time
+PreProc + Glove + BiLSTM	0.903	0.895	0.912	0.903	2 min
+PreProc + FasText+ BiLSTM	0.902	0.887	0.921	0.903	2 min
W2V + LSTM	0.93	0.93	0.93	0.93	2 min 30 s
+PreProc + W2V + LSTM	0.91	0.91	0.91	0.91	2 min 30 s

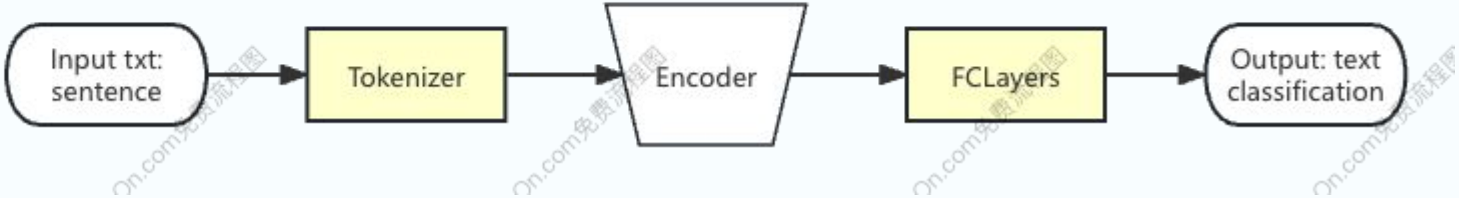
Conclusion & Insights

- The approach based on **deep learning and RNNs achieves excellent performance** significantly **outperforming** the other approach called “traditional ML”
- Both approaches work with minimal text pre-processing
- Both approaches seem to work slightly better or at least no worse with minimal preprocessing than with heavy preprocessing
- Static embeddings combined with traditional ML techniques and RNNs have proven to be not just an historical example but a simple and effective one.

Sentiment Classification with BERT

Model Architecture

Bert plus a fully connected neural network to cope with binary sentiment classification



Introduction to BERT

BERT is a bidirectional transformer pretrained on unlabeled text to predict masked tokens in a sentence and to predict whether one sentence follows another. BERT is also very versatile because its learned language representations can be adapted for other NLP tasks(sentiment analysis) by fine-tuning an additional layer or head.

Introduction to Datasets & Data Preprocessing

Datasets: amazon_review_full_csv (3M reviews)

Data Preprocessing:

1. **Ratings mapping: ratings < 3 were mapped to 0 (negative), and ratings > 3 were mapped to 1 (positive).**
2. **sampled 1% of the entire dataset as a subset: from this subset, 80% was used for training, 20% was used for validation.**

Introduction to Datasets & Data Preprocessing

2. From the remaining 99% of the original dataset (i.e., data not used in training/validation), I sampled another 1% as a test set to evaluate the model’s generalization performance.

3. Convert NAN in the title to 'NULL' & text preprocessing like convert the text to lowercase and so on.

Introduction to Model & Hyperparameters

Model: bert-base-uncased

- **parameters = 110M**
- **attention heads = 12**
- **hidden layers = 12**
- **hidden size = 768**
- **vocabulary size = 30522**
- **intermediate size = 3072**
- **activation function = gelu**

Introduction to Model & Hyperparameters

Model: Classifier head

Layer 1 (bert hidden size → 512) plus BatchNorm1d(512) plus GELU plus Dropout(0.5)

Layer 2 (512 → 256) plus BatchNorm1d(256) plus SiLU plus Dropout(0.3)

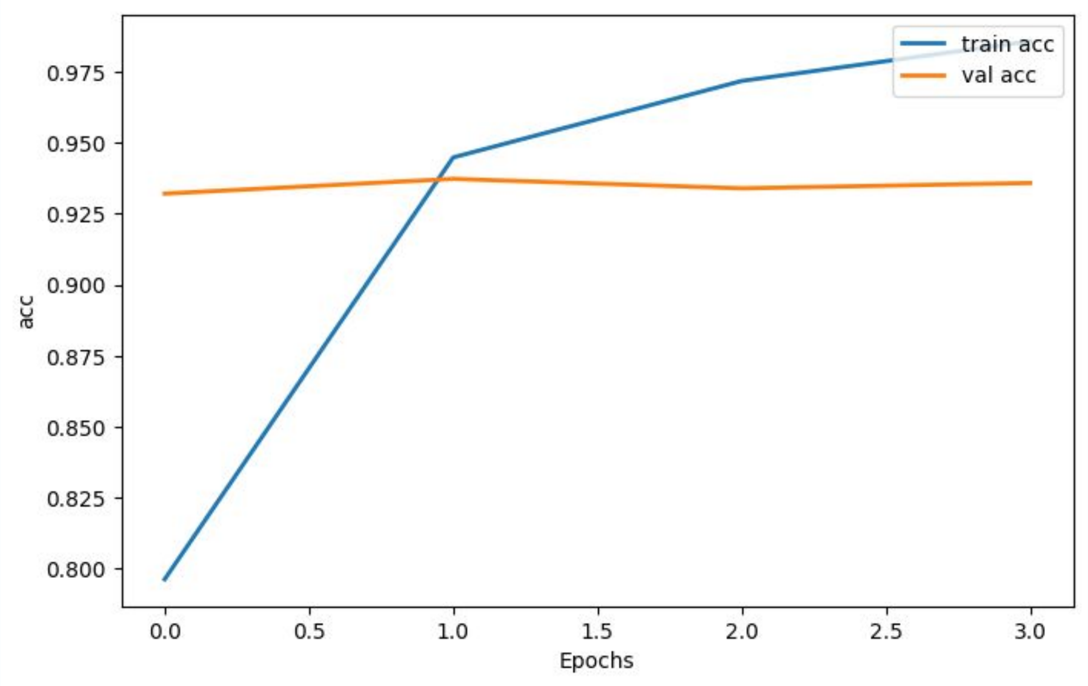
Layer 3 (256 → n_classes) n_classes in our case is 2

