NLP PROJECT

Rumor verification - Task 5 - Challenge ChechThat!

Piero Massimo Congiu - 08/07/2024

Challenge description

Rumor verification

- Each rumor has a timeline relative to the period of the publication
- Classify each rumor based on the timeline
- Each rumor can be classified as “SUPPORT”, “REFUTES” OR “NOT ENOUGH INFO”
- All the tweets in the timeline are from verified sources

[{“id”:”…”,
“rumor”:”…”,
“label”:”?”,
“timeline”:[…],
“evidence”:[?]},

…]

Dataset analysis

Content analysis

- The tweets in the dataset contains irrelevant information such as links, hashtags, emojis and tags

```
🔴 | The referees of the Zamalek and Pyramids match in the twenty-eighth round of the Premier League... ⬇️⬇️ #EFA https://t.co/kIo6rrP6aW
```

- Removing this extra content should give us a clean dataset for NLP tools

```
| The referees of the Zamalek and Pyramids match in the twentyeighth round of the Premier League  EFA
```

- Sometimes tweets were not correctly translated and instead there is an error message

```
["https://twitter.com/MOI_Qatar", "1258282285058666496", "ISSUE: couldn't translate"]
```

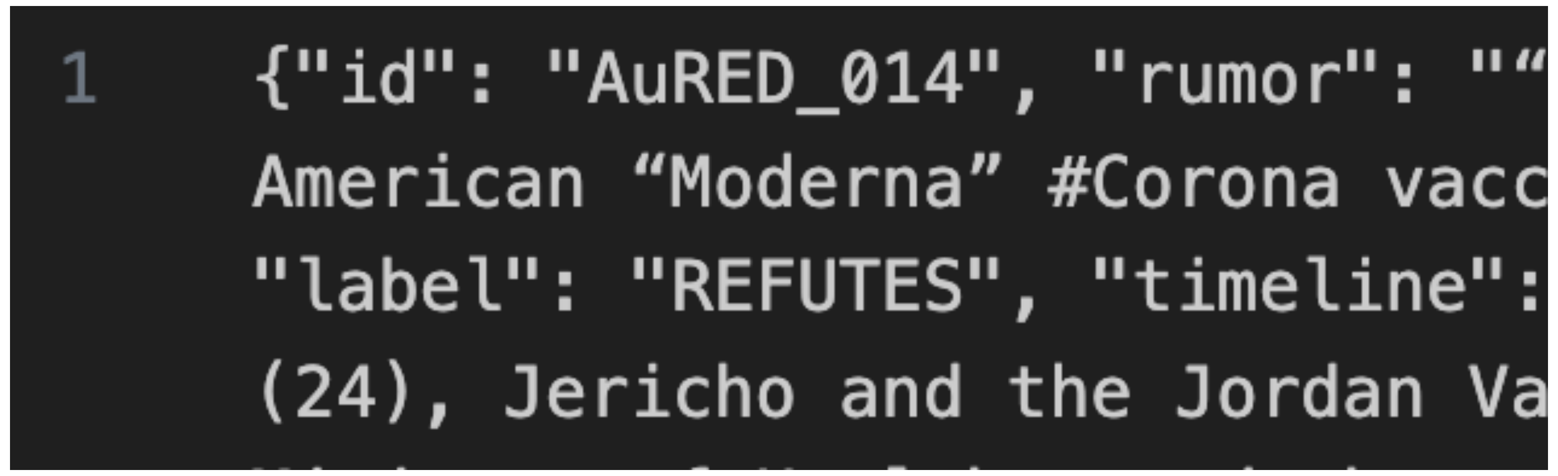
Dataset analysis

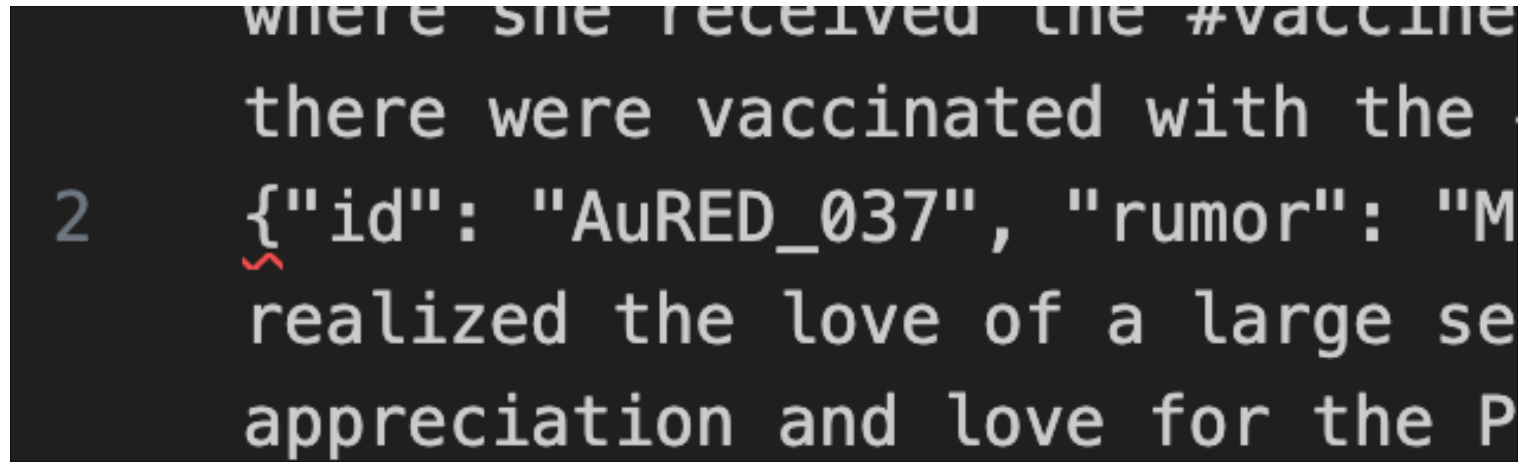
File analysis

- The given training, development and test files don’t follow the usual json format and require a “translation” process

- Files don’t have square brackets for the object separation

- Files don’t have the object separator (comma)





Dataset analysis

File distribution

- Files are divided in dev set, train set and test set
- Only dev and train set are labeled

Dev set

```
[{“id”:”…”,
  “rumor”:”…”,
  “label”:”…”,
  “timeline”:[…],
  “evidence”:[…]},
  …]
```

Train set

```
[{“id”:”…”,
  “rumor”:”…”,
  “label”:”…”,
  “timeline”:[…],
  “evidence”:[…]},
  …]
```

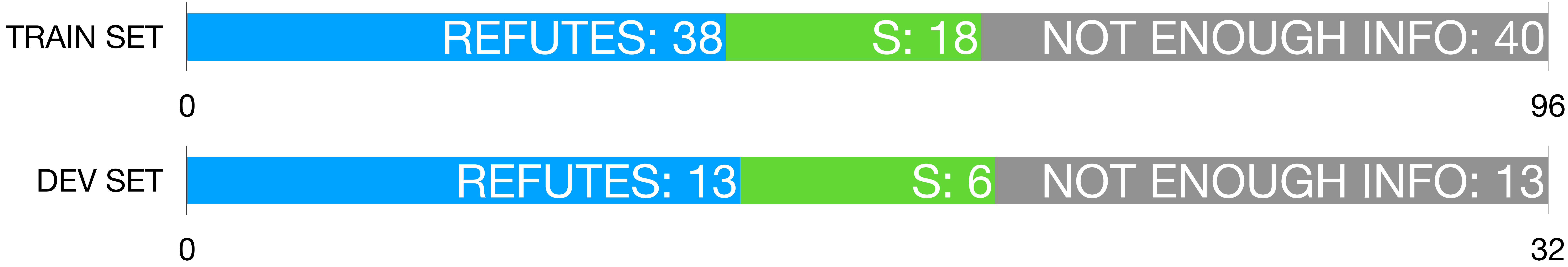
Test set

```
[{“id”:”…”,
  “rumor”:”…”,
  “timeline”:[…]},
  …]
```

