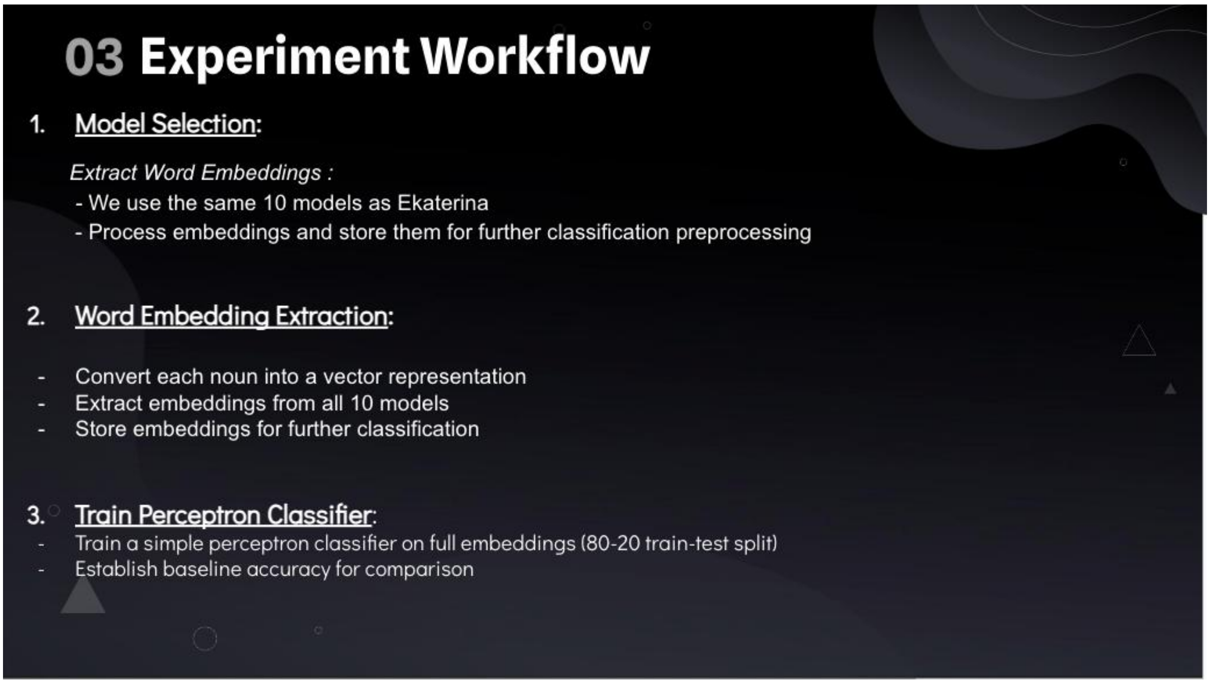
5. Give more details about perceptron architecture



Architecture of the Perceptron Used

The Perceptron model used in our experiment is implemented using sklearn.linear_model.

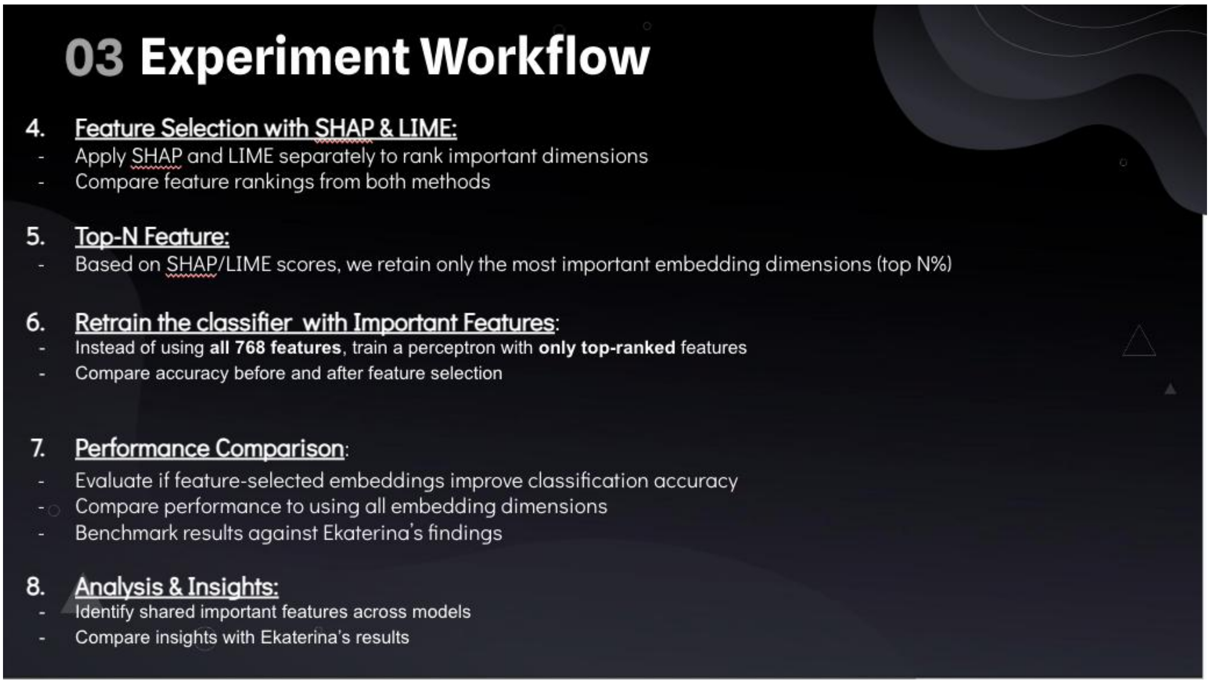
Perceptron, which is a single-layer linear classifier that learns a hyperplane to separate two classes.

Details About the Architecture:

1. Single-Layer Linear Model:The perceptron is a linear classifier, meaning it finds a linear decision boundary to separate masculine (1) and feminine (0) nouns based on their embeddings.
2. Training with Stochastic Gradient Descent (SGD):The model updates its weights iteratively using SGD, adjusting them to reduce classification errors.
3. Hyperparameters Used in Our Code:
   - max_iter=1000 → Allows the perceptron to update weights for up to 1000 iterations to optimize decision boundaries.
   - tol=1e-3 → Training stops if weight updates become smaller than 0.001, avoiding unnecessary computations.
   - random_state=42 → Ensures reproducibility by setting a fixed seed for random processes.

