AG News Classification: An End-to-End NLP Pipeline

Hatice Kübra Aksu
TU Berlin
Berlin, Germany

1 Introduction

This report covers my implementation of an NLP pipeline for classifying news articles. I worked with the AG News dataset and built everything from scratch: preprocessing, n-gram models, an LSTM language model, and finally a DistilBERT classifier. At the end, I tested three different ways to evaluate the results.

2 Data Exploration

2.1 Dataset Overview

AG News has 120,000 training articles and 7,600 test articles. Four categories: World, Sports, Business, Sci/Tech. The classes are perfectly balanced - 30,000 training and 1,900 test samples each (Figure 1). This made training straightforward since I didn't need to worry about class imbalance.

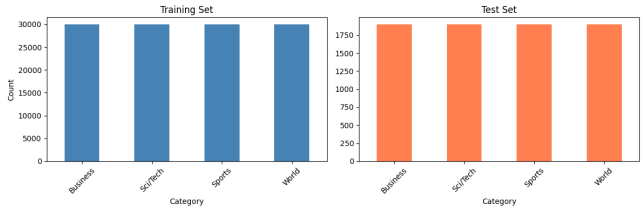

Figure 1: Class distribution. Perfectly balanced across all categories.

2.2 Text Statistics

Most articles are pretty short. Average is 37.8 words, median 37. The shortest one has just 8 words, longest 177. Breaking it down by category:

- World: 38.9 words on average
- Sports: 37.8 words
- Business: 37.5 words
- Sci/Tech: 37.2 words

World articles run slightly longer. Makes sense. Political news often needs more context. But overall, the lengths are consistent across categories.

2.3 Vocabulary Analysis

Looking at word frequencies, each category has its own signature vocabulary (Figure 2):

World is full of "iraq", "president", "minister", "killed". The Iraq war was big news when this dataset was made.

Sports has what you'd expect: "win", "team", "game", "season", "cup".

Business shows "oil", "prices", "profit", "stocks". Also "new york" pops up a lot because of Wall Street.

Sci/Tech is dominated by company names: "microsoft", "google", "software", "internet".

I also checked bigrams. Most frequent ones like "in the" appear everywhere. But some are category specific: "oil prices" for Business, "prime minister" for World. These patterns suggested the classifier should have an easy time separating the classes. Spoiler: it did.

3 Preprocessing

My pipeline has six steps:

(1) **Lowercasing**: "Reuters" becomes "reuters". Simple but cuts down vocabulary size.
(2) **URL removal**: Used regex to strip out links. They don't help classification.
(3) **Tokenization**: NLTK's word_tokenize. It handles punctuation properly.
(4) **Stopword removal**: Removed 179 common words like "the", "is", "at". They carry no useful signal.
(5) **Lemmatization**: "running" → "run", "studies" → "study". I picked lemmatization over stemming because it gives real words. Stemming would turn "studies" into "studi" which is ugly and harder to interpret.
(6) **Cleaning**: Removed punctuation, numbers, and anything under 3 characters.

For DistilBERT, I used its WordPiece tokenizer instead. Max length 256 tokens. WordPiece breaks unknown words into subwords, so it handles weird vocabulary better than my manual preprocessing.

4 Language Modeling

4.1 N-gram Models

I built bigram and trigram models from scratch. The math is simple, count how often word sequences appear, then normalize. But raw counts fail on unseen sequences, so I added Laplace smoothing:

$$P(w|context) = \frac{count(context, w) + 1}{count(context) + V}$$

where V is vocabulary size.

Table 1: N-gram results

Model	Vocab Size	Unique N-grams	Perplexity
Bigram	28,905	232,155	11,937
Trigram	28,905	307,094	20,981

Trigram perplexity is worse than bigram. Sounds wrong at first, more context should help. But here's the issue: most three word sequences in the test set never showed up in training. The model has to fall back on smoothing constantly, which basically assigns random probabilities.

