Text Classification on AG's News Corpus

Antonia Zeibel

Technical University of Cluj-Napoca

June 3, 2025

Summary

Introduction

The goal of this project is to classify a given article into a news category using the AG's News Corpus.

Road-map:

1. **Extract articles** from the AG's News Corpus and process them into a **suitable input format**
2. Explore the news categories and **create labels** for the classification task
3. Tackle the imbalance problem between different categories
4. Use **BERT Tokenizer** and **Transformers** for the classifier
5. Prepare **Dataset** and **Dataloader** using **tokenized data**
6. Train and evaluate the BERT News Classifier
7. Final evaluation and inference

Article Structure

The archive containing the articles is actually a **tab-separated values file**, having the following identified fields:

Fields

1. source
2. url
3. title
4. subtitle
5. category
6. content
7. score/rank
8. timestamp
9. extra

Article Extraction and Loading

After identifying the structure of the article, I have parsed the file and extracted each block of articles:

Algorithm 1 Article Extraction Process

```
1: articles = []
2: for each block in extracted_articles do
3:     article = PARSE_ARTICLE_BLOCK(block)
4:     articles.APPEND(article)
```

This **list of articles** is then transformed into a **DataFrame** using Pandas library.

Cleaning `content` column

The content column is thoroughly cleaned up since the classifier is relying heavily on it. Therefore, the following cleaning steps were applied:

Cleaning Steps

1. Remove any HTML tags and decode HTML entities
2. Replace separators with blank space
3. Normalize and remove non-ASCII characters
4. Remove any non-alphanumeric characters
5. Remove unnecessary whitespaces

After cleaning the text, I also made sure to remove any rows that do not actually have a value for the `content` column.

Filter `content` column

Before using the column as input for the classifier, I had to make sure that there were no outliers. Therefore, I have performed a statistical analysis as shown in 1.

Table 1: Statistics regarding `content` column

Statistic	**Value**
count	1,238,234
mean	180.37
std	208.17
min	1
25%	114
50%	173
75%	214
max	22,572

Filter `content` column

Using the **IQR method** to filter out outliers, I have calculated a **lower bound** and an **upper bound**.

IQR Method

Computes a range between the first quartile (25%) and the third quartile (75%), and whichever point is outside this range, it is considered an **outlier**.

Clean and aggregate the `category` column

Minimal cleanup is performed on this column:

Cleaning Steps

1. simple **lowercase transformation**
2. **removal of unnecessary white space**

Before moving any further with the pre-processing steps, I have also explored the **distribution of the `category` column** shown in 2.

Clean and aggregate the `category` column

Table 2: Distribution of News Categories

Category	**Count**
world	183 341
sci/tech	144 607
entertainment	135 847
business	132 606
italia	129 824
sports	116 490
top news	99 736
europe	90 033
top stories	61 577
u.s.	47 049
health	42 387
software and developement	13 223
music feeds	7401
toons	422
ryder cup - live!	4

Clean and aggregate the `category` column

The table above (2) shows that there are serious imbalance issues in the data. To address this issue, I **aggregated columns** where the categories overlapped.

Table 3: Distribution of Aggregated News Categories

Category	**Count**
world	611 560
sci/tech	157 830
entertainment	143 670
business	132 606
sports	116 490
health	42 387

Observation!

There are still some imbalance problems present: world and health categories. These issues are going to be addressed in the next slides.

Final cleaning steps

- The main focus was on the columns `category` and `content`, but I have also cleaned up the `title` and `source`.
- Some irrelevant columns were also dropped (`subtitle`, `published_at`, `extra`, `url`).
- The data was also saved as a `csv` file for further use.

Addressing Category Imbalance (approach 1)

`world` category

To address the issue of class imbalance, I decided to **down-sample the `world` category** to 130000 samples (the mean of the categories count was 137649, excluding the `health` category).

`health` category

Instead of up-sampling or creating artificial data for this category, I decided to assign **different weights for the loss function** according to their specific distribution in the data set.

Addressing Category Imbalance (approach 2)

`health` category

I have used **back translation** to generate new data for this category. Translating from English to French, and then back to English transforms the sentence in terms of words used or even phrasing.

other categories

I still computed different weights for the loss function according to their specific distribution in the data set.

Label Mapping

Since the categories were categorical fields, I also mapped them to numerical values and saved the mapping in a `json` file.

Mappings

- 0: world
- 1: sci/tech
- 2: entertainment
- 3: business
- 4: sports
- 5: health

BERT Tokenizer

- BERT = Bidirectional Encoder Representations from Transformers
- Loaded a tokenizer specific to a BERT model which was trained on lowercase data ('**bert-base-uncased**').

BERT Tokenizer Idea

- It transforms raw text data into tokens suitable for BERT model.
- It uses **WordPiece tokenization**, which breaks down rare or unknown words into sub-word units.
- The output of the tokenizer, **the encoding input ids** represent a tensor containing the **tokens (words)**, and the **attention mask** a tensor containing 1 values for actual tokens, and 0 for padding tokens.