Experiment & Results

60k samples - encoder trainable - 10 epochs (patience 3)

Metric	60k - Fine tuning
Train Accuracy	98.5728%
Validation Accuracy	93.7292%
Test Accuracy	**93.98%**
Time Spent	14minutes

Experiment & Results



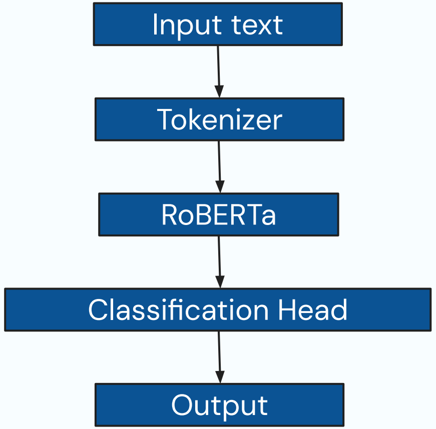
Accuracy plot

Future Work

1. To improve performance and robustness, future work will involve training and testing the model on various large-scale datasets.

2. To make the model lightweight and suitable for deployment on resource-constrained devices, I aim to experiment with distilled versions of BERT.

Sentiment Classification with RoBERTa



RoBERTa

Meta

Why RoBERTa (Meta AI, 2019)

1. Drops NSP (Next Sentence Prediction) objective.
2. Dynamic masking + larger mini batches and higher Learning Rate.
3. Pretrained on 160GB of text (10x BERT data).
4. Byte-level BPE tokenizer.

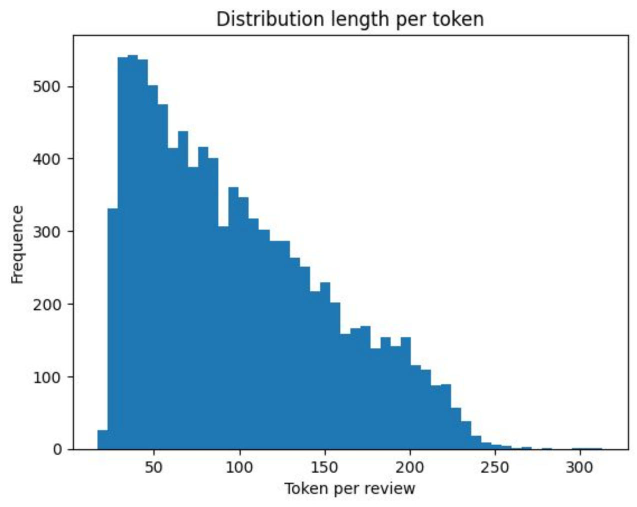
Sources: Meta AI blog, Hugging Face

Data Preparation and Sampling

- **Raw corpus loading**: *amazon_review_full_csv* (3M reviews).
- **Label mapping**: 1 -2 ⭐ -> 0 (negative) and 4-5 ⭐ -> 1 (positive).
- **Stratified sampling**:
  - Pilot run: 30K
  - Main runs: 200K
  - Maintains class ratio
- **Split**: Train 50% / Val 20% / Internal Test 30%.