The perceptron only works for linearly separable data and does not handle non-linearly separable cases well. It does not have hidden layers.

6. Depending on the perceptron architecture, finally, is there a correlation between weights given to features in the perception and the importance given by SHAPE or LIME?



STATUS: TO DO

Plan to Find the Correlation

To determine the correlation, we need to **compare**:

1. **Perceptron Weight Magnitudes** (`abs(perceptron.coef_)`).
2. **SHAP Importance Scores** (`shap_feature_dict` values).
3. **LIME Importance Scores** (`lime_feature_importance` values).

Statistical Approach:

- Compute **Spearman's Rank Correlation Coefficient** or **Pearson Correlation** between:
  - `abs(perceptron.coef_)` and SHAP scores.
  - `abs(perceptron.coef_)` and LIME scores.
- A high correlation (close to +1 or -1) means that features the Perceptron assigns high weight to are also considered important by SHAP/LIME.
- A low or zero correlation means that Perceptron weight importance does not align with SHAP/LIME importance.

768: it is not always 768, is it?

Yes, it is not always 768 because some models have more than 768 dimensions in our experiment.

8. How did you select the sub-sample?

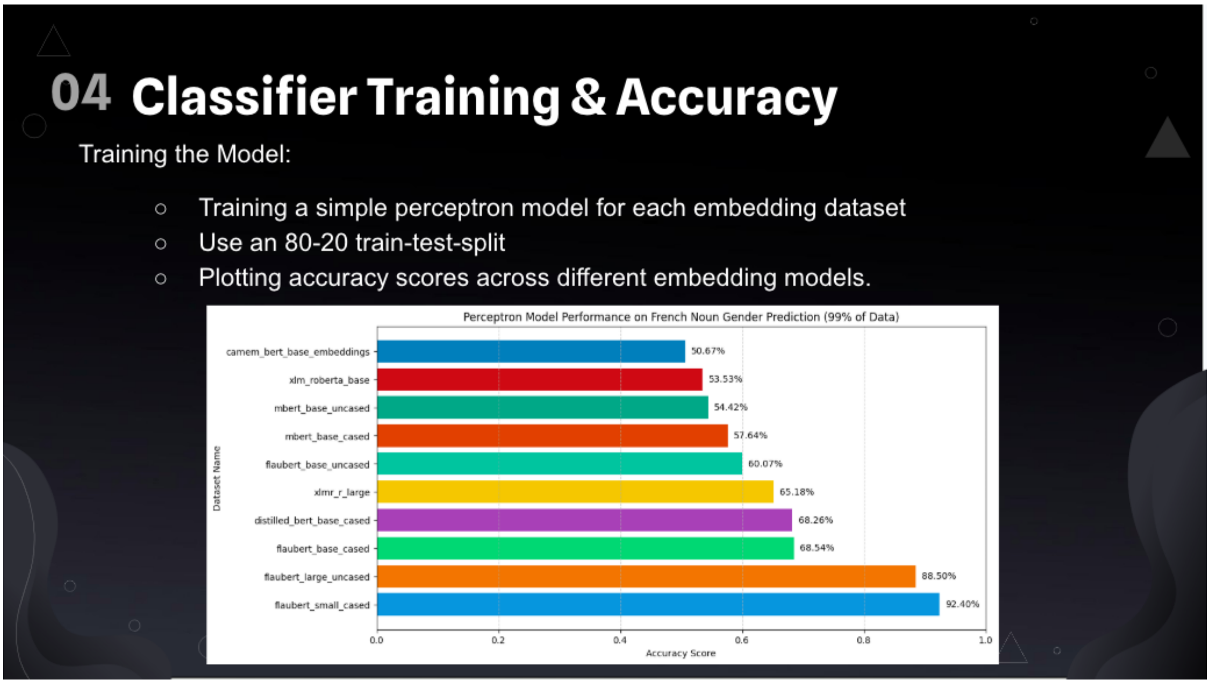
We used an **undersampling** strategy to balance the dataset across gender categories.

Selection Process for the Sub-Sample:

- For each dataset, we identified the number of masculine (1) and feminine (0) nouns.
- We determined the smaller class size between the two.
- Then, we randomly sampled an equal number of masculine and feminine nouns based on this minimum size.
- The selected samples were then concatenated and shuffled to ensure fairness in training.

This method ensures that both classes are equally represented, preventing bias toward the more frequent gender category.

9. 99% of data: what do you want to say?



- Initially we used a **subset of data (10%)** to train the model to check if everything works as expected
- Later on we **increased** dataset size **to 99%** to test model on a larger dataset
- 99% maybe confusing here, it would have been **100% ideally.**
- **(99% of data means that before training the perceptron, we randomly select 99% of the dataset instead of using all of it 100% of the dataset, we randomly select 99% from each model's embeddings. This helps the model generalize better and reduces the risk of overfitting while still keeping a large amount of data for training.)**

Propose a discussion about the link between number of features and model performance (also multi/mono lingual).

