Pirate Passau at Touché: Do We Need to Get Complex? A Comparative Analysis of Traditional and Advanced NLP Approaches for Advertisement Classification

Notebook for the Touché Lab at CLEF 2025

Tarek Al Bouhairi*, Alaa Alhamzeh

University of Passau, Germany

Abstract

This paper presents our contribution to the Touché shared task at CLEF 2025: Advertisement in Retrieval-Augmented Generation (RAG) – Sub-task 2. The task focuses on determining whether a system-generated response to a user query contains advertising content. We compare a variety of classification approaches, ranging from a traditional TF-IDF + Random Forest baseline to transformer-based methods such as sentence transformers (MiniLM, MPNet), few-shot classification with large language models (LLaMA 3.1, Qwen 2.5), and a Retrieval-Augmented Generation (RAG) pipeline that grounds LLM predictions in semantically similar examples. Each method is evaluated on the official task dataset using standard classification metrics: precision, recall, F1-score, and accuracy. Among all tested approaches, the fine-tuned sentence transformer (all-MiniLM-L6-v2) achieved the best performance, recording an F1-score of 0.97 on the test set. Our findings suggest that while prompt-based LLMs and RAG approaches offer flexibility, fine-tuned transformers remain the most effective for this task under the given conditions.

Keywords

Advertisement Classification, Sentence Transformers, Few-Shot Learning, Retrieval-Augmented Generation, Large Language Models

1. Introduction

Online information systems have content made by users and systems and it may be used to teach or to promote a product. To make search results useful, support good argument searching and avoid biased information, we should be able to differentiate between reliable info and ads. The goal of identifying advertisements in this task forms the core of Task 2 at CLEF 2025. When talking about advertising, advertisements are seen as tools that promote a product, service, or event by using persuasive or marketing language.

In this work, we examine the different techniques used to classify binary data. It is important for us to investigate how performance changes as we contrast traditional machine learning models, transformers for working with sentence data, prompt-based methods with large language models (LLMs) and models that use data retrieval for language generation.

Starting, a baseline using TF-IDF vectorization combined with a random forest [1] classifier was used. Each text is represented as a vector, with features being terms weighted by their TF-IDF values. It shows which terms exist and how well they inform the corpus, based on how often they are used in it. The random forest model uses a group of decision trees to sort out these vectors. The method is fully interpretable, can run on any computer and provides a very efficient benchmark. Then, we analyze the performance of fine-tuned sentence transformers for this task. We rely on two pretrained models: all-MiniLM-L6-v2 and mpnet-base-v2 which both have a classification head attached. They are trained in their entirety to learn special ways of processing input text for immediate classification.

Then, we test few-shot learning with LLaMA 3.1 by feeding a handful of examples and an instruction and only relying on learning in context rather than training the model. With this approach, LLMs don't

CLEF 2025 Working Notes, September 9 – 12 September 2025, Madrid, Spain

*Corresponding author.

albouh01@ads.uni-passau.de (T. A. Bouhairi); alaa.alhamzeh@uni-passau.de (A. Alhamzeh)

need to process huge datasets since they can adapt to new tasks using only format examples.

Finally, we implement a Retrieval-Augmented Generation (RAG) approach. In this setup, we use sentence embeddings to retrieve semantically similar examples from a labeled corpus via FAISS, rerank them using a cross-encoder, and provide the most relevant examples to an LLM for classification. This method allows the model to look at different examples which are the most similar to the query.

Our contributions in this paper are threefold:

1. We apply and compare a diverse set of methods, ranging from simple classifiers to prompt-based LLMs, for the binary classification of advertisements.
2. We develop a modular RAG pipeline that combines semantic retrieval and LLM reasoning to dynamically ground predictions in relevant training data.
3. We provide a detailed evaluation of all approaches using standard classification metrics such as precision, recall, F1-Score, accuracy and confusion matrix.

2. Related Work

Traditional approaches such as bag of words and TF-IDF have served as a baseline for different Natural Language Processing (NLP) tasks. Although, these techniques change raw text into simple, low-density vector forms with term frequency, so it becomes easy to use machine learning to categorize them [2]. Nevertheless, they lack understanding the deep meaning of words and sentences, so they might face difficulties for tasks such as advertisement detection.

To overcome these limitations, sentence transformers have emerged as powerful alternatives. These models, pre-trained on large-scale natural language data, encode text into dense vector embeddings that better reflect semantic similarity. While using sentence embeddings as static features already provides substantial improvements over lexical models, fine-tuning the transformer on the task-specific data further enhances performance by tailoring the representation space to the classification objective.

