# NLP Milestone 2

December 15, 2024

## 1 Naive Bayes

We decided to apply the Naive Bayes algorithm because it is simple, efficient, and performs well on text-based problems. Naive Bayes assumes that features (words in our case) are conditionally independent, making it computationally efficient and a good fit for tasks with high-dimensional input data, like text.

The first step was to preprocess the data. We cleaned the datasets by removing missing values and converted text columns (*hyp_lemmas* and *tgt_src_lemmas*) from string representations into lists for easier manipulation. These columns were then combined into a single feature called *combined_text*, which represents the full input text used for classification. We then extracted features (text) and labels (categories like *Hallucination* and *Not Hallucination*) for training and testing.

To prepare the text for machine learning, we used TF-IDF vectorization. This method converts the text into numerical form by calculating the importance of each word based on how frequently it appears, while penalizing commonly used words. Initially, we included stopwords, but we noticed that only stopwords were being returned as the most important features. This led us to remove them to focus on the most meaningful terms in the text. After removing stopwords, we observed slightly better results.

To address class imbalance in the dataset, we applied SMOTE (Synthetic Minority Oversampling Technique). We performed hyperparameter tuning using grid search with cross-validation. This step tested various configurations of the TF-IDF vectorizer and Naive Bayes classifier. Once the best model was identified, we evaluated it on a validation set to measure metrics like precision, recall, and F1-score. Finally, the best model was retrained using all the training data and tested on a separate test dataset.

### 1.1 Results and Analysis

The model was evaluated using training, validation, and test datasets, with hyperparameter tuning performed over 480 configurations. The best parameters identified for the Naive Bayes classifier were: `nb alpha=0.1`, `tfidf max features=10000`, `tfidf min df=1`, and `tfidf ngram range=(1, 2)`. On the validation dataset, the model achieved an accuracy of 70%, with a precision of

72% and recall of 66% for the *Hallucination* class, and 69% precision and 74% recall for the *Not Hallucination* class.

On the test set, the model achieved 59% accuracy, with a precision of 75% and recall of 58% for *Hallucination*, and 42% precision and 62% recall for *Not Hallucination*. The confusion matrix (Figure 1 showed that 574 *Hallucination* samples were correctly identified, while 424 were misclassified. For *Not Hallucination*, 313 samples were correctly classified, and 189 were misclassified.

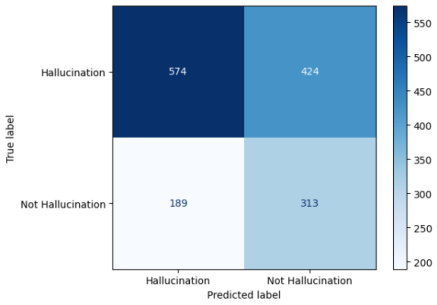



Figure 1: Overall performance of Naive Bayes shown by the plotted confusion matrix

The average length of correctly predicted samples was 77.09 characters, compared to 63.43 for incorrect predictions. For *Hallucination*, the hypothesis text was 7.16 characters shorter than the target text on average, and for *Not Hallucination*, it was 4.92 characters shorter.

The model uses different types of words to separate the two classes, focusing on rare and formal words for *Hallucination* and everyday conversational words for *Not Hallucination*. This helps the model tell the classes apart, but depending too much on specific words might make it struggle with new data.

The results show that the model performs better in terms of precision for the *Hallucination* class but struggles to achieve high recall, missing a significant portion of true hallucinations. The analysis of text lengths also suggests that shorter texts may be more challenging for the model.