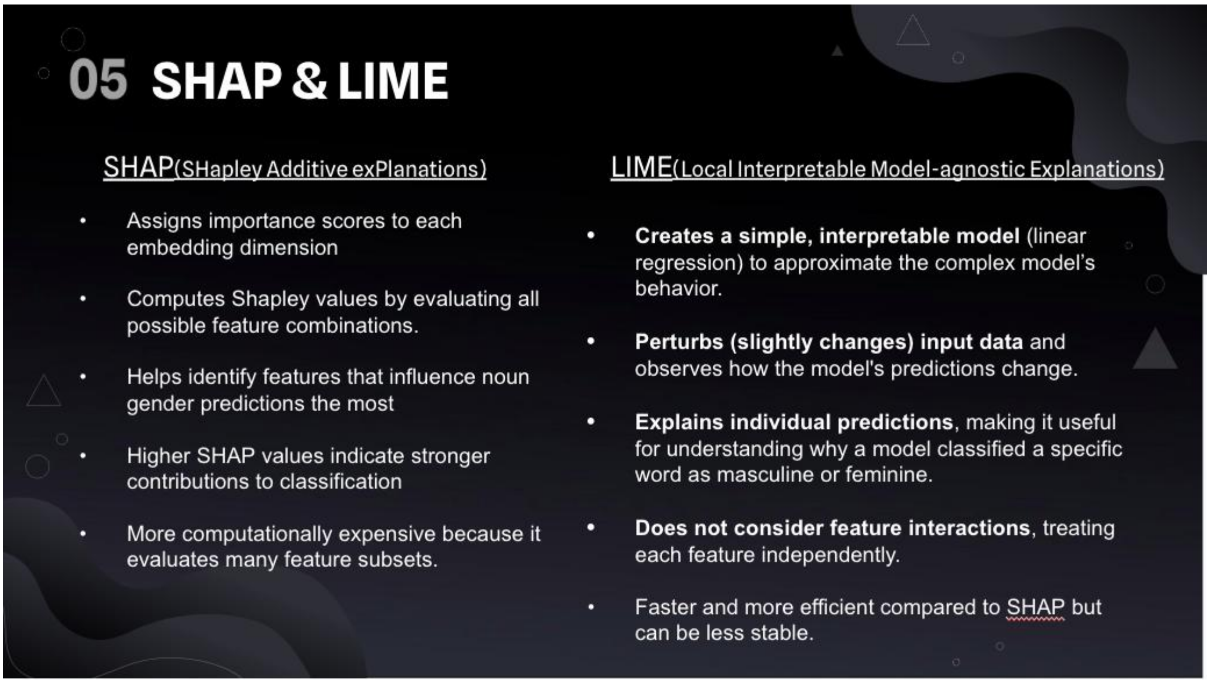
- **More features ≠ better performance:** Adding more dimensions can lead to overfitting, redundancy, and increased noise, reducing generalization.

- **Multilingual vs. Monolingual Models:** Monolingual models (e.g., Flaubert) may capture finer linguistic nuances, while multilingual models (e.g., XLM-R) generalize better but may underperform on single-language tasks.

Why Does FlauBERT-Small Outperform FlauBERT-Large in Our Experiment?

- Flaubert-Small outperforms Flaubert-Large, it suggests that a **lighter model with fewer features is sufficient,** and the large model may suffer from overfitting or noisy embeddings.
- Ekaterina's results showed FlauBERT-Large performing best, but she used a neural network with softmax activation
- Our Perceptron is a linear classifier and may benefit from more compact and focused gender-relevant dimensions in FlauBERT-Small.
- FlauBERT-Large has more parameters, which can lead to distributed gender encoding across many dimensions, making it harder for a simple model to capture compared to FlauBERT-Small.
- The difference in feature selection methods (SHAP/LIME vs. weight extraction) might also explain the discrepancy between our results and Ekaterina's.

10. Are SHAP and LIME independent of the perceptron training? In other words, do you apply SHAP and LIME on exactly the same model (same weights)?



- **Yes**, SHAP and LIME are **independent** of training.

- Same Model, Same Weights: We apply SHAP and LIME on the final trained model (same weights), ensuring that interpretations are based on the exact same decision boundary.

SHAP (Global Feature Importance Analysis)

- SHAP calculates feature importance by considering all test samples.
- It assigns an average contribution score to each feature across all predictions, making it a global explanation method.
- SHAP importance values are later used for feature selection, where models are retrained using only the most important features.

LIME (Local Feature Importance Aggregation)

- LIME explains individual predictions rather than the whole model at once.
- To extract a global ranking of important features, we apply LIME to multiple test samples and aggregate the feature importance scores.
- Like SHAP, LIME-selected features are used to retrain the perceptron model to evaluate their impact on performance.

LIME gives an explanation for one prediction, doesn't? Then, how do you extract the most important features on the whole test set?

LIME is **local**, so it explains a single instance at a time.

To get global feature importance, we:

1. **Sampled multiple test instances (1% of total dataset)**.
2. **Aggregated feature importance scores** across explanations.
3. **Computed mean absolute importance** per feature over the whole test set.

(…more explanation below)

Step 1: Apply LIME to Multiple Instances

- We randomly sample test instances from the dataset (ensuring class balance).
- Each instance is explained using LIME.
- The feature importances from different explanations are accumulated over multiple instances.

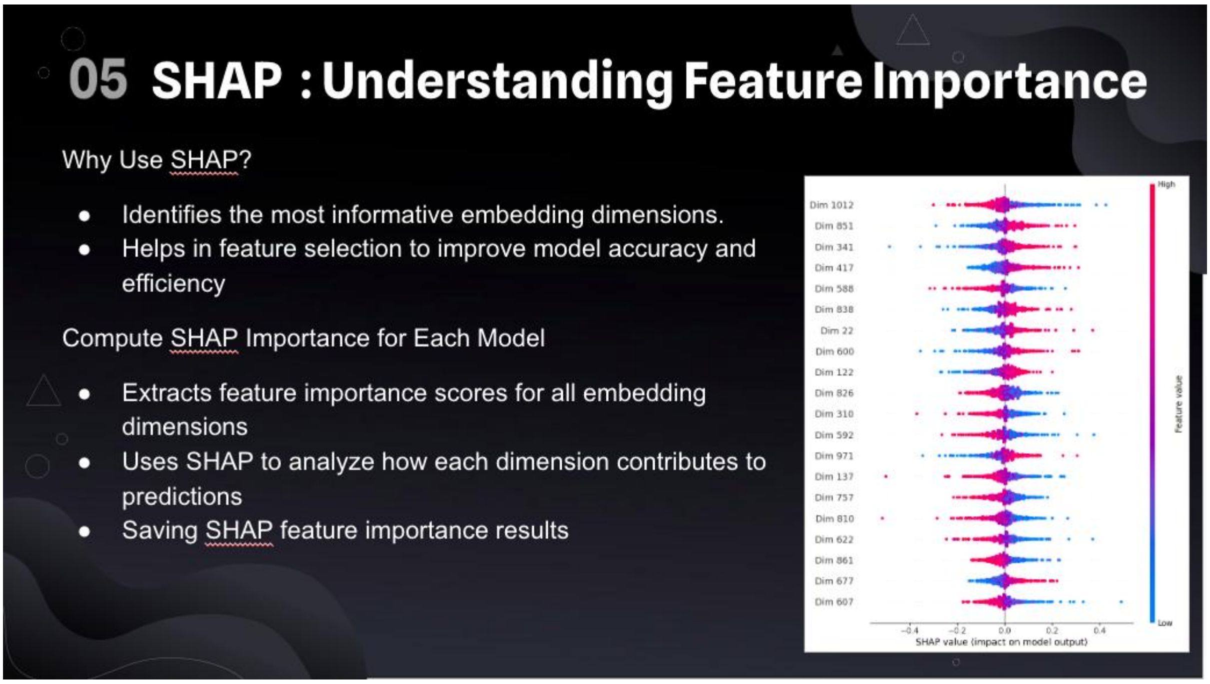
Step 2: Store and Normalize Feature Importance

- Importance scores from all explained instances are summed and averaged.
- This creates a global feature ranking instead of a per-instance explanation.