To overcome these challenges, sentence transformers can be used as a powerful solution. these models, pre-trained on a vast amount of data, are encoded into dense vector representations that better reflect semantic similarity. Even without fine-tuning the model, it also can be used as an improvement over lexical models. With fine-tuning the models, the internal weights of the model will start mimicking the specific task and give better results.

To build on this, exploring the capabilities of large language models (LLMs) with few-shot learning was an option. As a few examples to help the model understand what is an advertisement and what is not. A range of LLMs and sentence transformers were selected for these experiments, varying in size and architecture, as summarized in Table 1.

Model	Size
sentence-transformers/all-MiniLM-L6-v2	22.7M
sentence-transformers/all-mpnet-base-v2	109M
Llama3.1 (Meta)	70B
Qwen 2.5 (Google)	72B

Table 1
Used LLMs in Our Experiments

3. Methods

In this section, we examine several methods, ranging from basic machine learning approaches to advanced models based on language understanding, for categorizing binary advertisements. Every method is examined alone to determine its advantages, weaknesses and usefulness in solving the task in Touché Task 2. All experiments are kept as consistent as possible by using the same training and test subsets for every method. With the help of a high-performance computer system and an NVIDIA A100

80GB PCIe GPU, all experiments were optimized, making training, inference and embedding operations easy for transformer-based and retrieval-augmented models.

3.1. TF-IDF + Random Forest

For a baseline, we implemented a pipeline which has TF-IDF vectorization as a feature extractor and random forest for classification. This approach could be a baseline for comparison with modern transformer-based and LLM-based methods [2].

The response text from the training data will be transformed to numerical values using the Term Frequency-Inverse Document Frequency (TF-IDF) method. TF-IDF is a statistical method which is used to evaluate how similar words are to each other in a large collection of text. We get this score by multiplying the term's frequency in the document by its inverse document frequency to make common words less important.

This results in a high-dimensional sparse feature vector where each dimension corresponds to a word or token, and the value encodes how relevant that word is to the document.

To enhance generalization and minimize noise, we eliminated common English words and excluded terms that appeared in over 95% of the documents.

These features were then passed to a random forest classifier. This classifier builds a collection of decision trees, with each tree trained on a bootstrap sample of the data and utilizing a random selection of features at every split. The ultimate prediction is achieved by majority voting among all trees. In our configuration, we set up the classifier using 100 decision trees to guarantee adequate diversity in the ensemble. To tackle the issue of class imbalance, frequently seen in advertisement datasets, `class_weight='balanced'` setting was used in the random forest classifier. As a result, the model assigns higher importance to the minority class (typically the advertisements), helping to reduce bias toward the majority class during training and improving the model's ability to detect less frequent but important positive instances. We set the random seed to 42 to guarantee that results remain consistent across different runs.

3.2. Sentence Transformer Embeddings

To understand how well sentence transformers that show the semantic representation in the advertisement classification task, two approaches have been used: the first treats the sentence transformer as a feature extractor which generates embeddings that are classified by a different machine learning model, in this case, random forest. The second involves end-to-end-fine-tuning, which makes the model adapt its internal weights specifically for the task.

3.2.1. Sentence Embedding-Based Classification with Random Forest

The first approach to the embeddings of the transformer in sentences is to use a hybrid architecture that combines dense semantic embeddings generated from a pre-trained transformer model with a classical machine learning classifier. The goal is to check how effectively static sentence transformer works when combined with a standard classifier for deciding if an advertisement is promoting something or not.

"all-MiniLM-L6-v2" sentence transformer was used to transform each sentence into a 384-dimensional dense representation. By embedding the sentence, we extract its semantic message, which involves details such as tone, intention and the way the language is put together [3]. Significantly, the transformer is only applied in a frozen mode, which means none of its internal elements are modified during the feedforward pass. It consists only of an extractor part, providing a repeatable representation for both training and testing.

An embedding model generates results that are fed into a standard machine learning model for training. To capture the best results, we use the random forest classifier that unites several decision tree models. Each tree in the ensemble works out how to order the space for classifying something as an ad or a different type of media. Labels in the dataset are created by people who give each sentence an

advertisement status (label 1) or not (label 0). Whilst training, random forest selects 100 estimators that help ensure the dataset is statistically significant and sets up fixed weights to correct issues with labels distribution. Estimators refer to the number of trees in your trained forest. Decreased variation and better strength in forecasts are achieved by adding more trees, though it means training takes longer. When the training process ends, the model is serialized and saved to a data file for use at any time.