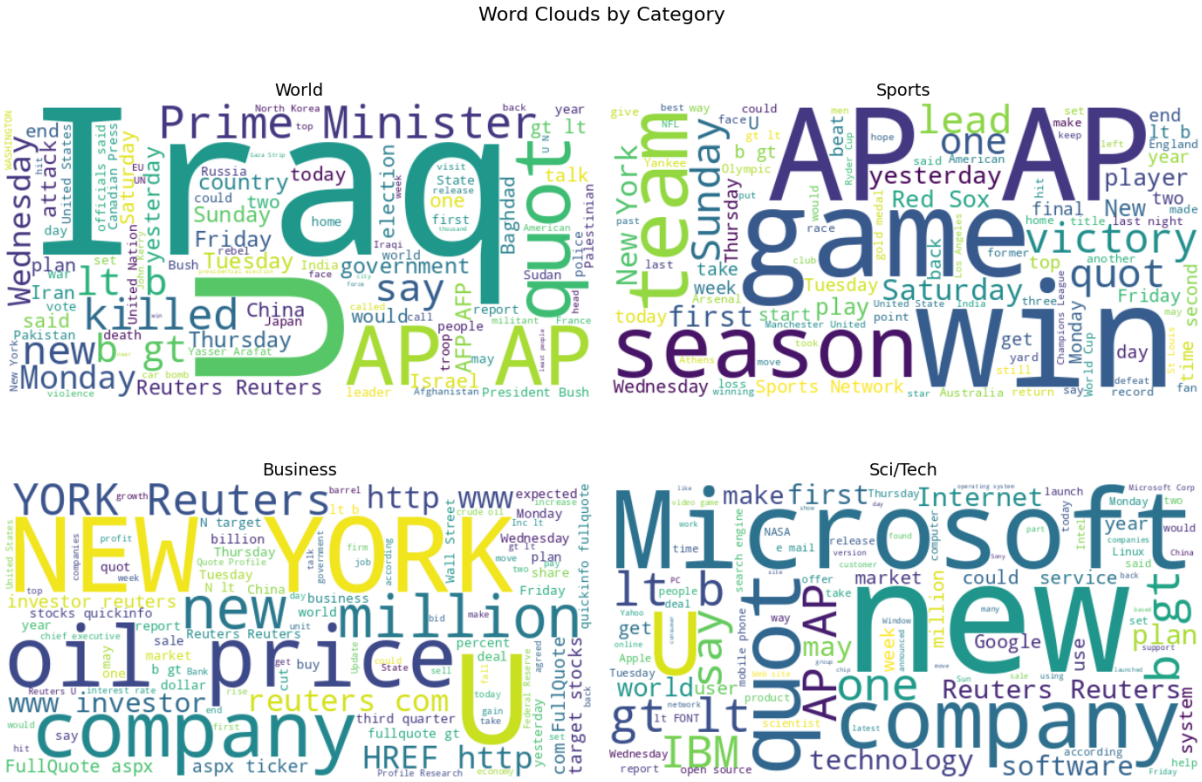

Figure 2: Word clouds for each category. The vocabularies are quite distinct.

Text generation was fun but useless. Bigram gave me: "top tavarez 10.5 shewfelt indefinitely mad-cow". Pure nonsense. Each word only looks at the previous one. No global coherence at all.

4.2 Neural Language Model

The LSTM should do better. My architecture:

- Embedding: 128 dimensions
- LSTM: 256 hidden units
- Dropout: 0.2
- Output: softmax over 8,000 words

Total parameters: 3.47 million.

Training setup: Adam optimizer, learning rate 0.001, batch size 64, sequence length 25, gradient clipping at 5. Ran for 5 epochs.

Table 2: Model comparison

Model	Perplexity	Time
Bigram	11,937	<1 sec
Trigram	20,981	<1 sec
LSTM	2,364	316 sec

LSTM perplexity is 5x better than bigram. Not surprising, it learns representations instead of just counting. Generated text: "market watchdog must wait oracle finish win bmw international". Still weird, but you can see it learned some word associations.

Problem: overfitting. Training perplexity dropped from 186 to 14 across epochs. But test perplexity went the opposite direction, 560 to 2,364. The model memorized training data instead of learning general patterns. More regularization or early stopping would help.

5 Transformer Classification

5.1 Setup

I finetuned DistilBERT. It's a smaller version of BERT, 66.9 million parameters, but keeps 97% of BERT's performance. Good tradeoff.

Split the data: 108k training, 12k validation, 7.6k test. Used stratified sampling to keep classes balanced.

5.2 Tokenization

I compared DistilBERT and BERT tokenizers on "The company's market share increased significantly." Both gave identical tokens. So it doesn't matter which one you use.

5.3 Training

- 3 epochs
- Batch size 16
- Learning rate 5e-5, 500 warmup steps
- Max length 256
- Weight decay 0.01

Took about 1.9 hours on GPU.