First approach

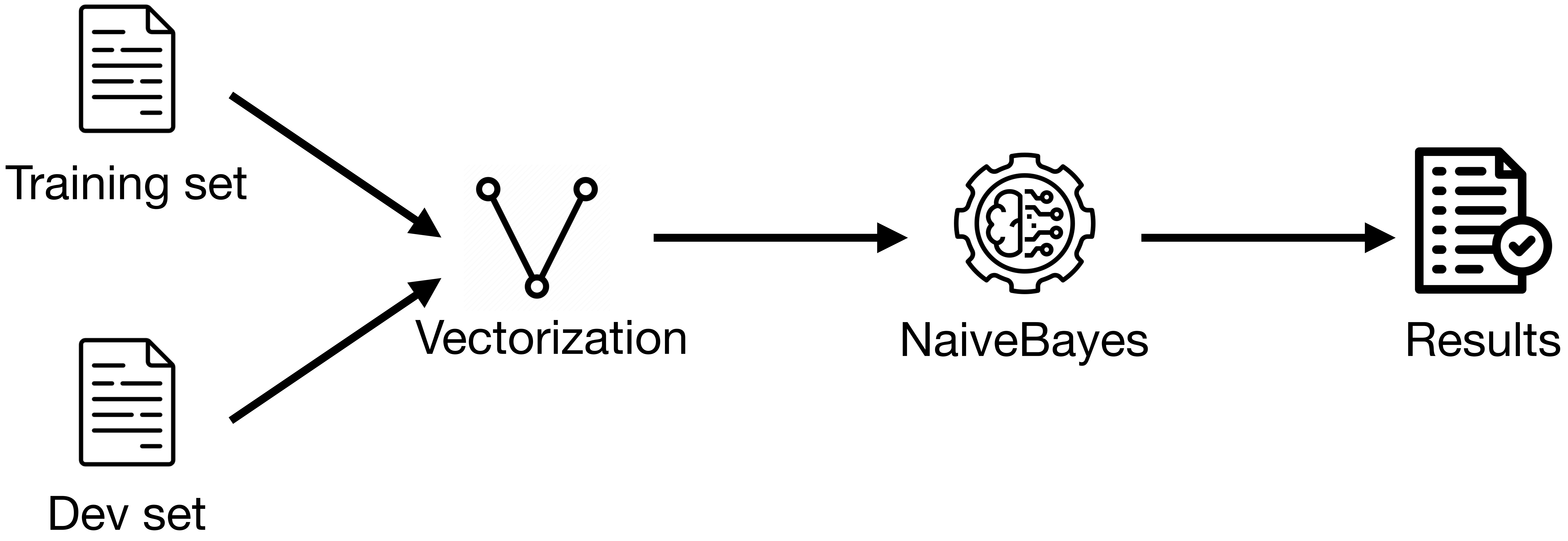
Tokenization and Machine Learning

- The challenge is designed for the use of Transformers
- Use of traditional ML require going off the rails
- Use the TF-IDF for a unique representation of the tweets
- Use of the training set labeled rumor for the training
- Naive Bayes was chosen due to the scarcity of labeled examples



First approach

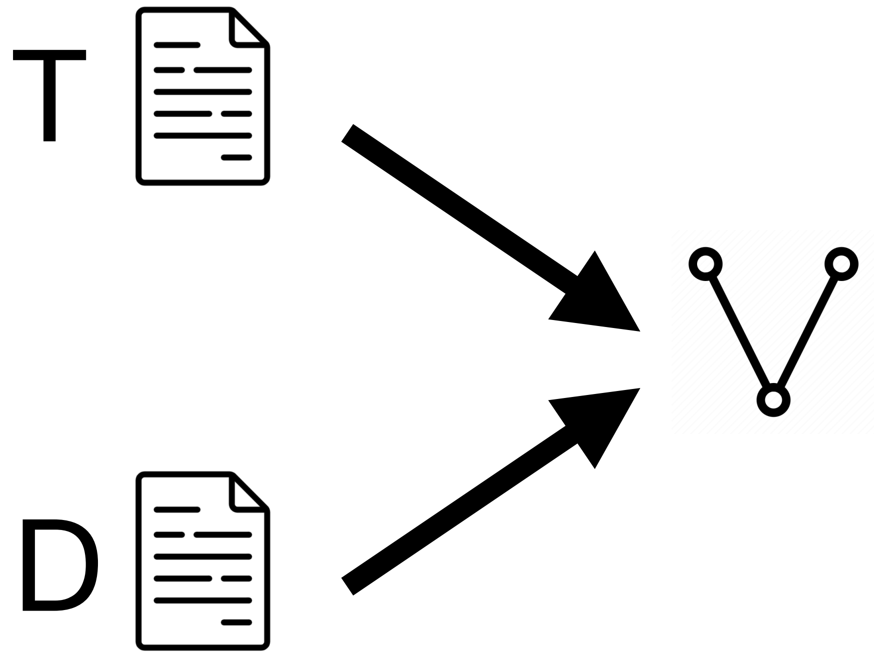
Tokenization and machine learning



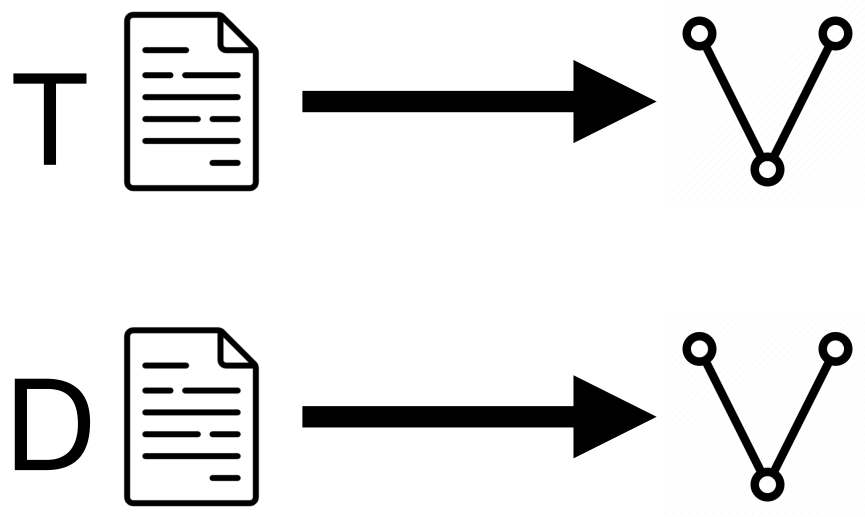
First approach

Tokenization

- The vectorizer used is the Sklearn TfidfVectorizer
- The training and test data need to have the same amount of features, so they need to be vectorized together
- Vecorizing them separately will lead to a different amount of features and the prediction will not be possible