The methods keep the same sentence transformer, encoding every new sentence from the test set by turning it into a 384-dimensional vector. After the data is turned into embeddings, it is sent to the random forest for prediction, which outputs only a 1 or a 0. The prediction reveals whether the text is advertising or not. All the decision trees in the ensemble send their results to the model, which then decides by voting for the most popular outcome.

Using this approach brings various benefits. By separating the two phases, the embeddings are efficient to use with several classifiers. Although the transformer is not fine-tuned, the high-quality semantic data provided by the pre-training model helps this pipeline. The findings of this approach give us a good and clear point for evaluating other more advanced models such as fine-tuned transformers and retrieval-aided generation approaches.

3.2.2. End-to-End Fine-Tuning of Sentence Transformers for Advertisement Detection

The second approach was using Sentence Transformers as a feature extractor and as a classifier. In this approach, a pre-trained sentence transformer embedding model was trained on the training and validation dataset for the binary classification task by training a custom classification head on top of the transformer model. This would help the model to change its internal weights and parameters specifically for this specific task of advertisement detection. Each experiment used a different Sentence Transformer model. One was the all-MiniLM-L6-v2 and the other was mpnet-base-v2.

In both experiments, a single-layer feedforward classifier has been added to the sentence transformer. The classifier comprised a linear layer that got the sentence embedding (768- or 384-dimensional, again depending on the model) and returned only one scalar value: the unnormalized logit for the positive class. Specifically, for a sentence embedding $\mathbf{e} \in \mathbb{R}^d$, the classifier computes the logit as:

$$\text{logit} = \mathbf{w}^\top \mathbf{e} + b$$

where d is 768 for MPNet [1] and 384 for MiniLM [2]. This scalar logit is then passed through a sigmoid activation function to produce a probability $\hat{y} \in [0, 1]$ for the positive class (label 1, indicating an advertisement). Binary classification is achieved by thresholding the sigmoid output at 0.5.

The models were made by feeding each text into the appropriate Hugging Face tokenizer (MPNet or MiniLM). For tokenization, we used both truncation and padding and made the maximum sequence length be the same as the model's default. The tokenizer creates tensors called input-ids and attention-mask, that are then given to the transformer. In batch mode, all samples were lengthened to the longest sequence in the set and then sent to the GPU thanks to the utility included in Sentence Transformers.

In each experiment, the models were fine-tuned end to end, which means that the transformer parameters and the classifier weights were updated during training. Then, we used the binary cross-entropy loss with logits that applies sigmoid activation and computes the advertisement classification loss in a numerical manner.

Training was conducted using the AdamW optimizer with a learning rate of 2×10^{-5}. Mini-batch gradient descent was applied with a batch size of 16. The model was trained for 3 epochs, with two key metrics monitored after each epoch: the average training loss and the F1-score on the validation set, focusing on the positive class (label 1, indicating an advertisement). During validation, the model's raw outputs (logits) were passed through a sigmoid function to obtain probabilities, which were then converted into binary predictions using a threshold of 0.5.

After training, the model's weights were saved to disk. For inference, the saved model was reloaded and evaluated on a separate test set that was not used during training or validation. Each test example

[1] https://huggingface.co/tarekb21/MPnet-finetune
[2] https://huggingface.co/tarekb21/All-Mini-LM-v2-FineTuned

was processed through the tokenizer, then passed through the transformer and the classifier head. The output from the classifier was passed through a sigmoid activation function to produce a probability, which was then converted into a binary label using a threshold. To evaluate performance, standard classification metrics were used, including precision, recall, F1-score, accuracy and the confusion matrix.

3.3. Few-Shot Learning with LLM

To assess how large language models (LLMs) work for advertising, a few-shot method [4] was applied using the llama3.1 model [5] and Qwen2.5 [6] in a way that simulates chat interactions. Unlike traditional supervised methods, this approach does not involve training a dedicated classifier. Instead, the model is guided entirely through prompting, relying on its pre-trained knowledge and in-context reasoning capabilities to perform classification.

The prompt was carefully designed to introduce the task and provide minimal supervision through a few labeled examples. It begins with a task description instructing the model to determine whether a given text is an advertisement, explicitly requesting a response of "1" for advertisements and "0" for non-advertisements. Next, there are four examples and each one features an extract from text with an accompanying label if it is an advertisement or not. The purpose of these examples is to show you what the decision boundary is. The added new information is then connected to the original prompt to create the whole in-context example set.

