manexagirrezabal at PAN 2026: Machine Learning for Detecting Texts Automatically Generated by AI

Notebook for the PAN Lab at CLEF 2026

Manex Agirrezabal

Centre for Language Technology (CST),
Department of Nordic Studies and Linguistics
University of Copenhagen

Abstract
In this paper we present our approach for the detection of texts generated by Generative Artificial Intelligence (GAI) using feature-based Machine Learning.

Keywords
Machine Learning, TFIDF, Stylometrics, Readability, POS tags, Character ngrams, Voting

1. Introduction

In recent years, the number of texts generated by Generative Artificial Intelligence has substantially increased. Chances that any text that we receive is written by AI have increased, and therefore, it is crucial to develop methods to detect AI-generated text. In this paper we present our approach for the Voight-Kampff Generative AI Detection Shared Task [1] at PAN 2026 [2].

2. Framework for experimentation

We built a framework to test different features. This framework, called Sky Tree for Shared Tasks (ST4ST),[1] allows to test a number of features on a data set with minimal coding. The only required job is to write how the data set should be read, which can be done by specifying a class that reads the `json` file that contains the data set, and saves the data into an internal variable of the class.

Once this class is defined, a number of functions and attributes are available, as this class inherits the functions of the parent class. using this generic class, the data set can be divided into a train/test sets, or also in a number of folds, in order to run K-Fold Cross Validation.

Then, for each experiment a configuration must be provided, and in this configuration one can specify the features that will be employed to represent the instances. The classifiers can also be specified in this configuration as well. The allowed features are:

- Bag of Words: As implemented by `scikit-learn`'s `CountVectorizer` package. In this way, one can use all attributes that CountVectorizer allows, e.g. `ngram_range`, `lower`, and so on.
- Character-level Bag of Words: Internally, it is implemented in the same as the previous item, as a `CountVectorizer`, where the `analyzer` feature is set to `char`.
- Term Frequency - Inverse Document Frequency: The same as the Bag of words, but with the counts normalized with the inverse document frequency. We use the `TFIDFVectorizer` implementation from `scikit-learn`.
- BERT representation, obtained using the `bert-base-cased` model.

CLEF 2026 Working Notes, 21 – 24 September 2026, Jena, Germany
manex.aguirrezabal@hum.ku.dk (M. Agirrezabal)
https://manexagirrezabal.github.io/ (M. Agirrezabal)
0000-0001-5909-2745 (M. Agirrezabal)

[1] https://github.com/manexagirrezabal/ST4ST/

- Part-of-Speech tags: In order to get this feature, at this point we use NLTK's default POS-tagger (`nltk.pos_tag`) and after preprocessing the whole dataset, we use `CountVectorizer` to get a Bag-of-POStags. If used in combination with `ngram_range`, this features can capture long POS-tag sequences, if they are frequent in the training data.
- Morphology: Morphological features can also be included by using Stanza's default morphological analyzer. These features, which are separated by a pipe (`|`) symbol, are obtained by using another CountVectorizer. In this way, we get the frequency of each morphological feature in each training instance.
- Sentence Bert Progression. This uses the "sentence-transformers/distiluse-base-multilingual-cased-v1" model to capture a representation of each sentence in a training instance. After that is calculated, we calculate the distance of all subsequent paragaphs. The average distance and the standard deviation are used as a way of encoding the similarity progression of sentences. The goal of this feature is to capture the meaning progression of a text, at a sentence level.
- Stylometrics: As a proxy of stylometrics, we use for different features. First, we encode function words by using the NLTK stop word lists for 33 languages. Second, we include also a punctuation distribution, which counts the frequency of all punctuation symbols in `string.punctuation`. The last three stylometric features that we include are the type-token ratio (ttr) and average sentence length (in characters and in words).
- Readability: We include readability metrics as provided by the `readability` Python package, in its readability grades.

Currently, nine different classifiers[2] can be used, but extending this is rather easy.

3. Methods

For the participation in the Voight-Kampff Generative AI Detection shared task, we decided to implement five different classifiers from ST4ST. We built five different feature vectorizers, and each feature representation was used to fit a Logistic Regression. We decided to use this classifier because of its simplicity and because it allows easy interpretation of results.

The first classifier is trained on TF-IDF features (word-level), using unigrams (`ngram_range=(1,1)`) and a maximum of 1,000 features (`max_features=1000`). The second classifier uses the stylometrics related features mentioned above. The third one uses the readability measures. The fourth classifier uses Part-of-Speech n-grams, specifically uni-, bi-, tri-, and tetragrams (`ngram_range=(1,4)`) and as in previous cases, a maximum of 1,000 features. Finally, the fifth classifier employs character level ngrams, with the same `ngram_range` and `max_features` values as in previous cases (`(1,4)` and `1000`, respectively).

We built a weighted voting mechanism, where each classifier was given a weight. This weight was the same as the performance of each single classifier on some heldout data.

3.1. Additional data set

As the results of the models trained on the provided data set were very high, we decided to create an additional data set, which we used as training material in the experiment presented here. The dataset consists of 4435 human-authored texts, which come from the following sources:

- 1,500 fiction texts from Project Gutenberg (750 word excerpts from top 10 ebooks[3])
- 1,500 news texts from the CNN news dataset on Kaggle[4]

[2]Multinomial Naive Bayes, K-Nearest Neighbors, Logistic Regression, Linear Support Vector Machine, Support Vector Machine, Multilayer Perceptron, Decision Tree, Random Forest, Gradient Boosting

[3]https://gutenberg.org/browse/scores/top#books-last30

[4]https://www.kaggle.com/datasets/hadasu92/cnn-articles

- 1,435 essays from the IELTS sample essays dataset on Kaggle[5]

We generated equivalent AI-generated versions following the same approach as the organizers did in the previous edition [3].