Step 3: Select Top Features & Retrain Model

- We select top N% most important features based on aggregated LIME scores.
- The perceptron is then retrained using only these features to analyze the impact of feature selection.
- Accuracy comparisons between the full model and LIME-selected feature models are made.

11. Note sure how to read the figure



This SHAP summary plot visualizes the impact of different embedding dimensions on the model's gender classification decision for the FlauBERT-Small-Cased model.

X-axis: SHAP value (impact on model output)

- **Negative SHAP values (left side):** These dimensions contribute to predicting feminine nouns (class 0).
- **Positive SHAP values (right side):** These dimensions contribute to predicting masculine nouns (class 1).
- **Values close to zero:** These dimensions have minimal impact on the classification.

Y-axis: Embedding dimensions (Dim X)

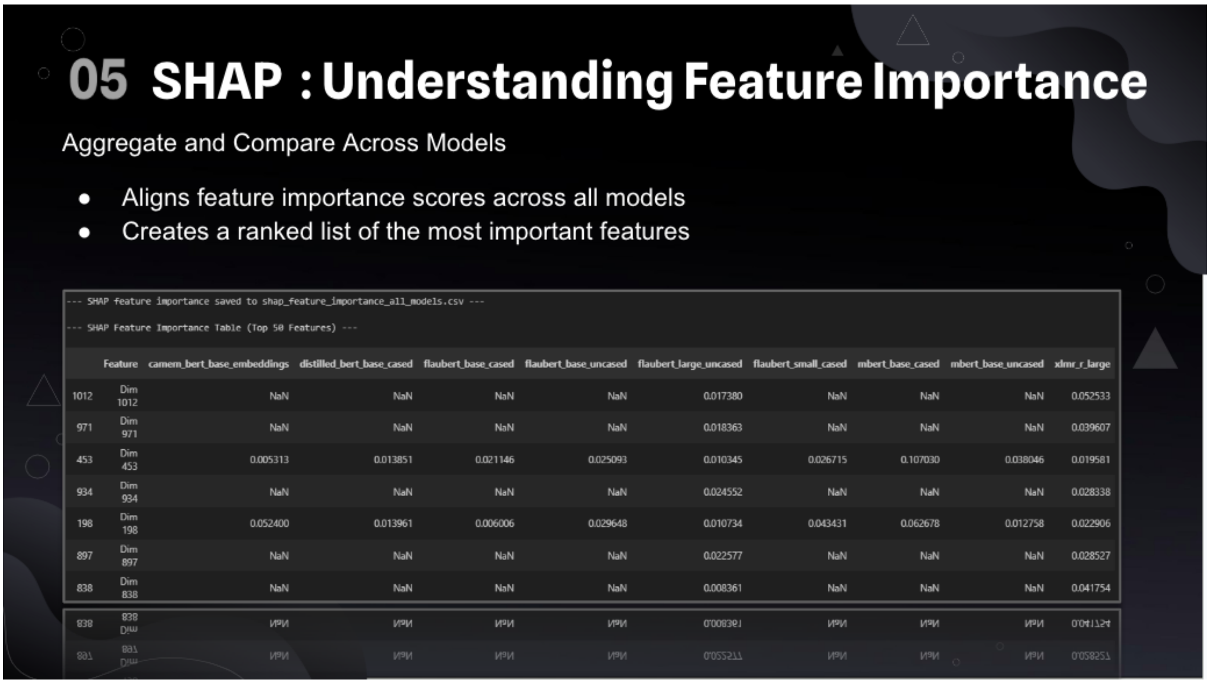
Each row represents one dimension from the word embeddings (Dim 350, Dim 265, etc.).

The most important dimensions are listed at the top.

ANALYSIS:

- Certain embedding dimensions are highly correlated with gender classification (Dim 350, Dim 265, Dim 198 are important).
- The relationship between feature values and SHAP impact varies across dimensions, showing some features are more important for distinguishing masculine vs. feminine nouns.
- This insight helps us select the most important embedding dimensions for more efficient gender classification (using the top 10%-30% of features).

12. How to understand NaN? I think that scores for features can not be compared from one model to another model because you can not be sure that feature x in model A corresponds to feature x in model B.



	Feature	camem_bert_base_embeddings	distilled_bert_base_cased	flaubert_base_cased	flaubert_base_uncased	flaubert_large_uncased	flaubert_small_cased	mbert_base_cased	mbert_base_uncased	xlmr_r_large
1012	Dim 1012	NaN	NaN	NaN	NaN	0.017380	NaN	NaN	NaN	0.052533
971	Dim 971	NaN	NaN	NaN	NaN	0.018363	NaN	NaN	NaN	0.039607
453	Dim 453	0.005313	0.013851	0.021146	0.025093	0.010345	0.026715	0.107030	0.038046	0.019581
934	Dim 934	NaN	NaN	NaN	NaN	0.024552	NaN	NaN	NaN	0.028338
198	Dim 198	0.052400	0.013961	0.006006	0.029648	0.010734	0.043431	0.062678	0.012758	0.022906
897	Dim 897	NaN	NaN	NaN	NaN	0.022577	NaN	NaN	NaN	0.028527
838	Dim 838	NaN	NaN	NaN	NaN	0.008361	NaN	NaN	NaN	0.041754

NAN

- Some models have more than 768 dimensions so NAN was used for dimensions higher larger than 768