BERT Tokenizer

BERT Tokenizer Parameters

- text
- add_special_tokens=True
- max_length=128
- return_token_type_ids=False
- padding='max_length'
- truncation=True
- return_attention_mask=True
- return_tensors='pt'

Transformer Architecture

The classifier is built using single transformer encoder blocks which has the following structure:

TransformerBlock Internal Structure

- Multi-Head Self-Attention Layer
- LayerNorm After Attention
- Feedforward Network
- Dropout Layer

Transformer Architecture

The classifier is built using single transformer encoder blocks which has the following structure:

TransformerBlock Internal Structure

- **Multi-Head Self-Attention Layer**
  - Allows the model to focus on different parts of the sequence simultaneously using multiple attention heads.
  - Self-attention: each token attends to all other tokens in the sequence.
- LayerNorm After Attention
- Feedforward Network
- Dropout Layer

Transformer Architecture

The classifier is built using single transformer encoder blocks which has the following structure:

TransformerBlock Internal Structure

- Multi-Head Self-Attention Layer
- **LayerNorm After Attention**
  - Normalizes the output from the attention layer (plus residual connection).
  - Residual connections preserve gradients and stabilize training by adding the input back to the output of each sub-layer.
- Feedforward Network
- Dropout Layer

Transformer Architecture

The classifier is built using single transformer encoder blocks which has the following structure:

TransformerBlock Internal Structure

- Multi-Head Self-Attention Layer
- LayerNorm After Attention
- **Feedforward Network**
  - Applies two linear layers with a ReLU in between.
- Dropout Layer

Transformer Architecture

The classifier is built using single transformer encoder blocks with the following structure:

TransformerBlock Internal Structure

- Multi-Head Self-Attention Layer
- LayerNorm After Attention
- Feedforward Network
- **Dropout Layer**
  - Applied after both attention and feedforward layers for regularization.
  - Prevents overfitting.

Transformers Architecture

- Then the actual classifier uses these **transformer encoder blocks**, while taking into account the **token and their positions embeddings**.
- Moreover, it captures the **dependencies** between tokens using **self-attention**, and extracts sentence level embedding.
- The final step is the output of such embedding which basically is the classification itself.

Dataset and Dataloader

- Based on the previously processed data, I have created a Dataset and a Dataloader.
- The **Dataset** takes into account the **content**, the **categories** (labels), and the **tokenizer**.
- The **Dataloader** manipulates the created dataset in batches and supports shuffling.

Training

- Training was accelerated using **CUDA**.
- The loss function used was **Cross-Entropy** with computed weights for each class to address the **imbalance issue**.
- The optimizer used was **AdamW**.
- The model was trained for **5 epochs**, but only the model with the **best accuracy** was saved.

Training (approach 1)

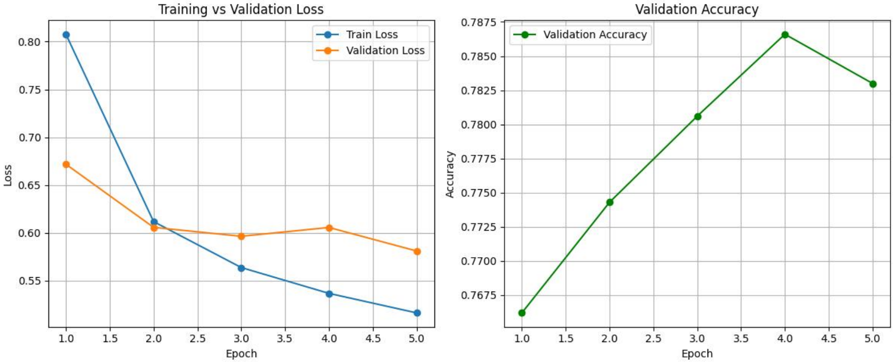


Figure 1: Loss & Accuracy

Training (approach 2)