For inference, the query was transmitted via a chat-based API interface that conforms to the OpenAI standard. The temperature was set to 0.1 to ensure uniform performance, and the maximum output tokens were limited to 1, as only a single numeric label was expected. The model's response was taken directly from the output.

This few-shot setup allows for assessing how well a large pre-trained model like llama3.1 can generalize the advertisement classification task based on some in-context examples, without needing specific fine-tuning

3.4. Retrieval-Augmented Generation (RAG)

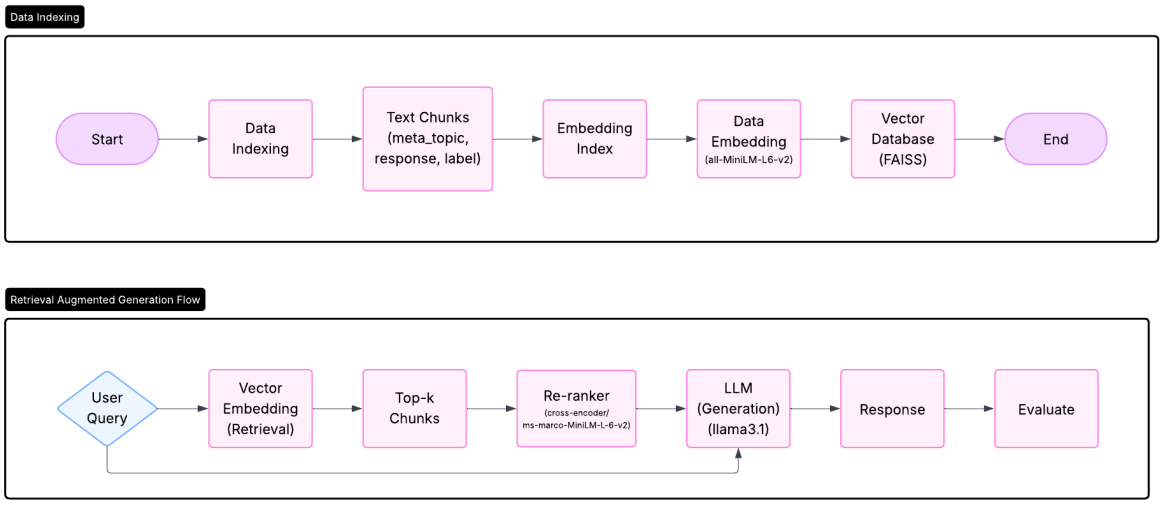

Figure 1: Overview of the Retrieval-Augmented Generation (RAG) pipeline

Retrieval Augmented Generation (RAG) has been implemented to enhance LLMs classification performance by grounding predictions in semantically similar labeled examples. The pipeline follows the same as in Figure 1 which is document indexing, embedding-based retrieval, re-ranking, and classification via prompting [7].

As a first step, merging the training and validation dataset was essential to widen the variety of examples that will be embedded in order for the language model to have the most similar examples to the test query. The dataset consists of response, meta topic, and binary label (1 for advertisement and 0 for not advertisement). For each meta topic, examples were grouped and indexed separately by label. After that, the responses were embedded using the "all-MiniLM-L6-v2" sentence transformer, and then stored in a FAISS (Facebook AI Similarity Search) [8]. This structure allowed for per-topic, per-label nearest neighbor searches during retrieval.

For the retrieval phase, the system will encode the test query that was given to the large language model and turn it into a vector embedding using the same sentence transformer (all-MiniLM-L6-v2). It then performs an approximate nearest-neighbor search using FAISS to retrieve the top K which is in our case 5 per label which will be 10 in total from the index. If the topic does not exist in the index, a global fallback search across all topics is executed.

The first set of examples is further improved by applying a cross-encoder model known as cross-encoder/ms-marco-MiniLM-L-6-v2 to calculate the semantic similarity between each pair of the query and each candidate. As a result, each example is given a relevance score and the candidates are reordered so that the system can choose the M most informative ones (M=4). To address label bias, we arrange two ad examples and two non-ad examples in the final context in the same format.

At each stage of the pipeline, a specific query context was built, the retrieval, reranking and classification procedures were run, and the final label was gathered. Using standard classification methods, predictions were checked using standard metrics: precision, recall, F1 score and a confusion matrix.