Text Cleaning and Tokenization

- **Cleaning**: drop NaN, merge title+text, replace '\n' with space.
- **Sequence length probe**: tokenize a random 10k reviews -> 95th percentile ≈ 200 tokens.
- **Max length = 205**: trims only the 5% longest reviews while reducing padding for the rest.
- **Tokenizer**: RoBERTa byte-level BPE.
- **Caching**: save *inputs_ids*, *attention_mask*, *label* to NPZ -> tokenize once.




```
Token length stats:
Mean: 98.8
95th percentile: 203
Max: 313
```

Data Pipeline and Dynamic Padding

- NPZ -> Python generator -> `tf.Data.Dataset.from_generator`
- Shuffle (`buffer_size = train_size`): shuffle of the training split at each epoch.
- Repeat (`dataset.repeat()`): let's the dataset loop so `model.fit` can stream multiple epochs.
- `padded_batch`:
  - Pads each bach to its longest sequence (dynamic).
  - Uses *<pad>* ID for *input_ids*, 0 for the attention mask.
- Why **dynamic padding**?
  - Less GPU vs global pad
  - Smaller tensors -> faster throughput
- **Prefatch:** `tf.data.AUTOTUNE` -> runtime chooses the buffer size to keep GPU busy without wasting RAM.

Model and Hyperparameters

- **Base model**: *facebook/roberta-base*
  - ***parameters*** = 125M
  - ***hidden layers*** = 12
  - **hidden encoder size** = 768
  - ***vocabulary size*** = 50265
  - ***feed-forward size*** = 3072
  - **activation =** GELU
- **Custom Classification Head**: Dense 768 -> 256 -> Dropout (0.3) -> 64 -> 1
- **Loss / Optimizer**: *BinaryCrossentropy* / AdamW lr 2e-5
- **Batch size**: 32
- **Early Stopping:** patience=2 (fine-tune), patience=5 (frozen) with *restore_best_weight=True*

Experiment flow

1. **Experiment A:**
    - 30k samples – encoder trainable – 3 epochs (patience 2)
2. **Experiment B:**
    - 200k samples – encoder trainable – 3 epochs (patience 2)
3. **Experiment C:**
    - 200k samples – encoder frozen – 5 epochs (patience 2)
4. **Experiment D:**
    - 200k samples – encoder frozen – 20 epochs (patience 5)

Results: Experiments A and B

Metric	A - 30k (15k TR) - fine tune	B - 200k (100k TR) - fine tune
Train acc	90.3%	94%
Val acc	94.1%	95.5%
Internal Test acc	94.5%	95.7%
Training time	7.5 min	41 min

Both trained for 3 epochs, restored weights at epoch 1 (based on val loss).

Validation accuracies are reported at the epoch where the validation loss reached its minimum (i.e., restore_best_weights=True).

– Runned on GPU NVIDIA A100 –

Results: Experiments C and D

Metric	C - 200k (100k TR) - frozen - 5 epochs	D - 200k (100k TR) - frozen - 20 epochs
Train acc	78 %	83.3 %
Val acc	83.2 %	89 %
Internal Test acc	86.3%	89.4 %
Total time	35 min	135 min

Validation accuracies are reported at the epoch where the validation loss reached its minimum (i.e., restore_best_weights=True).

– Runned on GPU NVIDIA A100 –

Comparative Summary

Exp	Tot. Samples	Encoder	Epochs	Val %	Int. Test %	Time (min)
A	30k	Trainable	3	94.1%	94.5%	7.5
B	200k	Trainable	3	95.5%	95.7%	41
C	200k	Frozen	5	83.2%	86.3%	35
D	200k	Frozen	20	89%	89.4%	135

Validation accuracies are reported at the epoch where the validation loss reached its minimum (i.e., restore_best_weights=True).

– Runned on GPU NVIDIA A100 –

Framework and Libraries

Main frameworks and libraries used to develop RoBERTa-based experiments:

- Python 3.11
- TensorFlow 2.18 / Keras API
- Hugging Face Transformers 4.51.3
- Pandas 2.2.2
- Numpy 2.0.2
- Scikit-learn 1.6.1

RoBERTa: Future works

- **Scale training set** to a larger sample from the raw dataset (e.g >= 400k)
- Experiment with **DistilRoBERTa** for a faster and lightweight baseline

High Level Comparison of All Approaches

- **Static + ML:**
  - ~ 80% accuracy ❌
  - Training time: 2-14 min ✓
  - Simple and easy to run ✓
  - Ran on Colab CPU

- **Static + RNNs:**
  - Trained with more data
  - ~ 93.5% accuracy ✓
  - ~ 0.5 to 4.5 min training ✓
  - Ran on Colab GPU

- **BERT (Fine-tuned):**
  - ~ 94% accuracy ✓
  - Training time: 12 min (30k TR) ✓
  - Ran on Colab A100 GPU ⚠️

- **RoBERTa (Fine-tuned) A / B:**
  - ~ 94.5% / 95.7% accuracy ⭐
  - Training time:
    - A. 7.5 min ✓ (15k TR)
    - B. 41 min ⚠️ (100k TR)
  - Ran on Colab A100 GPU ⚠️

Thanks for the attention

19/05/2025

Mihnea Molnar, Gemma Ragadini, Andrea Lepori, Pan Zhang, Loris Giunta

- Group 4 -