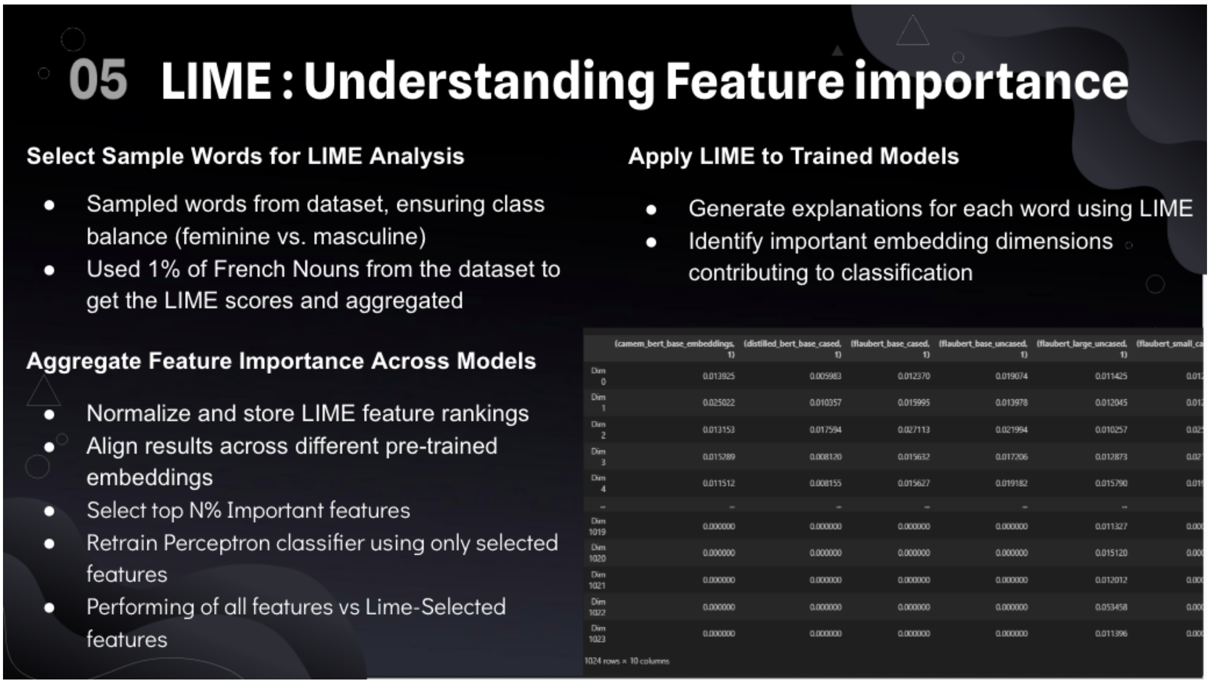
I think that scores for features can not be compared from one model to another model because you can not be sure that feature x in model A corresponds to feature x in model B.

13. top N features are not the same for each model. Yes?

- **Yes, the Top N%** are **different** for each model.
- **For convenience**, we stored results of all models in **a single file.**
- The top N features vary across models because each model learns and encodes information differently.Therefore, we cannot assume that a highly ranked feature in one model is equally important in another model.

14. Not sure to understand this slide. You address the problem I cite in 10. ?

Is the 1% linked to the 99% of slide 9.?



No, the 1% in LIME does not refer to the 99% used for training in Slide 9.

- The 99% in Slide 9 refers to the percentage of data used for training the perceptron (leaving 1% as unused or validation).
- The 1% in LIME refers to the percentage of word samples used for feature explanation (selecting 1% of words to analyze with LIME instead of running it on the entire dataset).
- This 1% sampling is done to speed up the LIME computation, which is otherwise expensive on large datasets.

Explained before:

LIME is **local**, so it explains a single instance at a time.

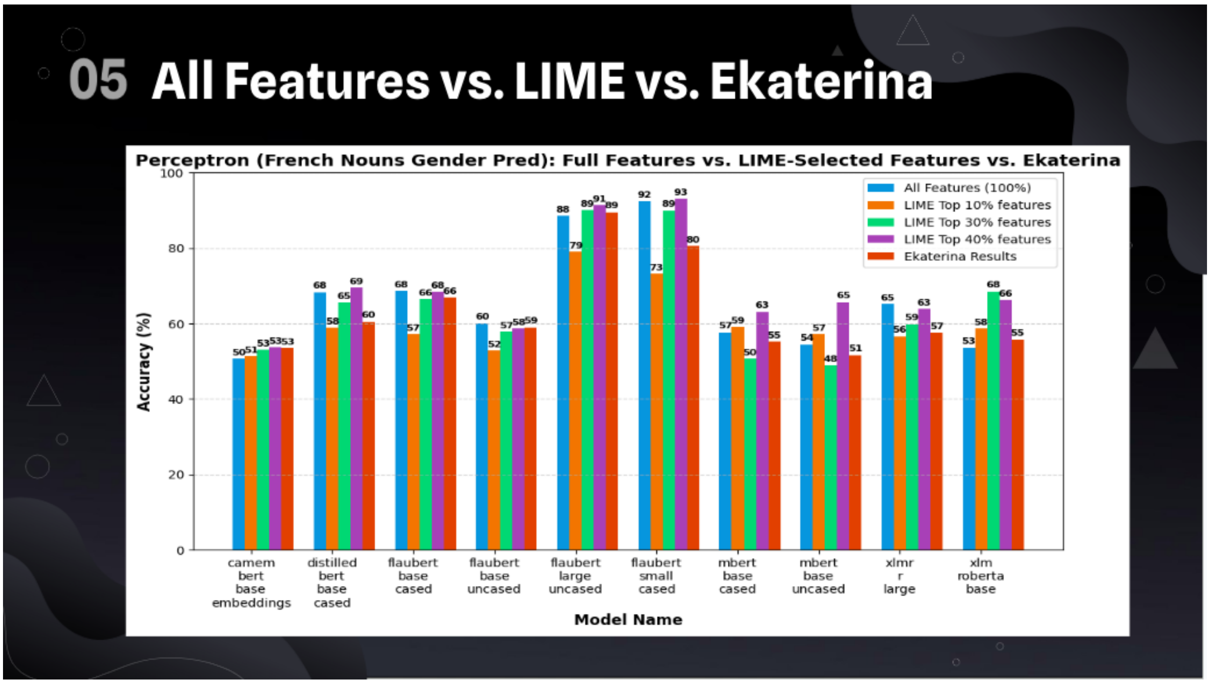
To get global feature importance, we:

4. **Sampled multiple test instances (1% of total dataset)**.
5. **Aggregated feature importance scores** across explanations.
6. **Computed mean absolute importance** per feature over the whole test set.