### 1.1.1 DM (Dictionary Matching)

The DM task showed the best results with an accuracy of 76%. The model did well in identifying hallucinations (97% recall) but had trouble with non-hallucinations, often labeling them as hallucinations. This suggests the model leans towards predicting hallucinations. The DM task seems to have clearer patterns for hallucinations, using more specific language. This likely explains why the model performed better here but struggled with shorter, more generic text.

| Label | Precision | Recall | F1-Score |
|---|---|---|---|
| No Hallucination | 0.36 | 0.06 | 0.11 |
| Hallucination | 0.78 | 0.97 | 0.86 |

Table 1: Performance Naive Bayes for DM

#### 1.1.2 MT (Machine Translation)

The MT task performed worse, with an accuracy of 47%. The model misclassified many hallucinations (314) and non-hallucinations (314), indicating it had trouble separating the two classes. This is because translation outputs are more varied and inconsistent, making it harder for the model to detect patterns.

| Label | Precision | Recall | F1-Score |
|---|---|---|---|
| No Hallucination | 0.4 | 0.84 | 0.54 |
| Hallucination | 0.72 | 0.25 | 0.37 |

Table 2: Performance Naive Bayes for MT

#### 1.1.3 PG (Paraphrase Generation)

The PG task had an accuracy of 52%. The model struggled with both hallucinations and non-hallucinations, correctly identifying only 39 hallucinations out of 212. The PG task did not perform well because the paraphrased language is very varied, and removing stopwords made it a bit harder for the model to understand the context.

| Label | Precision | Recall | F1-Score |
|---|---|---|---|
| No Hallucination | 0.46 | 0.79 | 0.59 |
| Hallucination | 0.66 | 0.31 | 0.42 |

Table 3: Performance Naive Bayes for PG

### 1.2 Manual Evaluation

When looking at the incorrectly classified instances, you notice that the model had a lot of problems classifying the sentences and mostly learned to predict the majority although we tried to use some rebalancing methods for our data.

The model had troubles despite high similarity between the two sentences meaning that the words were almost the same.

**Example:**
Model Output: A person who is not a police officer.
Correct Output: (informal) One who is not a police officer.
Correct Label: Not Hallucination

It also classified instances as non-hallucinations when some key words were overlapping between the sentences although the overall meaning was different. This can be explained by the kind of model Naive Bayes is, because it just looks at the words and not the semantic meaning of the sentence.

**Example:**
Model Output: The state of being normal.
Correct Output: A tendency to consider most deviations as within the bounds of "normal".
Correct Label: Hallucination

On the other hand, we noticed while looking at the labels, that we would not agree with some of them. Some sentences were classified as hallucinations although for us the meaning is pretty similar. The labels of the test data were generated by a majority voting of 5 annotators that labeled the data as hallucinations or not. Sometimes these annotators could also struggle to differ Hallucinations from Non-Hallucinations. This can make up for some wrongly classified instances by our model but is by far only a minority.
This can be seen in this examples:

**Example:**
Model Output: (transitive) To cause (hair) to frizz.
Correct Output: (transitive) To curl; to make frizzy.
Correct Label: Hallucination

**Example:**
Model Output: Perfect in every way.
Correct Output: (archaic) Wholly perfect.
Correct Label: Hallucination

**Example:**
Model Output: (intransitive) To become regular.
Correct Output: To make or become regular; regularize.
Correct Label: Hallucination

## 2 Vectara

For the deep learning baseline we decided to use the *Vectara* model due to its advanced semantic understanding and context alignment. Vectara is an already pretrained model for hallucination detection in LLMs so it is well fitted for our task. Since we were given a desired outcome and we want to compare it with the given output of a model, the comparison between these two is very important. *Vectara* emphazises meaning preservation and relevance scoring, making it a robust basis for hallucination detection, since it allows it to evaluate whether the generated output aligns closely with the desired output given. For the baseline, we decided to not finetune the model any further but instead use the architecture as it is and predict the labels of the testing data for it. For the further part of our project, the goal is to finetune it with more data and therefore create a better model for our task.
In contrast to *Naive Bayes*, we chose not to compare the overall performance of a unified model with that of separate models for each task. Given that the model already uses pretrained neural networks, the outcome were expected to show no variation.