Same number of features



Different number of features

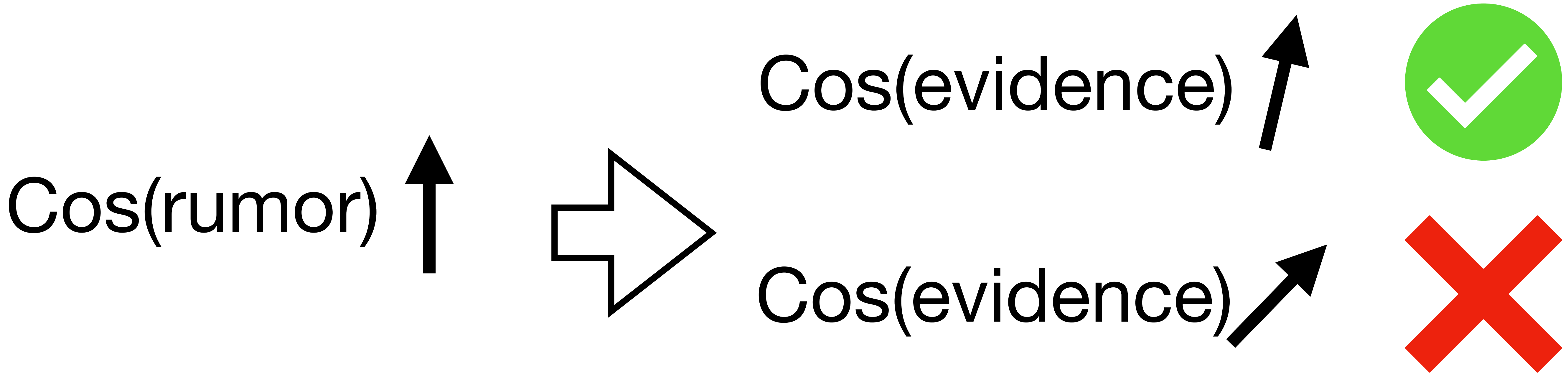
First approach

Naive Bayes

- As said before not having a huge number of training examples lead us to usa a very bias ML technique
- This technique, vectorization + ML classifier is a well known approach. Not perfectly suited for this challenge
- Model can’t be trained on the timeline because all the tweets in a timeline are verified, there is no class subdivision
- With the vectorization and cosine similarity we can find tweets in the timeline with the same context, but can’t say if they agree, disagree

First approach
Evdence retrieval

- The task requires to label the rumor and to recover at most five tweet from the timeline that supports the chosen label
- Using this approach the timeline isn't directly used during the classification
- Another round of vectorization is used to find the timeline tweets that are the most similar (context wise) to the rumor. These tweets are used as evidence



Secondo approach

Prompt engineering and Transformer models

- Unlike the previous technique, this models can be prompted to include the timeline and give a much more accurate answer
- Three models have been chosen, one small (distilbert-base-uncased-mnli), one medium (xlm-roberta-large-xnli) and one large (bart-large-mnli)
- All these models can perform zero-shot-classification. Where a prompt can be labeled given a set fo labels

Second approach

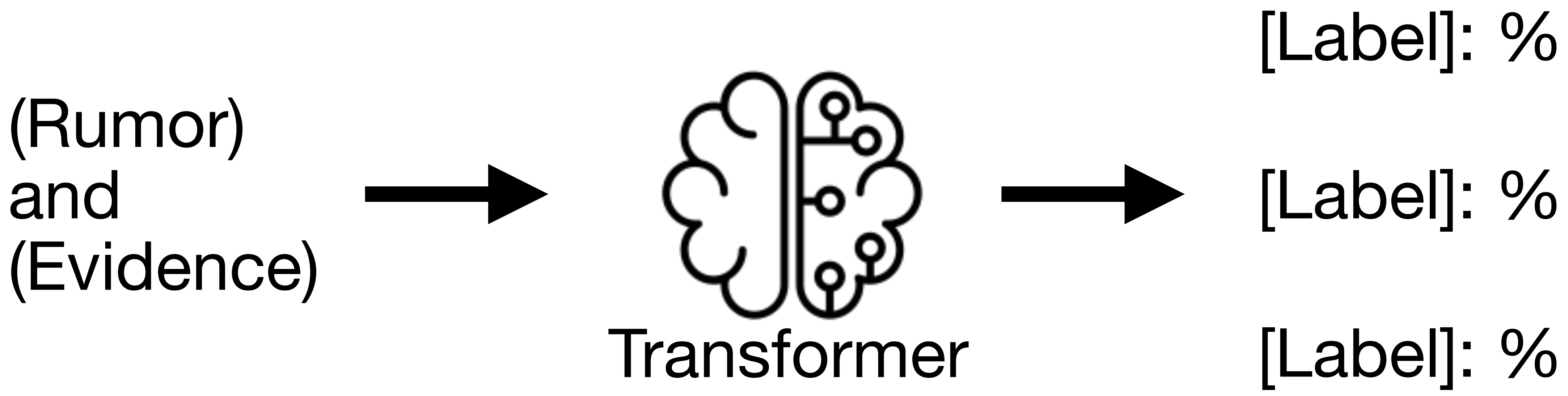
Transformer models

- Typeform/distilbert-base-uncased-mnli is the smallest, has 66 millions parameters and 6 layers.
- Joeddav/xlm-roberta-large-xnli is the medium size model, with 1.5 billions parameters and 24 layers. Despite the name, it's considered a medium size model.
- Facebook/bart-large-mnli has 137 billion parameters and 96 layers.

× FACEBOOK AI

Second approach

Prompt engineering



- When the model is run on a prompt it returns a probability distribution of the given labels
- Each timeline tweet is prompted together with the rumor in the following way: (rumor) and (timeline tweet)
- Foreach tweet in the timeline the system memorize the probability distribution of the labels
- The most probable label is chosen as the rumor label
- The most probable tweets (according to the label) are chosen as evidence

Second approach

Labels choosing

[“SUPPORTS”,”REFUTES”,”NOT ENOUGH INFO”]

[“AGREE”,”DISAGREE”,”OFF TOPIC”]

[“CONFIRM”,”OPPOSE”,”NOT ENOUGH INFO”]

```
Bart(test_prompts, candidate_labels=labelsBart)
✓ 0.4s
{'sequence': '(Naturalization decree in preparation: Lebanese passports for
 'labels': ['OPPOSE', 'NOT ENOUGH INFO', 'CONFIRMS'],
 'scores': [0.5613054037094116, 0.36275196075439453, 0.07594260573387146]}
```

```
Bart(test_prompts, candidate_labels=labelsBart)
✓ 0.4s
{'sequence': '(Naturalization decree in preparation: Lebanese passports for
 'labels': ['OPPOSE', 'CONFIRMS', 'NOT ENOUGH INFO'],
 'scores': [0.45437753200531006, 0.3421109914779663, 0.20351149141788483]}
```

```
Bart(test_prompts, candidate_labels=labelsBart)
✓ 0.5s
{'sequence': '(Some reporters says that the Pope is dead because of some na
 'labels': ['CONFIRMS', 'NOT ENOUGH INFO', 'OPPOSE'],
 'scores': [0.7761325240135193, 0.12497550994157791, 0.09889192879199982]}
```

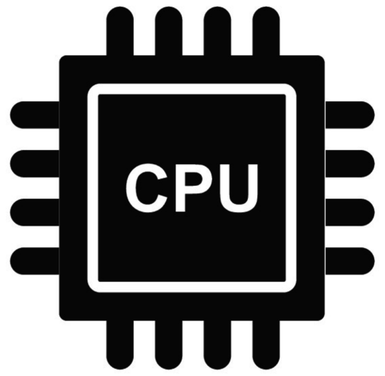
[“VALIDATES”,”DISAGREE”,”NOT ENOUGH INFO”]

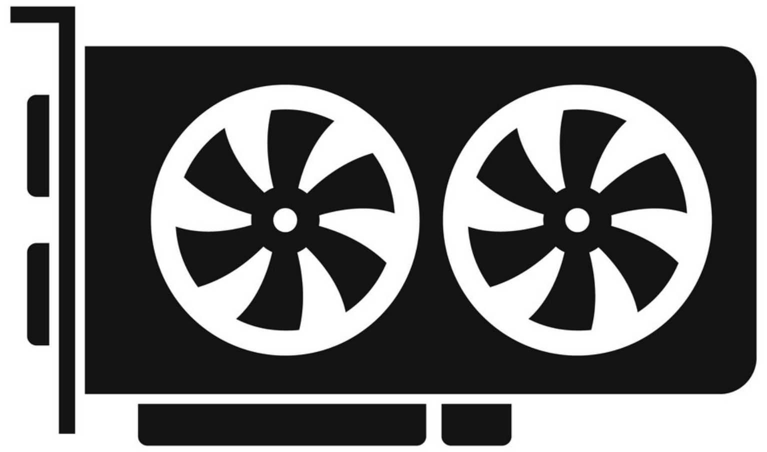
```
Roberta(test_prompts, candidate_labels=labelsRoberta)
✓ 2.5s
{'sequence': '(Naturalization decree in preparation: Lebanese passports for
 'labels': ['DISAGREE', 'VALIDATES', 'NOT ENOUGH INFO'],
 'scores': [0.7227831482887268, 0.16914376616477966, 0.10807310044765472]}
```

```
Roberta(test_prompts, candidate_labels=labelsRoberta)
✓ 0.4s
{'sequence': '(Some reporters says that the Pope is dead because of some nat
 'labels': ['VALIDATES', 'DISAGREE', 'NOT ENOUGH INFO'],
 'scores': [0.8322266340255737, 0.11874688416719437, 0.04902655258774757]}
```

```
Roberta(test_prompts, candidate_labels=labelsRoberta)
✓ 0.3s
{'sequence': '(Some reporters says that the Pope is dead because of some na
 'labels': ['VALIDATES', 'DISAGREE', 'NOT ENOUGH INFO'],
 'scores': [0.42388516664505005, 0.3674059212207794, 0.20870891213417053]}
```