19. 10% leads to ~80 features. This is very high and I want to know if a very small subset of features can encode gender. What are the results with 0.5, 1, 2, 3, .., 9%?

What is the number of feaures for Ekaterina?

For the future: if results for 0.5, 1, 2, 3, .., 9% are very bad, which model could we use instead of perceptron?



What are the results with 0.5, 1, 2, 3, .., 9%?

- TO-DO: Planned for the Next Experiment

which model could we use instead of perceptron?

- We can test with Logistic Regression, or a some other models initially on a small subset of dataset to find the best performing model

What is the number of features for Ekaterina?

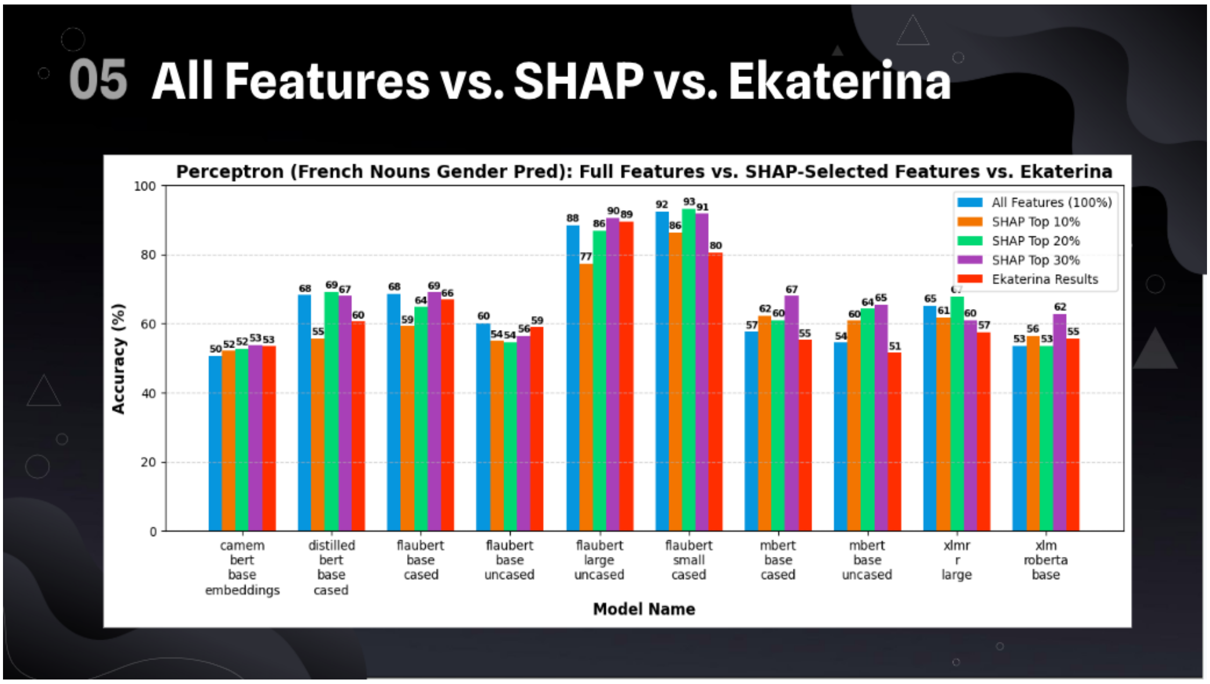
	flau_small_c	flau_base_u	flau_base_c	flau_large_c	cam_base	xlm_large	xlm_base	bert_base_u	distilbert_base	bert_base_c
Perc1	7	14	13	13	22	33	20	26	18	14
Perc5	33	78	65	77	113	166	103	115	101	103
Perc10	67	143	128	144	204	320	232	229	190	238
Perc25	158	333	295	335	443	689	548	510	432	562
Perc50	315	585	527	629	695	971	738	727	704	737
Perc75	450	747	738	900	765	1024	767	764	765	766

Shapes of performance differ strongly from one model to another. How to explain that?

- Some models **encode gender information more effectively**, leading to better performance such as **Monolingual models (Flaubert)**

- Smaller models (Flaubert-small) may **retain only the most essential dimensions,** leading to better generalization.

18. vs 19.



Results between LIME and SHAP are very close except for:

fluabert_small_cased, top10%

mbert_base_cased, top30%

mbert_base_uncased, top30%

xlmr_large top 10, 30

slmr_roberta, 30

how to explain?

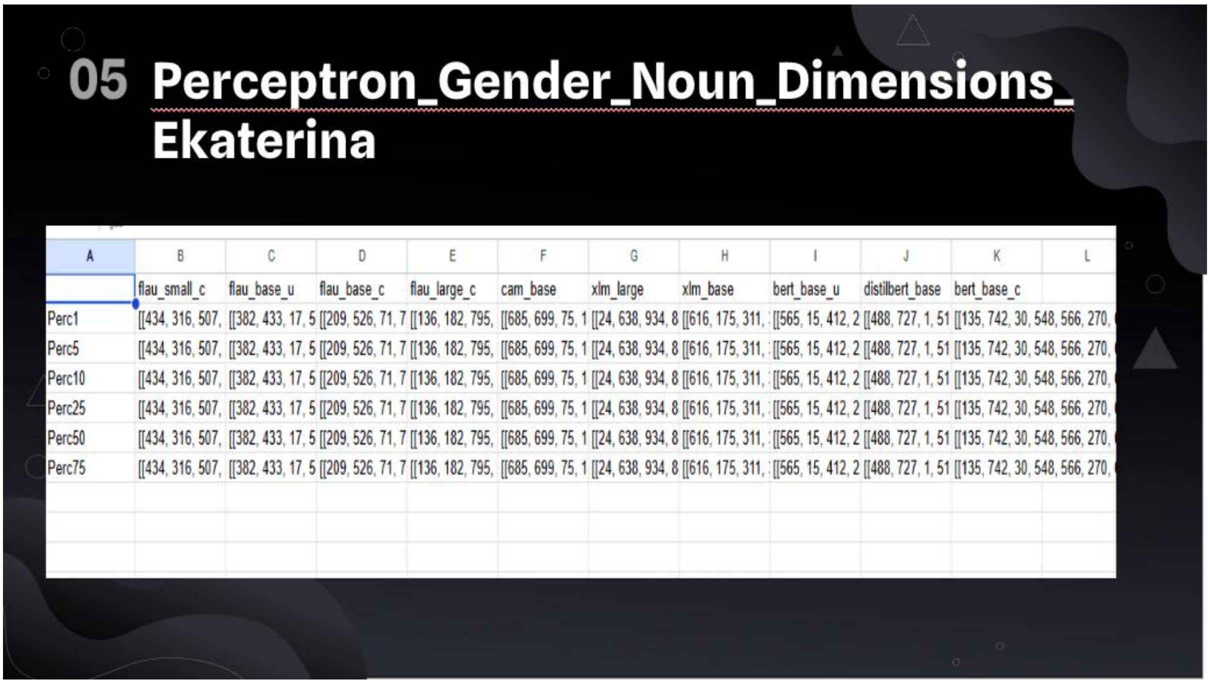
What is the intersection between LIME and SHAP subsets of features?