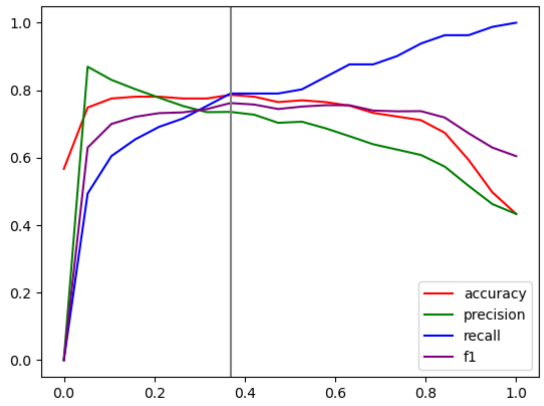



Figure 2: Model Performance of Vectara for different thresholds on the validation data for task MT.

Since Vectara predicts a value between 0 and 1 with higher values meaning a higher probability that the sentence pair is a hallucination, we had to convert this probability into a binary label. For this we first used the validation data to find a good threshold. We tested for which threshold the F1-score is the highest, meaning that this threshold balances well between precision and recall. For example for the validation data of MT for different thresholds the performance of the model can be seen in Figure 2. The threshold for MT was 0.37, for DM

0.31. For PG the validation data was not very useful for determining a threshold which is why we set the threshold for PG manually to the mean of the other two.
After determining the threshold, this was then used when predicting the testing data. For this we just inputted the reference sentence and the sentence that could be a hallucination as a tuple and converted the resulting probability in a label. This label is then stored and used for model evaluation.

## 2.1 Evaluation

### 2.1.1 Quantitative Evaluation

We first evaluated the overall performance of the Vectara model across all tasks. The overall accuracy was 70% with a recall of 73% and a precision of 61%. Without finetuning the model, these results are already quiet good. How the values differed between Hallucination and Non-Hallucination can be seen in Table 4. You can see that the Recall for Hallucination was higher but the Precision was lower. As you can see in Figure 3, the problem was that Non-Hallucinations were sometimes predicted as Hallucinations, reducing the Precision for hallucinations and Recall for the Non-Hallucinations.

| Label | Precision | Recall | F1-Score |
|---|---|---|---|
| No Hallucination | 0.79 | 0.67 | 0.73 |
| Hallucination | 0.61 | 0.73 | 0.66 |

Table 4: Performance Vectara Overall

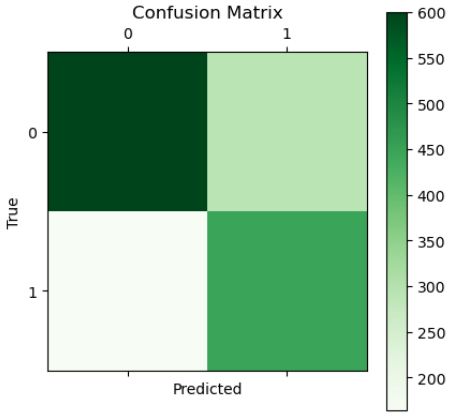



Figure 3: Model Performance of Vectara Overall visualized with a heatmap.

After looking at the overall performance, we then started to look at the tasks separately and tried to identify patterns or tasks where the model had more problems than with the others.

- **DM**
  The overall accuracy for DM was 71% with a precision of 56% and a Recall of 68%. These are similar values as for the overall performance. Here it is noteworthy that the testing data was very imbalanced with almost twice as much Non-Hallucinations than Hallucinations. How the values differ between the categories is displayed in table 5.

| Label | Precision | Recall | F1-Score |
|---|---|---|---|
| No Hallucination | 0.81 | 0.72 | 0.76 |
| Hallucination | 0.56 | 0.68 | 0.61 |

Table 5: Performance Vectara DM

- **MT**
  For the task *Machine Translation (MT)* we have a accuracy of 72% with a precision of 65% and a recall of 64%. So in comparison to the overall performance we get a slight improvement in accuracy and precision, but in return a deterioration in recall. In table 6 we can see the performance of the vectara model using solely the *MT* data.