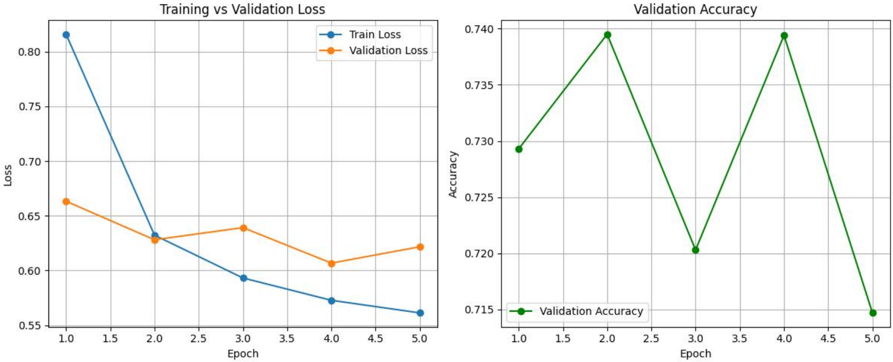


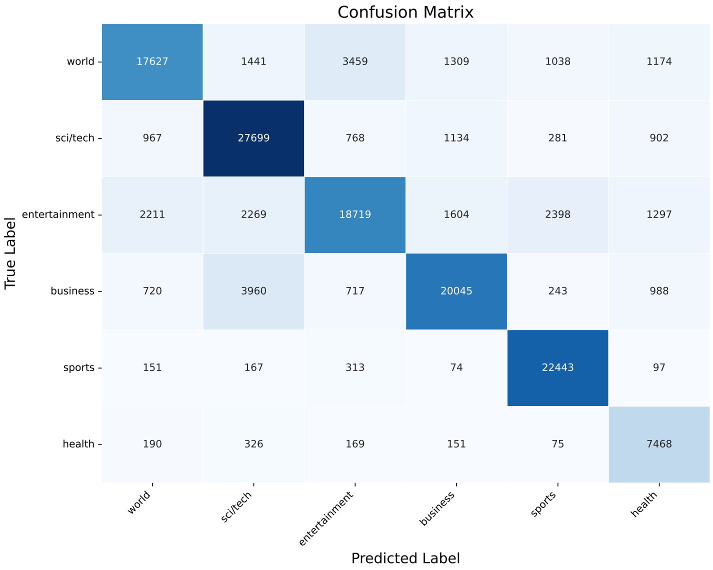
Figure 2: Loss & Accuracy

Evaluation

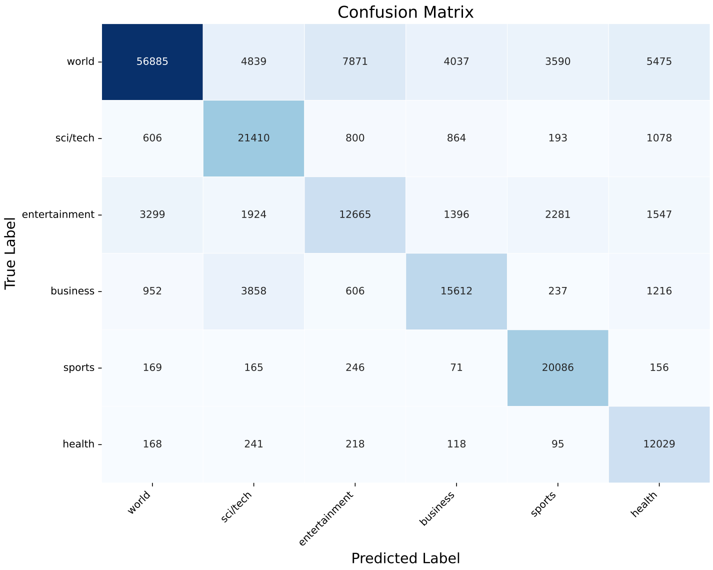
To evaluate the performances of the classifier, I have used:

- Confusion Matrix
- Classification Report

Evaluation (approach 1)



Evaluation (approach 2)



Evaluation (approach 1)

Table 4: Classification Report (approach 1)

Category	Precision	Recall	F1-Score	Support
world	0.81	0.68	0.74	26,048.00
sci/tech	0.77	0.87	0.82	31,751.00
entertainment	0.78	0.66	0.71	28,498.00
business	0.82	0.75	0.79	26,673.00
sports	0.85	0.97	0.90	23,245.00
health	0.63	0.89	0.74	8379.00
accuracy		0.79		144,594.00
macro avg	0.78	0.80	0.78	144,594.00
weighted avg	0.79	0.79	0.79	144,594.00

Evaluation (approach 2)

Table 5: Classification Report (approach 2)

Category	Precision	Recall	F1-Score	Support
world	0.92	0.69	0.79	82,697.00
sci/tech	0.66	0.86	0.75	24,951.00
entertainment	0.57	0.55	0.56	23,112.00
business	0.71	0.69	0.70	22,481.00
sports	0.76	0.96	0.85	20,893.00
health	0.56	0.93	0.70	12,869.00
accuracy		0.74		187,003.00
macro avg	0.69	0.78	0.72	187,003.00
weighted avg	0.77	0.74	0.74	187,003.00

Evaluation Comparison

Table 6: Comparison of Metrics (health and world)

Metric / Category	**Approach 1**	**Approach 2**
World Precision	0.81	0.92
World Recall	0.68	0.69
Health Precision	0.63	0.56
Health Recall	0.89	0.93

Evaluation

Some key-observations based on the classification report, and confusion matrix:

- Overall the performances achieved using the 1st approach, more specifically the accuracy percentage (79%), indicates that the model is classifying the articles well.
- The `sports` category had really good scores, with an F1-score of 0.90.
- The `health category` has a high recall score, but out of all the other categories, the smallest precision score.
- Using the 2nd approach, the `health` category increased its recall score, but altogether, it did not have better scores than the previous one.
- Even though the `world` category presented better performances using the 2nd approach, the trade-off with the other categories may not be ideal.

Future Improvements

- Explore different balancing techniques.
- Implement other augmentation method than the **back-translation**.

Thank you!