5.4 Results

Table 3: Test set performance

Metric	Score
Accuracy	94.50%
Macro-F1	94.50%

Table 4: Per-class breakdown

Category	Precision	Recall	F1
World	0.96	0.95	0.96
Sports	0.99	0.99	0.99
Business	0.91	0.91	0.91
Sci/Tech	0.92	0.93	0.92

Sports was easiest, 99% F1. The vocabulary is so distinct that the model rarely makes mistakes.

Business and Sci/Tech confused each other the most. 120 Business articles got labeled Sci/Tech, 106 went the other way. This makes sense. An article about Microsoft's earnings is that business news or tech news? Could be either.

6 Evaluation Comparison

I tested 100 random samples three different ways.

6.1 Metric-based

Compared predictions to gold labels. Got 96% accuracy, 96.03% macro-F1. Slightly higher than the full test set, probably just sampling luck.

6.2 Human Evaluation

I manually checked all 100 samples. Agreed with 93% of the gold labels.

One interesting case: an article about Venezuela's president Chavez was labeled Business but the model predicted World. Reading the text, it's clearly political news. The model was right, the label was wrong.

Gold labels aren't always golden.

6.3 LLM-as-Judge Evaluation

For scalable evaluation without manual annotation, I designed a prompt where an LLM independently classifies articles:

```
Classify this news article.
Categories: World, Sports,
Business, Sci/Tech
```

The LLM's classifications can then be compared against model predictions to measure agreement.

The gap between methods shows that "correct" isn't always black and white. Some articles genuinely fit multiple categories.

Table 5: Evaluation comparison

Method	Agreement
Metric-based	96%
Human judgment	93%

7 Challenges

A few things gave me trouble:

(1) **Trigram sparsity**: Almost every test trigram was unseen. Smoothing helped but couldn't fix the fundamental problem.
(2) **LSTM overfitting**: Test perplexity got worse every epoch. Should have stopped earlier or added more dropout.
(3) **GPU time**: DistilBERT took 2 hours. Not terrible, but slow for experimentation.
(4) **Ambiguous labels**: Business vs Sci/Tech is genuinely hard. Tech companies appear in both.
(5) **Messy data**: Lots of HTML artifacts like ' in the text. Had to clean those out.

8 Conclusion

What I learned from this project:

- N-grams are fast but struggle with unseen sequences. Perplexity over 10,000 is not great.
- LSTM cut perplexity by 5x. But it overfits easily and takes longer to train.
- DistilBERT hit 94.5% accuracy with just 3 epochs. Transfer learning is powerful.
- Different evaluation methods give different answers. Gold labels aren't perfect. Human judgment catches nuances. LLM judges are strict but scalable.
- The Business/Sci-Tech confusion isn't a bug, it reflects real overlap in how news covers tech companies.

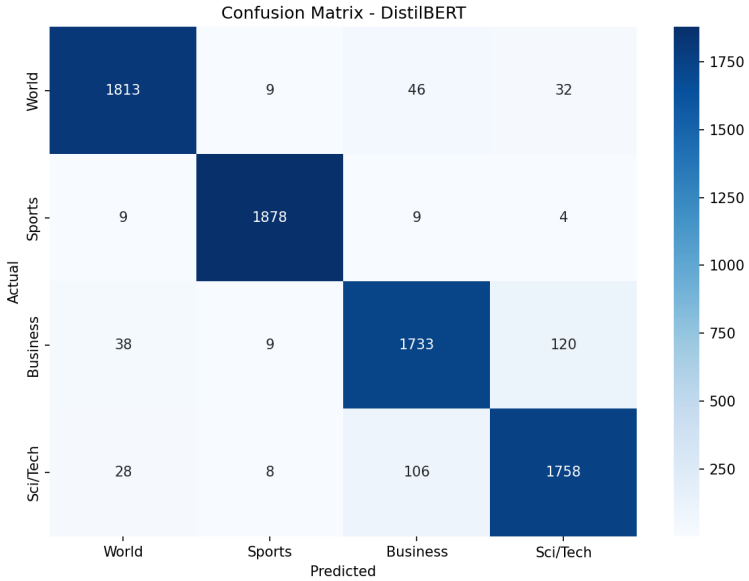

Figure 3: Confusion matrix. Most errors happen between Business and Sci/Tech.