4. Results and Discussion

Here we present the results of all models, trained on our own data set presented above. The evaluation was carried out on the validation set.[6]

Classifier 1: TF-IDF

	Precision	Recall	F-score	Support
0	0.75	0.95	0.84	1277
1	0.97	0.83	0.89	2312
accuracy			0.87	3589
macro avg	0.86	0.89	0.87	3589
weighted avg	0.89	0.87	0.87	3589

Based on the results, this model works reasonably well. It is interesting that it shows a very high precision when capturing machine-made texts (label=1).

Classifier 2: Stylometrics

	Precision	Recall	F-score	Support
0	0.78	0.89	0.83	1277
1	0.93	0.86	0.90	2312
accuracy			0.87	3589
macro avg	0.86	0.88	0.86	3589
weighted avg	0.88	0.87	0.87	3589

This model performs similarly to the previous one. It is worth mentioning that precision and recall are more similar than in the first classifier.

Classifier 3: Readability

	Precision	Recall	F-score	Support
0	0.44	0.46	0.45	1277
1	0.69	0.67	0.68	2312
accuracy			0.60	3589
macro avg	0.57	0.57	0.57	3589
weighted avg	0.60	0.60	0.60	3589

[5] https://www.kaggle.com/datasets/mazlumi/ielts-writing-scored-essays-dataset
[6] https://zenodo.org/records/14962653

This model performs substantially worse than the previous two models, guessing correctly in 3 out of 5 cases (60% of accuracy). In experiments performed on data from the same domain, we observed a substantially superior performance of this classifier.

Classifier 4: POS-tag ngrams

	Precision	Recall	F-score	Support
0	0.62	0.90	0.73	1277
1	0.93	0.70	0.80	2312
accuracy			0.77	3589
macro avg	0.77	0.80	0.77	3589
weighted avg	0.82	0.77	0.77	3589

In this case, results are regular. The confusion matrix shows that many machine generated texts (700 out of 2312) were predicted as human-made. In a possible application, if false negatives were problematic, this classifier would not be a good idea.

Classifier 5: Character-level ngrams

	Precision	Recall	F-score	Support
0	0.66	0.92	0.77	1277
1	0.94	0.74	0.83	2312
accuracy			0.80	3589
macro avg	0.80	0.83	0.80	3589
weighted avg	0.84	0.80	0.81	3589

The results of this last classifier look rather similar to the previous one, where for human-authored texts, the model has a low recall and a higher precision, and for machine-generated texts, the opposite happens, the precision is high and recall low.

Voting based classifier

	Precision	Recall	F-score	Support
0	0.67	0.97	0.80	1277
1	0.98	0.74	0.84	2312
accuracy			0.82	3589
macro avg	0.83	0.86	0.82	3589
weighted avg	0.87	0.82	0.83	3589

Results of the weighted combination of different classifiers do not seem to have improved the model performance much, especially in terms of accuracy. One notable difference is that the precision of machine-generated texts (label=1) is the highest, if we compare it to all other cases. Out of the 1739 texts that were predicted to be made by a machine, only 33 were incorrectly classified.

Further experiment

Besides, we tested the Sentence-Bert progression method. Results are shown below:

	Precision	Recall	F-score	Support
0	0.45	0.38	0.41	1277
1	0.68	0.74	0.71	2312
accuracy			0.61	3589
macro avg	0.56	0.56	0.56	3589
weighted avg	0.60	0.61	0.60	3589

This approach seems to work similarly to the Readability approach. The main difference is that while the readability approach requires very little computational capacity, the requirements for this approach are substantially higher. It also takes quite long to build the feature representation.

5. Conclusion and Future Work

In this paper, we have presented our approach for the Voight-Kampff Generative AI Detection Shared Task. The final model was a weighted voting classifier, where the weights were calculated based on the performance of each classifier in some heldout data. The five classifiers are based on TF-IDF, stylometrics, readability, POS-tags and character n-grams.

References

[1] J. Bevendorff, R. R. Gunti, J. Karlgren, M. Fröbe, M. Potthast, B. Stein, Overview of the third "Voight-Kampff" Generative AI / LLM Detection Task at PAN and ELOQUENT 2026, in: A. Barrón-Cedeño, A. G. S. de Herrera, E. S. Salido, S. MacAvaney, J. M. Struß (Eds.), Working Notes of CLEF 2026 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR-WS.org, 2026.

[2] J. Bevendorff, M. Fröbe, A. Greiner-Petter, A. Jakoby, M. Mayerl, P. Nakov, H. Plutz, M. Potthast, B. Stein, M. N. Ta, Y. Wang, E. Zangerle, in: M. Hagen, M. Potthast, B. Stein, P. Schaer, E. Zangerle, A. Barrón-Cedeño, A. G. S. de Herrera, E. S. Salido, S. MacAvaney, J. M. Struß (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Seventeenth International Conference of the CLEF Association (CLEF 2026), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2026.

[3] J. Bevendorff, Y. Wang, J. Karlgren, M. Wiegmann, M. Fröbe, J. Su, Z. Xie, M. T. Abassy, J. Mansurov, R. Xing, et al., Overview of the "voight-kampff" generative ai authorship verification task at pan and eloquent 2025 (2025).