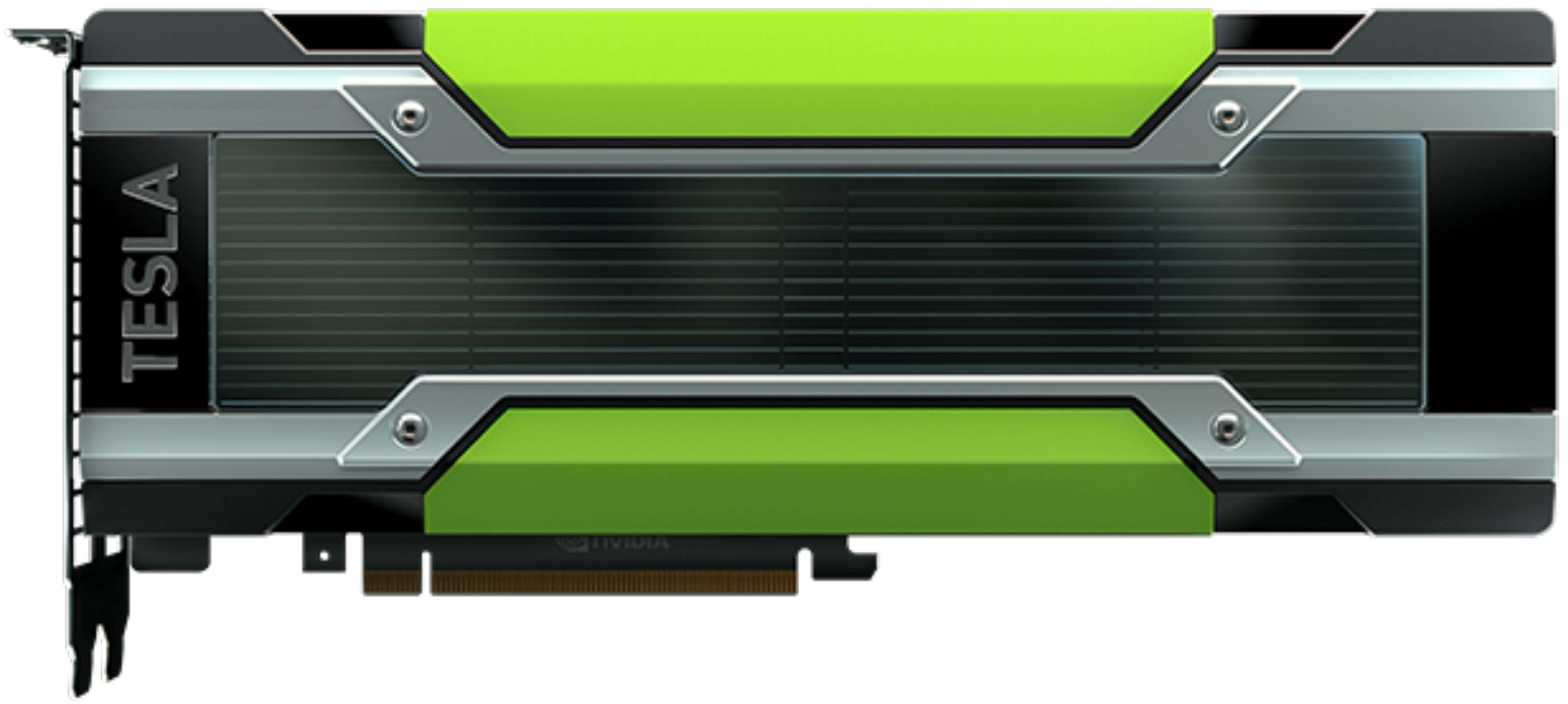
Second approach
Performance





- Compared to classic ML, transformers models are much more resource intensive
- The bigger the model, the more precise it will be and the more resource it will use
- Small models like distilBert use at most 400Mb of memory, more complex model like Bart use more than 2Gb
- Running these models on CPU is almost impossible. First runs on CPU acceleration required some tricks.

Second approach
Performance



- These models have been run using the transformer library from Hugging Face
- Pipeline is a universal model interface that uses CPU as default
- Because of that the first implemented classifier used a threshold method to reduce the number of execution of the model
- The device attribute let the programmer choose a graphic accelerator to run the model (0 should be fine)
- Distributed cloud computing platform such as Google Colab don’t support the pipeline/device combo and require PyTorch

Second approach
Performance

- All models runs at in acceptable time on GPU acceleration
- Distilbert, the smallest model, run on the entire dataset in around three minutes
- Roberta, the medium size model, run in around eight minutes
- Bart, the largest model run in around 11 minutes
- More performance could be extracted if prompts were run in parallel instead of sequentially. As suggested by the pipeline itself

```
You seem to be using the pipelines sequentially on GPU. In order to maximize efficiency please use a dataset
```

Results

Challenge baseline and leaderboard

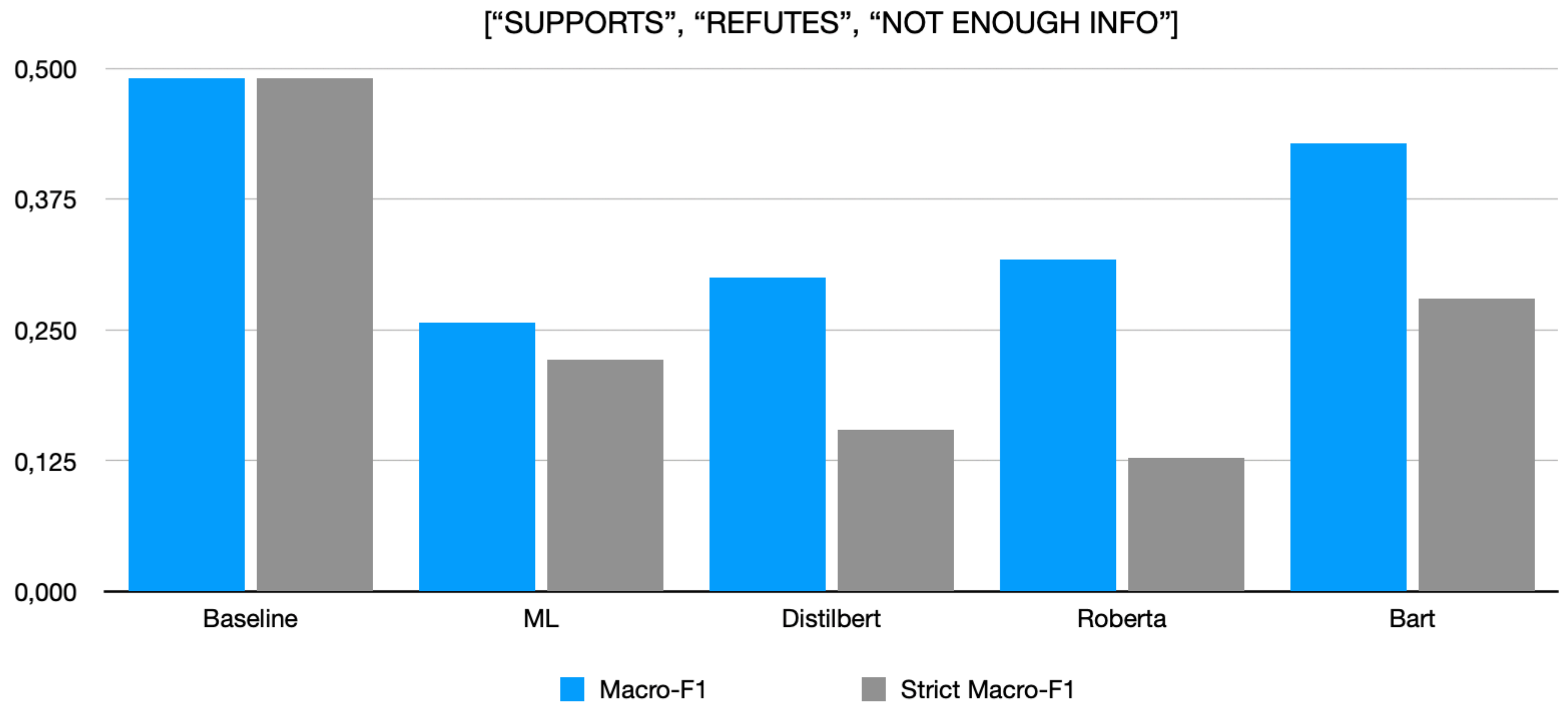
- Authors of the challenge used fine-tuned BERT for the evidence retrieval and the claim verification
- Metrics chosen for the rumor verification are Macro-F1 and Strict Macro-F1
- Metrics chosen for the evidence retrieval are MAP and R@5