- TO-DO: Planned for next experiment

- So far we only found intersection between LIME vs Ekaterina and SHAP vs Ekaterina

20.

Not useful



A	B	C	D	E	F	G	H	I	J	K	L
	flau_small_c	flau_base_u	flau_base_c	flau_large_c	cam_base	xlm_large	xlm_base	bert_base_u	distilbert_base	bert_base_c	
Perc1	[[434, 316, 507,	[[382, 433, 17, 5	[[209, 526, 71, 7	[[136, 182, 795,	[[685, 699, 75, 1	[[24, 638, 934, 8	[[616, 175, 311,	[[565, 15, 412, 2	[[488, 727, 1, 51	[[135, 742, 30, 548, 566, 270,	
Perc5	[[434, 316, 507,	[[382, 433, 17, 5	[[209, 526, 71, 7	[[136, 182, 795,	[[685, 699, 75, 1	[[24, 638, 934, 8	[[616, 175, 311,	[[565, 15, 412, 2	[[488, 727, 1, 51	[[135, 742, 30, 548, 566, 270,	
Perc10	[[434, 316, 507,	[[382, 433, 17, 5	[[209, 526, 71, 7	[[136, 182, 795,	[[685, 699, 75, 1	[[24, 638, 934, 8	[[616, 175, 311,	[[565, 15, 412, 2	[[488, 727, 1, 51	[[135, 742, 30, 548, 566, 270,	
Perc25	[[434, 316, 507,	[[382, 433, 17, 5	[[209, 526, 71, 7	[[136, 182, 795,	[[685, 699, 75, 1	[[24, 638, 934, 8	[[616, 175, 311,	[[565, 15, 412, 2	[[488, 727, 1, 51	[[135, 742, 30, 548, 566, 270,	
Perc50	[[434, 316, 507,	[[382, 433, 17, 5	[[209, 526, 71, 7	[[136, 182, 795,	[[685, 699, 75, 1	[[24, 638, 934, 8	[[616, 175, 311,	[[565, 15, 412, 2	[[488, 727, 1, 51	[[135, 742, 30, 548, 566, 270,	
Perc75	[[434, 316, 507,	[[382, 433, 17, 5	[[209, 526, 71, 7	[[136, 182, 795,	[[685, 699, 75, 1	[[24, 638, 934, 8	[[616, 175, 311,	[[565, 15, 412, 2	[[488, 727, 1, 51	[[135, 742, 30, 548, 566, 270,	