| Label | Precision | Recall | F1-Score |
|---|---|---|---|
| No Hallucination | 0.78 | 0.55 | 0.64 |
| Hallucination | 0.60 | 0.82 | 0.69 |

Table 6: Performance Vectara MT

- **PG**

| Label | Precision | Recall | F1-Score |
|---|---|---|---|
| No Hallucination | 0.82 | 0.86 | 0.84 |
| Hallucination | 0.53 | 0.45 | 0.49 |

Table 7: Performance Vectara PG

#### 2.1.2 Manual Evaluation

For a deeper evaluation, we manually checked the classifications, especially the wrong ones and tried to find patterns that indicate problems that the model has when predicting Hallucinations.

The following patterns were noticed:

- **Difference in Length of Target vs. Hypothesis**
  The first thing we noticed was that especially when the source and the

target differed a lot in their length, the model often failed to classify correctly. We checked this also by computing the average difference between sentence lengths for correctly vs. incorrectly classified instances. For the sentence length we looked at the number of tokens. For the correctly classified instances the difference was on average only 7.8 while for incorrectly classified ones, the average difference was 12.7. This overall difference was mainly driven by the task "DM", for the other two tasks there is almost no difference in length difference between correctly and incorrectly classified instances. For DM the average token difference between source and target is also by far the largest with 28.8 for incorrectly classified and 24.4 for correctly classified instances.

**Example:**
Model Output: (intransitive) To live forever.
Correct Output: (intransitive, nonstandard) To live forever; live constantly or continually; remain alive or active.
Correct Label: Not Hallucination

- **Words having ambiguous meanings**
  Another problem the model had, was that it classified something as Hallucination because a key word in the hypothesis or target has different meanings depending on the context. One example is the word drug which can also be a medicine but here the model did not catch the right meaning therefore classifying it incorrectly as a hallucination.

**Example:**
Model Output: The fear or dislike of drugs.
Correct Output: The irrational fear or avoidance of a medicine, or of medicines in general.
Correct Label: Not Hallucination

  Another example would be that the word "heaven" is not associated to "afterlife" by the model.

- **Abbreviations, Slang Words**
  The model also had problems when abbreviations were used that it could not relate to the original word. For example "wiki" for "Wikipedia".

**Example:**
Model Output: (Internet) The process by which an article on a wiki is linked to other articles on the wiki.
Correct Output: (Wiktionary and WMF jargon) The automated process of adding links to Wikipedia to specific words and phrases in an arbitrary text (e.g. a news article).
Correct Label: Not Hallucination

  Additionally, slang words cause problems, because the model does not know them.

**Example:**
Model Output: (informal) A quid.
Correct Output: (UK, slang) pounds' worth, in terms of money
Correct Label: Not Hallucination

- **Problems despite high Similarity**

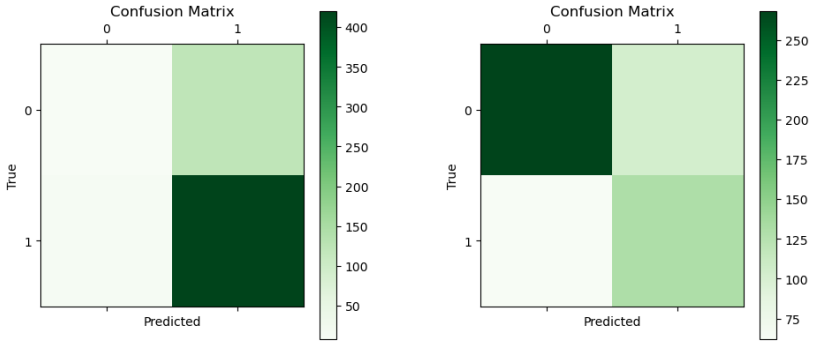
Surprisingly the model also showed a lot of problems even though the sentences were very similar. Maybe a solution for this could be to also introduce a new feature as input which measures the similarity between the two sentences and can be used for better prediction.