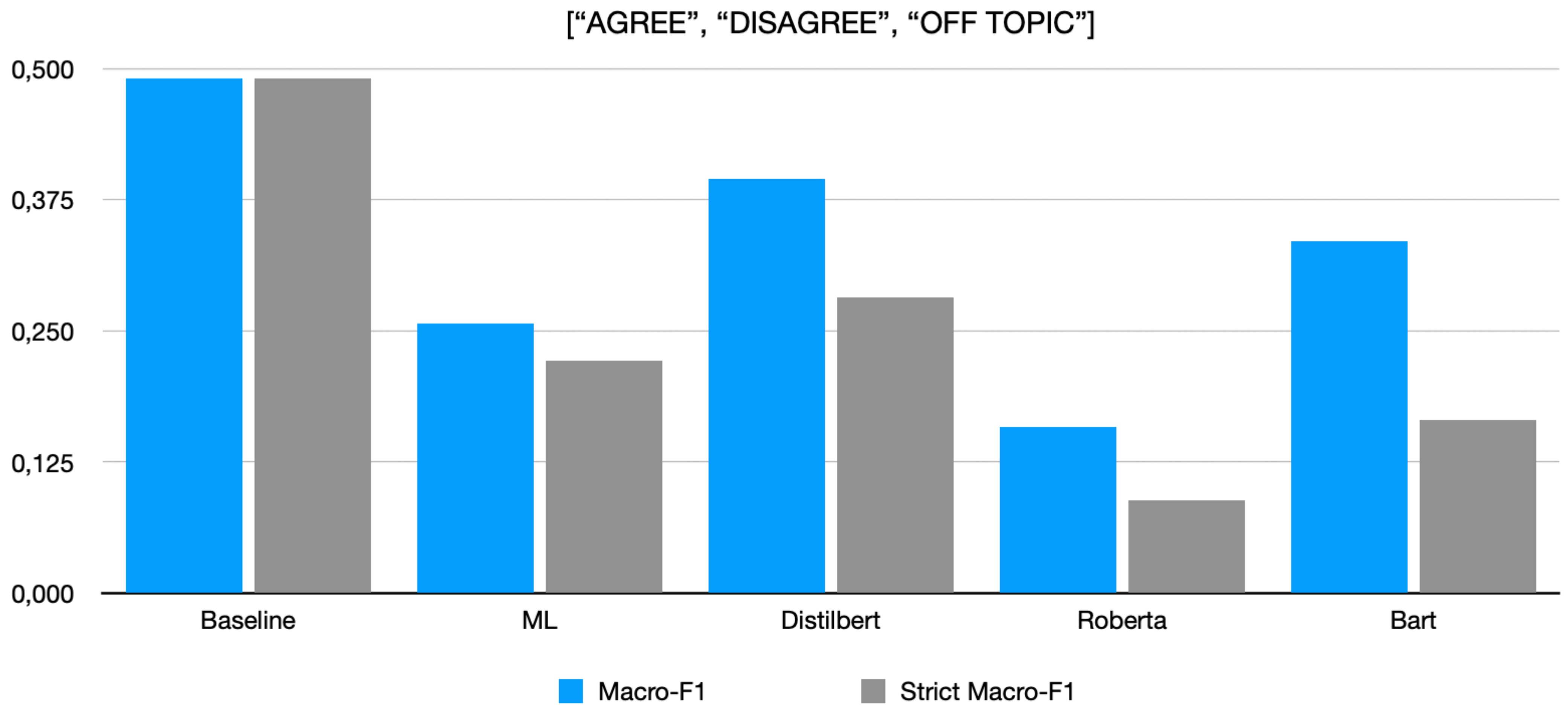
VERIFICATION	Macro-F1	S Macro-F1
Baseline	0,49	0,49
ML	0,257	0,222
Distilbert	0,395	0,282
Roberta	0,317	0,128
Bart	0,428	0,280

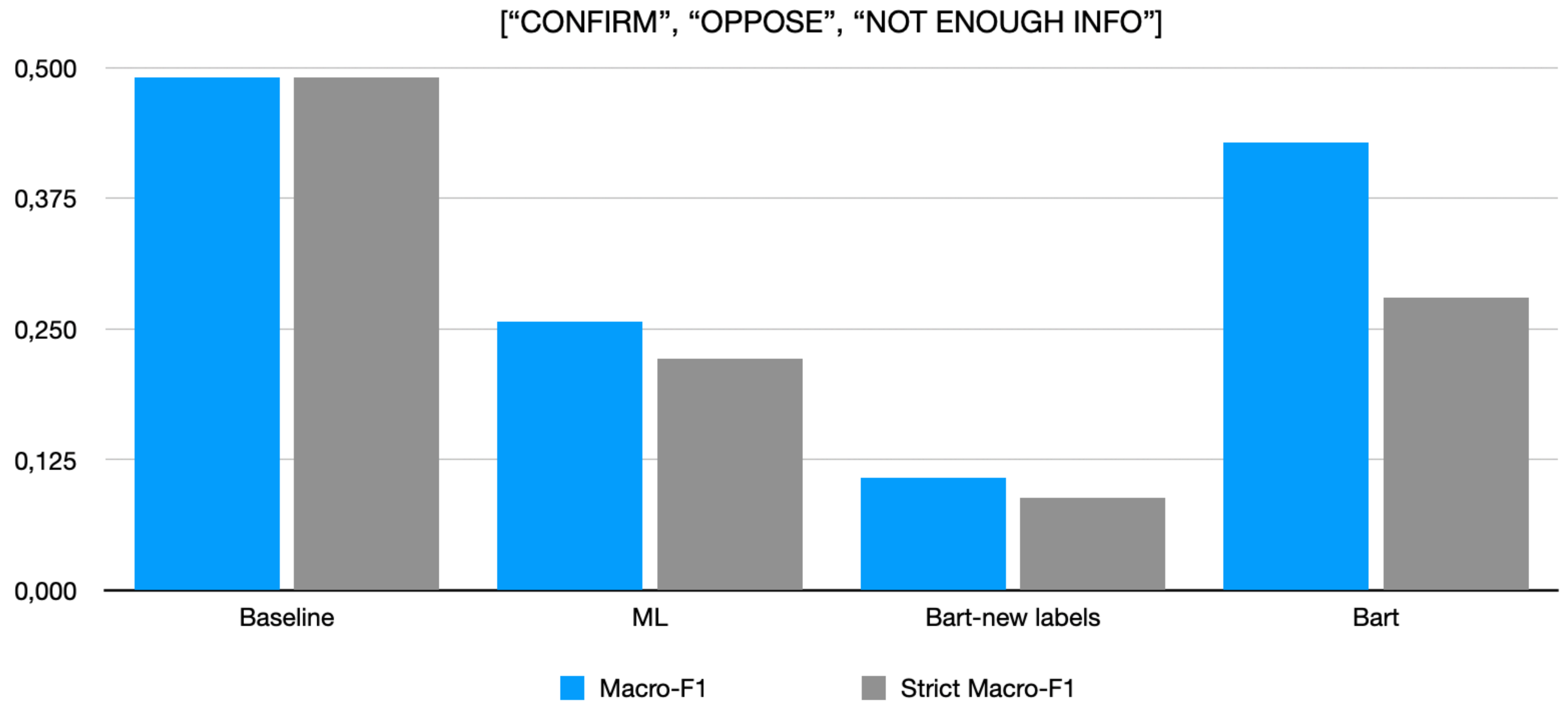
EVIDENCE R.	MAP	R@5
Baseline	0,335	0,445
ML	0,196	0,298
Distilbert	0,283	0,405
Roberta	0,158	0,228
Bart	0,373	0,449