21. What is Perceptron? (= Ekaterina?)

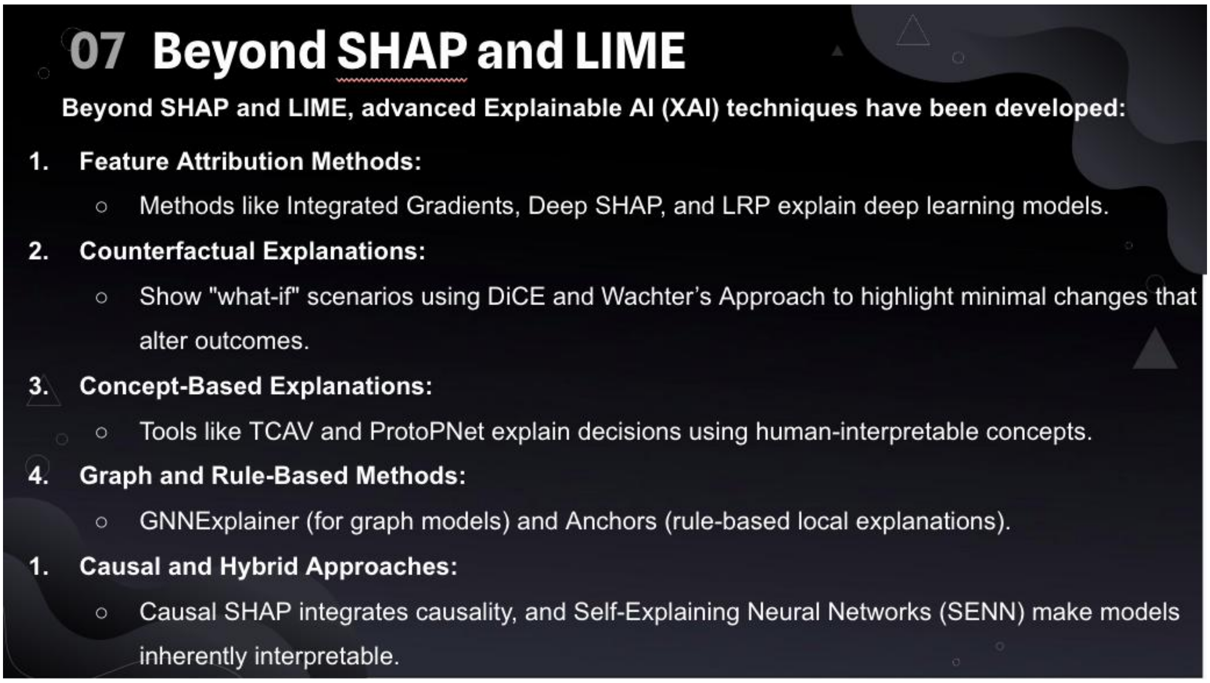
	flau_small_c	flau_base_u	flau_base_c	flau_large_c	cam_base	xlm_large	xlm_base	bert_base_u	distilbert_base	bert_base_c
Perc1	[[434, 316, 507, 245, 100], [434, 162, 245, 316, 377], [245,	[[382, 433, 17, 508, 138, 735, 130], [382, 138, 17, 456, 671, 546,	[[209, 526, 71, 752, 70, 425, 130], [526, 749, 209, 752, 587, 70, 611],	[[136, 182, 795, 972, 15, 575, 1016, 1022, 862, 760], [136, 795, 972, 182,	[[685, 699, 75, 147, 173, 579, 368], [558, 702, 147,	[[24, 638, 934, 849, 173, 939, 153, 136, 224, 885],	[[616, 175, 311, 307, 517, 114, 257], [70, 616, 272,	[[565, 15, 412, 204, 11, 145, 515], [332, 758, 412, 15, 497, 73, 565], [270,	[[488, 727, 1, 51, 207, 76, 302], [727, 1, 270, 488, 673, 353, 278], [196,	[[135, 742, 30, 548, 566, 270, 610], [721, 548, 566, 135, 270, 30, 195], [270, 195,
Perc5	[[434, 316, 507, 245, 100, 192, 377, 117,	[[382, 433, 17, 508, 138, 735, 130, 456, 662,	[[209, 526, 71, 752, 70, 425, 130, 749, 398,	[[136, 182, 795, 972, 15, 575, 1016, 1022,	[[685, 699, 75, 147, 173, 579,	[[24, 638, 934, 849, 173, 939,	[[616, 175, 311, 307, 517, 114,	[[565, 15, 412, 204, 11, 145, 515, 653, 385,	[[488, 727, 1, 51, 207, 76, 302, 394, 75,	[[135, 742, 30, 548, 566, 270, 610, 636, 332,
Perc10	[[434, 316, 507, 245, 100, 192, 377, 117,	[[382, 433, 17, 508, 138, 735, 130, 456, 662,	[[209, 526, 71, 752, 70, 425, 130, 749, 398,	[[136, 182, 795, 972, 15, 575, 1016, 1022,	[[685, 699, 75, 147, 173, 579,	[[24, 638, 934, 849, 173, 939,	[[616, 175, 311, 307, 517, 114,	[[565, 15, 412, 204, 11, 145, 515, 653, 385,	[[488, 727, 1, 51, 207, 76, 302, 394, 75,	[[135, 742, 30, 548, 566, 270, 610, 636, 332,
Perc25	[[434, 316, 507, 245, 100, 192, 377, 117, 186, 162, 250,	[[382, 433, 17, 508, 138, 735, 130, 456, 662, 117, 604, 671,	[[209, 526, 71, 752, 70, 425, 130, 749, 398, 277, 337, 663,	[[136, 182, 795, 972, 15, 575, 1016, 1022, 862, 760, 342,	[[685, 699, 75, 147, 173, 579, 368, 568,	[[24, 638, 934, 849, 173, 939, 153, 136,	[[616, 175, 311, 307, 517, 114, 257, 384,	[[565, 15, 412, 204, 11, 145, 515, 653, 385, 402, 150, 144,	[[488, 727, 1, 51, 207, 76, 302, 394, 75, 301, 37, 512,	[[135, 742, 30, 548, 566, 270, 610, 636, 332, 477, 578, 177,
Perc50	[[434, 316, 507, 245, 100, 192, 377, 117, 186, 162, 250,	[[382, 433, 17, 508, 138, 735, 130, 456, 662, 117, 604, 671,	[[209, 526, 71, 752, 70, 425, 130, 749, 398, 277, 337, 663,	[[136, 182, 795, 972, 15, 575, 1016, 1022, 862, 760, 342,	[[685, 699, 75, 147, 173, 579, 368, 568,	[[24, 638, 934, 849, 173, 939, 153, 136,	[[616, 175, 311, 307, 517, 114, 257, 384,	[[565, 15, 412, 204, 11, 145, 515, 653, 385, 402, 150, 144,	[[488, 727, 1, 51, 207, 76, 302, 394, 75, 301, 37, 512,	[[135, 742, 30, 548, 566, 270, 610, 636, 332, 477, 578, 177,

Conclusion : the intersection si very small, but the performance are quite closed --> we have two distincts subsets with same capacity of prediction!!!!!

- That is a valid point
- Possible Next Step: Can we use Ekaterina dimensions to rerun the model on a small subset to verify our findings so far?

27. 28.

OK, the main part of these new approaches will be described in your final report. Maybe you will use some approaches in your work.



- Planned for the final report as suggested

As said last meeting, next time, you will explain how do SHAP and LIME work and how it is possible to extract "interesting features". I would like you to explain also how you do that with Python (SHAP/LIME package?)

Using SHAP in Python

```
import shap
import numpy as np

# Dictionary to store feature importance for each model
shap_feature_dict = {}

# Loop through trained models and apply SHAP
for name in trained_models.keys():
    print(f"\n--- Processing SHAP Feature Importance for {name} ---")

    # Retrieve stored X_test for the model
    X_test = test_data[name]  # Use the stored test dataset
    perceptron = trained_models[name]  # Use the already trained model

    # Get embedding dimension length dynamically
    embedding_dim = X_test.shape[1]

    # Apply SHAP
    explainer = shap.Explainer(perceptron.predict, X_test)
    shap_values = explainer(X_test, max_evals=embedding_dim * 2 + 1)

    # Compute mean absolute SHAP values per feature
    mean_abs_shap_values = np.abs(shap_values.values).mean(axis=0)

    # Store feature importance for this model
    shap_feature_dict[name] = mean_abs_shap_values

    # Plot SHAP summary for this model
    print(f"\n--- SHAP Feature Importance Plot for {name} ---")
    shap.summary_plot(shap_values, X_test, feature_names=[f"Dim {i}" for i in range(embedding_dim)])
```