4. Results

Approach	Precision	Recall	F1-Score	Accuracy
TF-IDF + Random Forest	0.88	0.84	0.85	0.87
Sentence Transformer (all-mini-lm-v2) + Random Forest (no fine-tuning)	0.62	0.60	0.61	0.66
Sentence Transformer (all-mini-lm-v2) fine-tuned	0.97	0.97	0.97	0.97
Sentence Transformer (MPnet) fine-tuned	0.87	0.90	0.87	0.88
Few-shot (4 shots, LLaMA 3.1)	0.62	0.61	0.53	0.53
Few-shot (4 shots, Qwen 2.5)	0.62	0.57	0.45	0.47
RAG	0.693	0.688	0.622	0.622

Table 2
Performance comparison of different models and configurations

Table 2 shows all the results of the evaluated models using: precision, recall, F1-Score, and accuracy. Based on the table, the best performance was achieved by the fine-tuned Sentence Transformer (all-MiniLM-L6-v2), which has a F1-Score of 0.97 and an accuracy of 0.97. This shows the effectiveness of fine-tuning a pre-trained model for a specific task.

An impressive result was achieved with the classical TF-IDF + Random Forest model, with an F1-score of 0.85 and accuracy of 0.87. According to this, well-prepared traditional pipelines are still very valuable when resources for computations are not plentiful. Sentence embeddings without fine-tuning (i.e., all-MiniLM-L6-v2 + Random Forest) resulted in an F1-score of only 0.61 which emphasizes that sentence embeddings alone may fail to capture the important differences needed for this task.

The few-shot LLM-based classifier underperformed in comparison to the sentence transformer, even though given some examples of the task. This might be the case when limited number of in-context examples provided. The RAG approach improved compared to the few-shot LLM-based classifier, reaching an F1-score of 0.62. Although the RAG approach did not perform as well as the fine-tuned transformer models, it still shows promise. In the future, using fine-tuned sentence transformer embeddings for retrieval might help the system find better examples and improve its results.

5. Conclusion

This research addressed the task of advertisement classification by a variety of approaches, ranging from classical machine learning baselines to recent advances in language modeling. At the start, we looked at a TF-IDF model linked to a random forest classifier, which formed a powerful and understandable baseline.

Additionally, we implemented two configurations for a sentence transformer based approach: one using embeddings as input to a traditional classifier, and another via full fine-tuning with an added classification head. The fine-tuned sentence transformer model, performed impressively well compared to this specific task.

Moreover, a few shot LLM-based classifiers were used for this task, which showed limited performance compared to other approaches, highlighting the challenges of relying solely on prompting without task-specific training. Finally, we implemented a retrieval-augmented generation approach that retrieves relevant training examples by semantic similarity, and uses them to guide LLM predictions. Even though this method did not perform as well as the fine-tuned models, it could still be useful when there is not much training data or when the type of content changes often. That's because it can find helpful examples on the fly without the need to be retrained.

All in all, our best outcomes were delivered from models based on sentence transformers, but prompt-based LLMs and RAG-based pipelines still have potential for use in advertisement classification. For example, future work may incorporate fine-tuned embeddings during the retrieval phase to improve contextual grounding.

Generative AI Declaration

During the preparation of this work, the authors used ChatGPT, and LanguageTool in order to: Grammar and spelling check, Paraphrase and reword. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] G. Louppe, Understanding random forests: From theory to practice, 2015. URL: https://arxiv.org/abs/1407.7502. arXiv:1407.7502.

[2] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. E. Barnes, D. E. Brown, Text classification algorithms: A survey, Information 10 (2019) 150. doi:10.3390/info10040150.

[3] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on NLP (EMNLP-IJCNLP), Association for Computational Linguistics, 2019, pp. 3982–3992.

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, in: Advances in Neural Information Processing Systems (NeurIPS), volume 33, 2020, pp. 1877–1901.

[5] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample, LLaMA: Open and efficient foundation language models, CoRR abs/2302.13971 (2023). ArXiv:2302.13971.

[6] Qwen, :, A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Tang, T. Xia, X. Ren, X. Ren, Y. Fan,

Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, Z. Qiu, Qwen2.5 technical report, 2025. URL: https://arxiv.org/abs/2412.15115. arXiv:2412.15115.

[7] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive NLP tasks, in: Advances in Neural Information Processing Systems (NeurIPS), volume 33, 2020, pp. 9459–9474.

[8] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, H. Jégou, The faiss library, 2025. URL: https://arxiv.org/abs/2401.08281. arXiv:2401.08281.