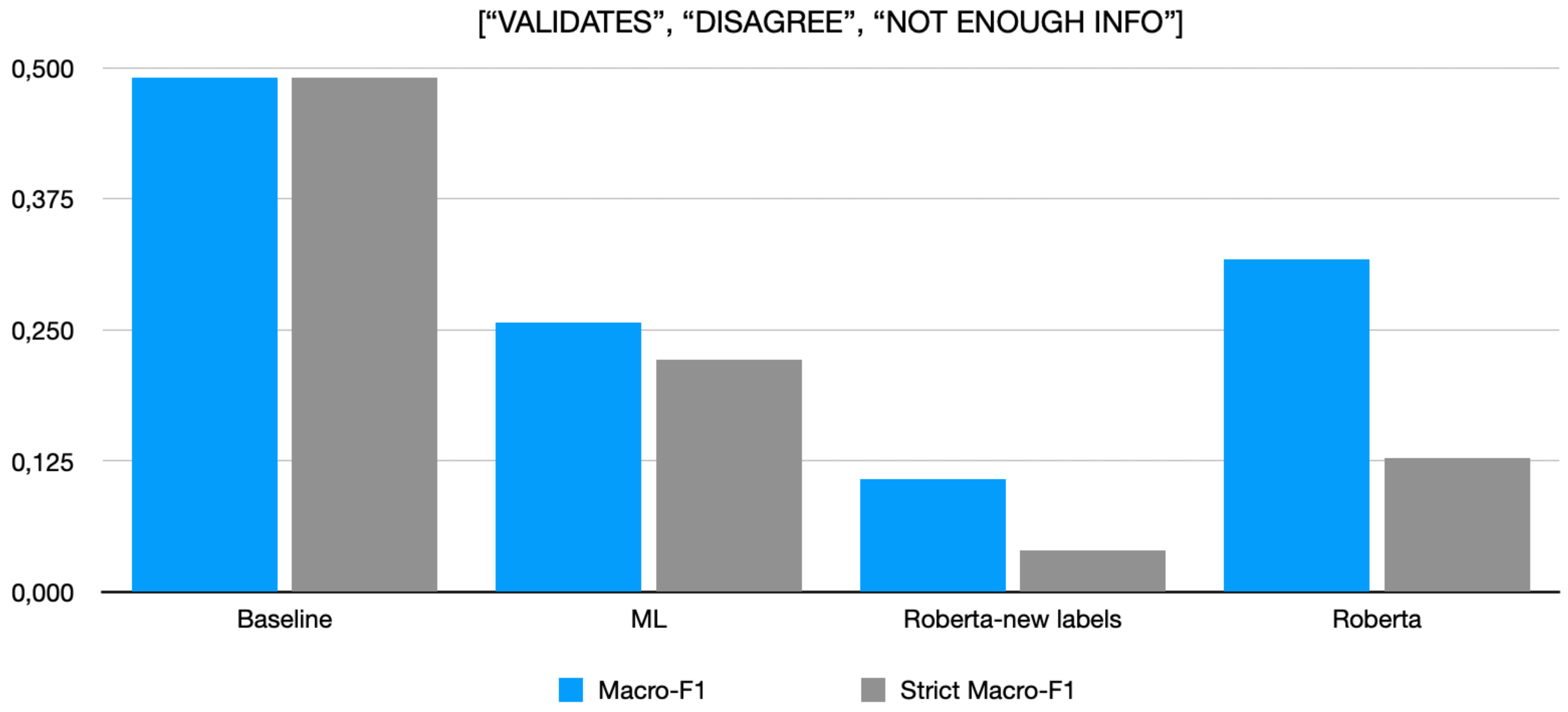
Verification Results

Labels - Macro F-1 and Strict Macro F-1

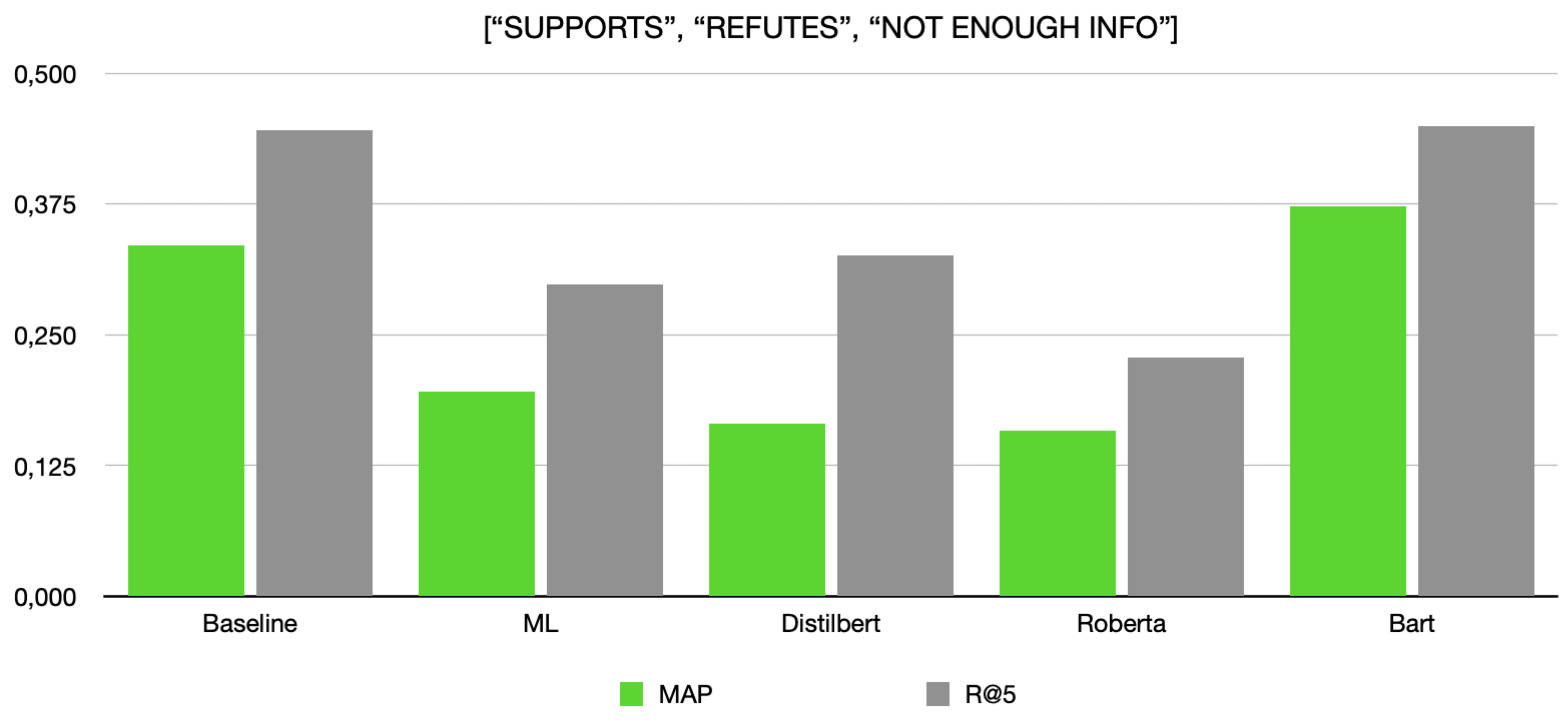


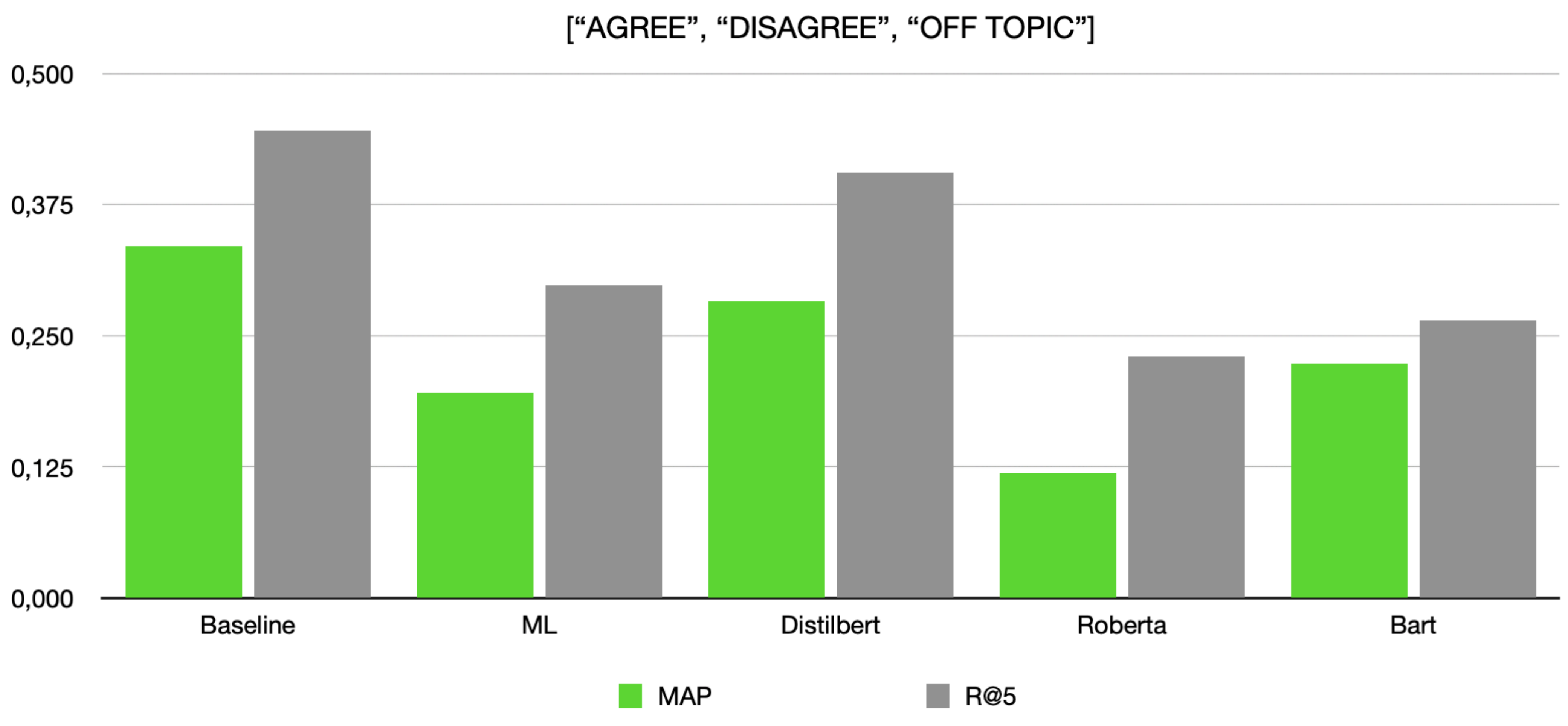


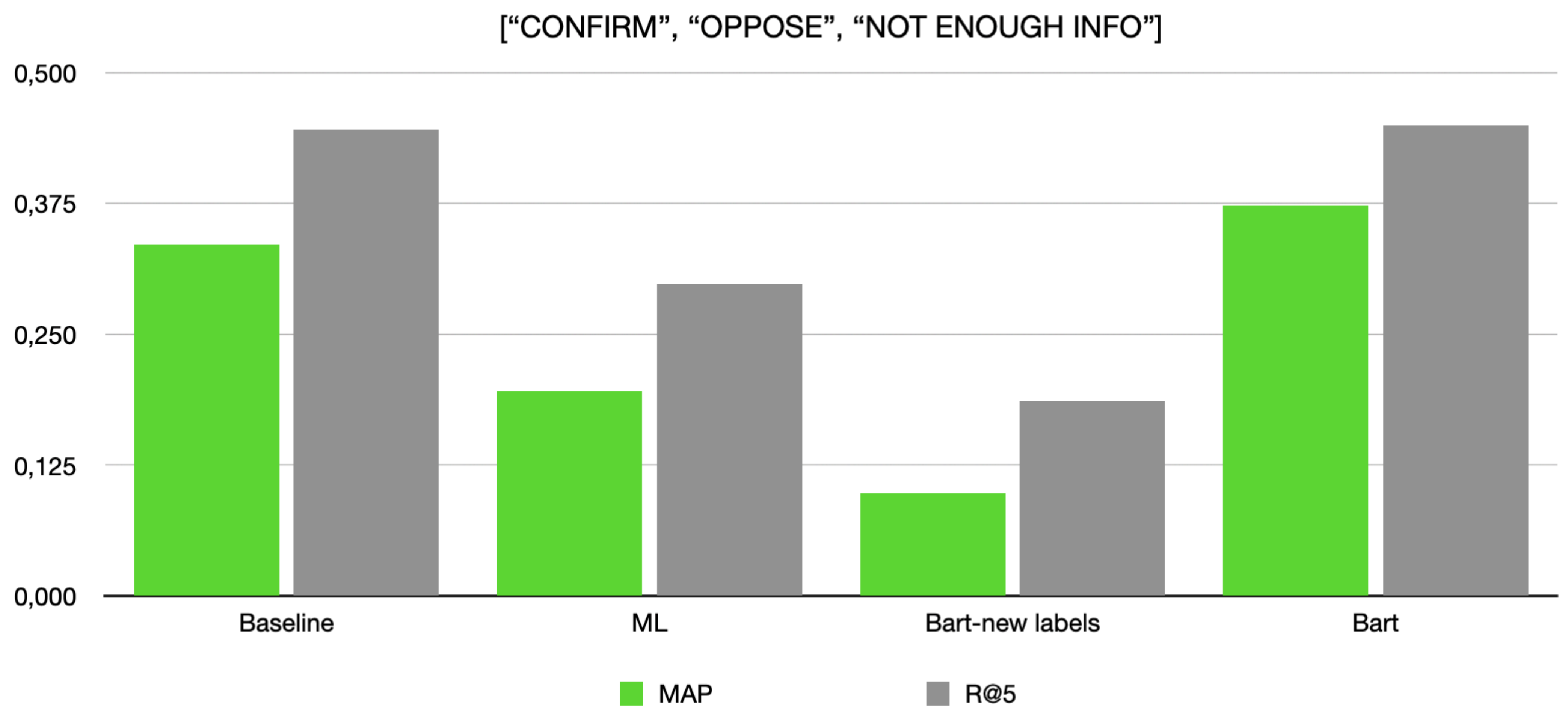


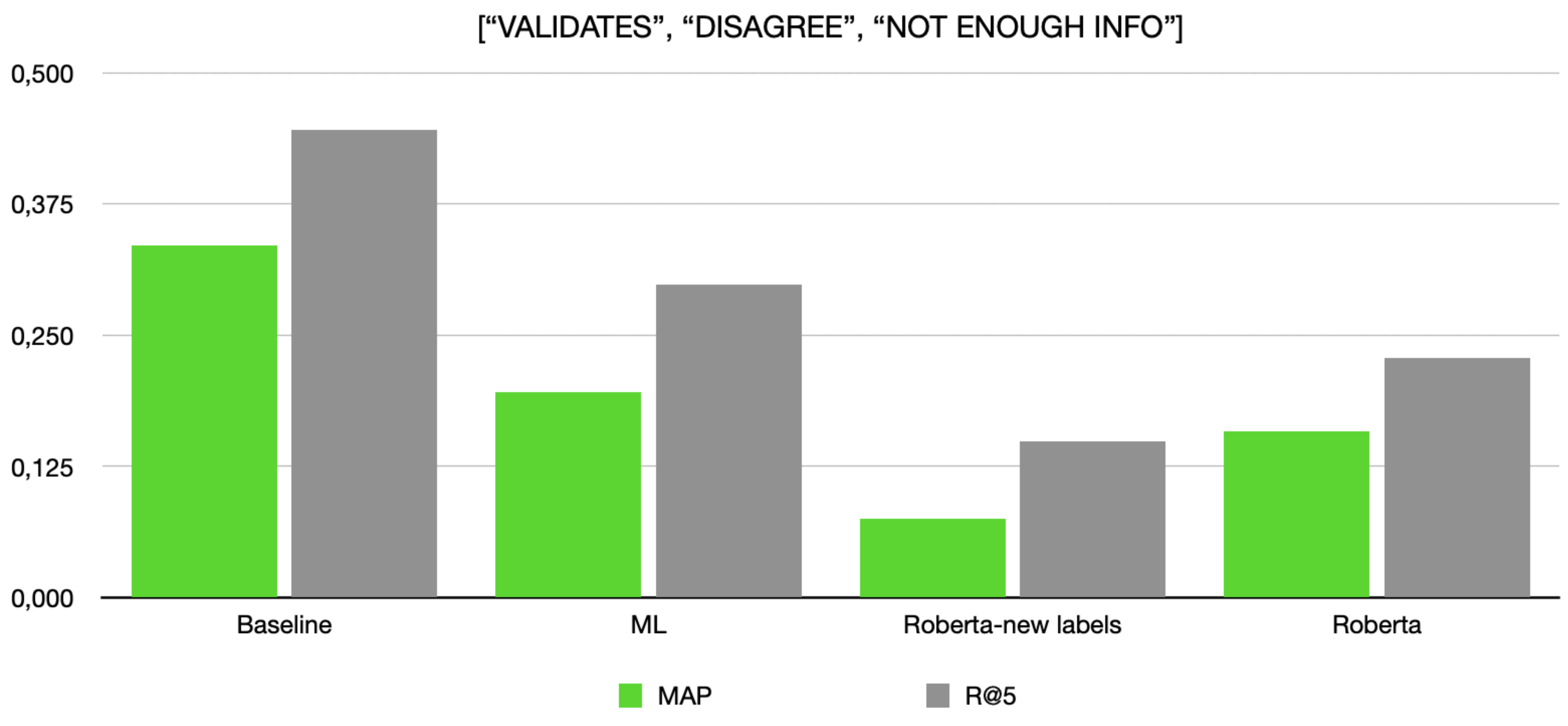