**Example:**
Model Output: Active in cyberspace.
Correct Output: Active in cyberspace or on the Internet.
Correct Label: Not Hallucination

# 3 Comparison between two models

- **DM**

  You can see by the confusion matrix in Figure 4 that both models perform equally good at correctly classifying Non-Hallucinations. The main difference between the models is that Vectara could also classify some Hallucinations correctly while Naive Bayes classified most Hallucinations not as such.




(a) Naive Bayes (b) Vectara

Figure 4: Comparing Confusion Matrices of Vectara and Naive Bayes for Task DM

Vectara outperformed Bayes 71 times. When you look at cases where Vectara outperformed Naive Bayes, you can see that especially when not only words are important but also how they are related in the sentence, Vectara had an advantage because it also considers semantics.

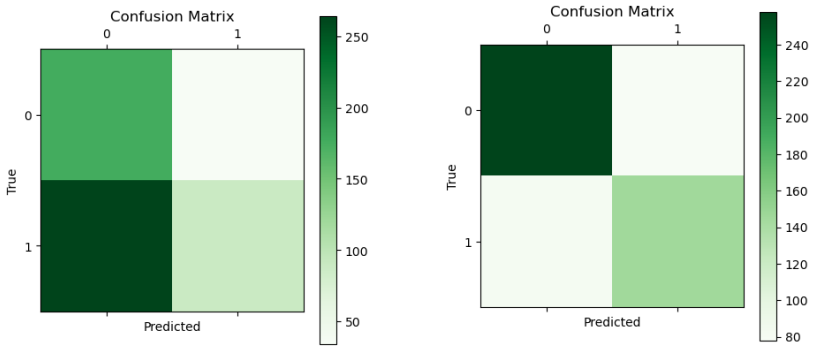
**Example:**
Model Output: (anatomy) Relating to the diaphragm.
Correct Output: (medicine) Below the diaphragm.
Correct Label: Hallucination

Naive Bayes also performed bad when the two sentences to be classified were very similar, showing that it did not learn the task well.

**Example:**
Model Output: In the middle of a scream.
Correct Output: The midpoint of a scream.
Correct Label: Not Hallucination

- **MT**

  In figure 5 we can see the confusion matrix for both models. Naive Bayes (left side) performs worse than *Vectara* (right side), as we can see that there are almost no True Predicted Hallucinations Since *Vectara* outperforms *Naive Bayes* overall, we decided to examine cases where *Naive*




(a) Naive Bayes (b) Vectara

Figure 5: Comparing Confusion Matrices of Vectara and Naive Bayes for Task MT

*Bayes* performs better compared to *Vectara*. Interestingly, we identified specific cases, where *Vectara* encounters issues. For instance, when the sentence structure in the target column differs significantly from the one in hypothesis, *Vectara* often missclassifies the entry as a hallucination, even though the meaning behind of the sentence remains the same.

**Example:**
Model Output: People generally greet each other by shaking hands in my country.
Correct Output:In my country, people usually welcome each other by hand
Correct Label: Not Hallucination

In contrast *Naive Bayes*, which assumes that all entries in a sentence are independent, disregards sentence structure completely, and thus enables the model to correctly classify such cases, unaffected by structural differences.

Another example were *Naive Bayes* outperforms *Vectara* are when negation is involved.

**Example:**
Model Output: I hate that color.
Correct Output: I can´t tolerate this color.
Correct Label: Not Hallucination

This appears to be the case sometimes, when not only a negation is involved, but also the verb is changing it the sentence. Although both verbs mean the same thing, *Vectara* missclassifies these cases as hallucination.
But there were also times where the label is to be questioned.