Evidence Results
Labels - MAP and R@5



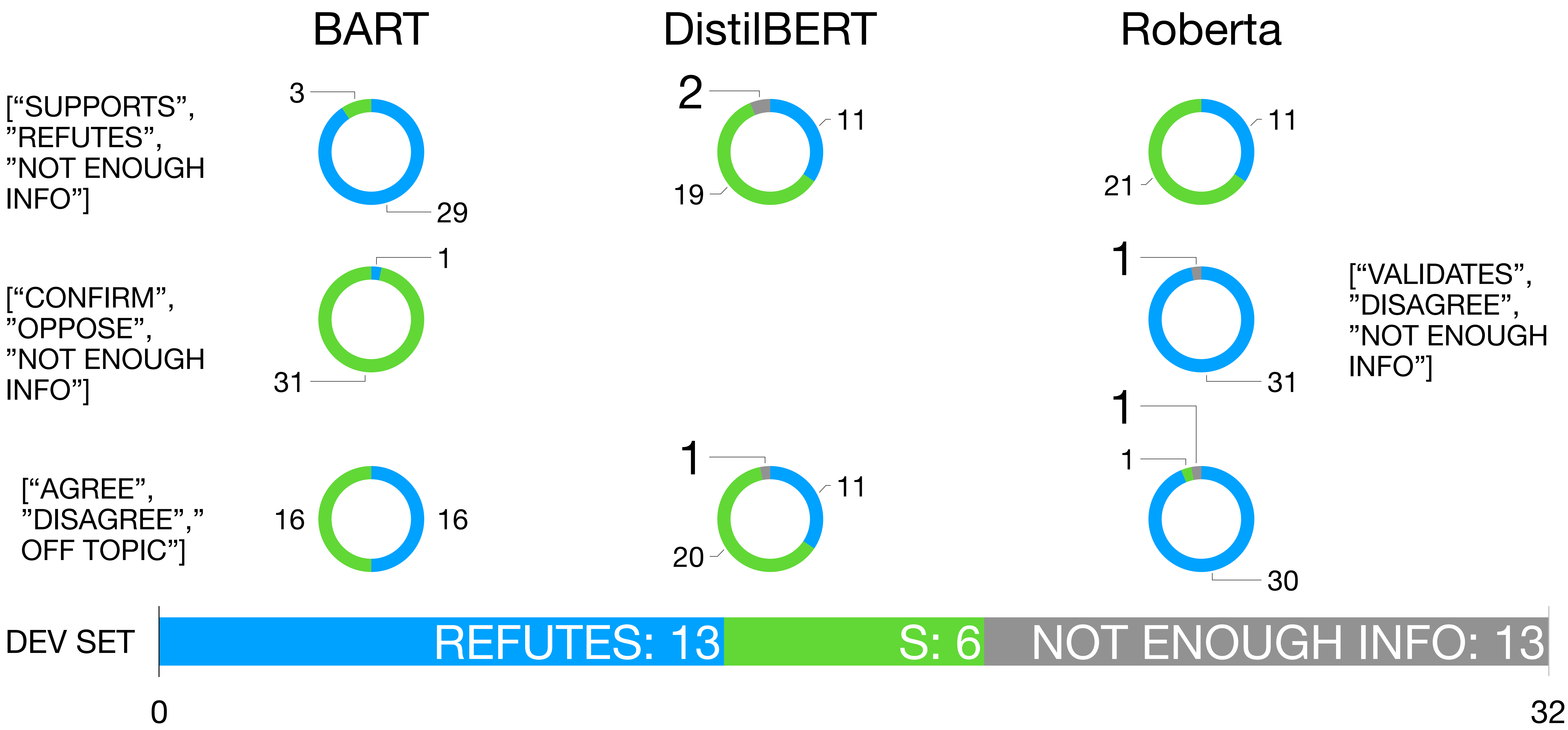






Results

Model bias



Improve approaches
Classic ML

- Stemming words may increase performance
- Use dense vectors
- The class are mutually exclusive, create multiple class vs class classifier
- Check in the current rumor comes from a verified source

Improve approaches
Transformer models

- Fine tuning of the best performing model
- Search other labels
- Find another way to prompt the model
- Search other, more specific, models

Fin.