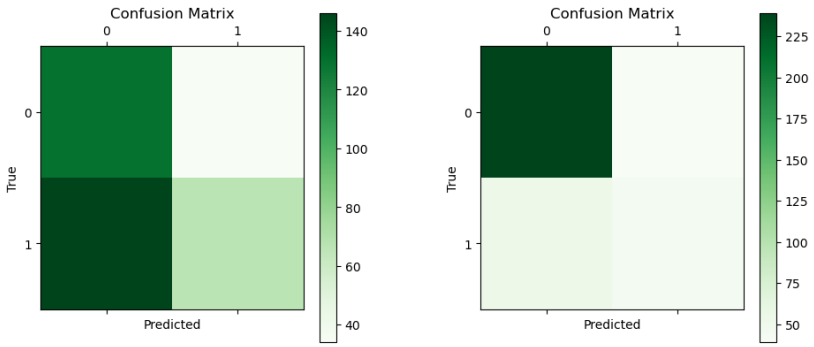
**Example:**
Model Output: Do you want to hear a joke?
Correct Output: You want an anecdote?
Correct Label: Not Hallucination

These two sentences have the label "No hallucination", but the meaning of those two sentences differ. Maybe in some speech, one could be slang for the other, but if we look at this sentence without interpretation, these

two mean different things, so in this case the label could be questioned.

- **PG**

  By the confusion matrix in Figure (b) we see that Vectara outperforms Naive Bayes in correctly classifying Class 1 instances. While both models perform decently well in identifying Class 0 instances, Vectara demonstrates better recall for Class 1, which Naive Bayes struggles to classify.




(a) Naive Bayes (b) Vectara

Figure 6: Comparing Confusion Matrices of Vectara and Naive Bayes for Task PG

Vectara outperformed Naive Bayes 79 times. When you look at cases where Vectara outperformed Naive Bayes, you can see that the semantic understanding of the task played a significant role in Vectara's improved performance.

**Example:**
Model Output: Thank you so much, mate.
Correct Output: Thank you so much, fella.
Correct Label: Not Hallucination

Analysis: Vectara correctly classified this as Not Hallucination. Despite minor differences like "mate" and "fella," the core meaning remains unchanged, showing Vectara's ability to understand word substitutions in a semantic context.

**Example:**
Model Output: Everythings fine.
Correct Output: Everything will be allright.
Correct Label: 1

Analysis: Naive Bayes likely struggles with this example because it relies heavily on exact word matches. Minor spelling differences (Everythings vs. Everything) and phrasing changes (fine vs. will be allright) might confuse the model, leading to a misclassification.

This comparison highlights that Vectara's ability to consider relationships and semantics gives it a clear advantage over Naive Bayes for Task PG.

For Vectara the mean difference for correctly classified sentences (-5.66) indicates that the model was able to detect clear distinctions between the predicted and target outputs, leading to accurate predictions. In contrast, the mean difference for incorrectly classified sentences (-3.01) is smaller, suggesting that these examples were more ambiguous or had overlapping features, making them harder to classify. This highlights that larger differences correlate with higher confidence and correctness in predictions, while smaller differences reflect greater model uncertainty.

# 4 Conclusion

For Milestone 2 we used two baseline models. One traditional Machine Learning model, an implementation of Naive Bayes, and a Deep Learning Model, a pretrained model for hallucination detection called Vectara.
As our results show the Naive Bayes model performs worse on the data than the Vectara model. This is not surprising because the task requires a semantic comparison between two sentences to check whether it is a hallucination or not. For this it is not enough to just check words but the model has to consider the context and multiple meanings.
But still also Vectara has some problems which results in an overall accuracy of 70%. This still leaves room for further improvement. To improve the model, a finetuning with the given training data is necessary and also feature engineering could be an option.
Nevertheless, we have to consider that using a deep learning model is less transparent than traditional approaches. Therefore, it makes sense to also consider other approaches that are more explainable. One approach could be to measure the similarity between the sentences and decide based on this value. For this you could use different similarity measures and different encodings.
Another problem we found was that some of the ground-truth labels are not reflecting our understanding of hallucinations / non-hallucinations. So we also have to recheck the labeling for our ongoing project because this can also affect the evaluation of our models.