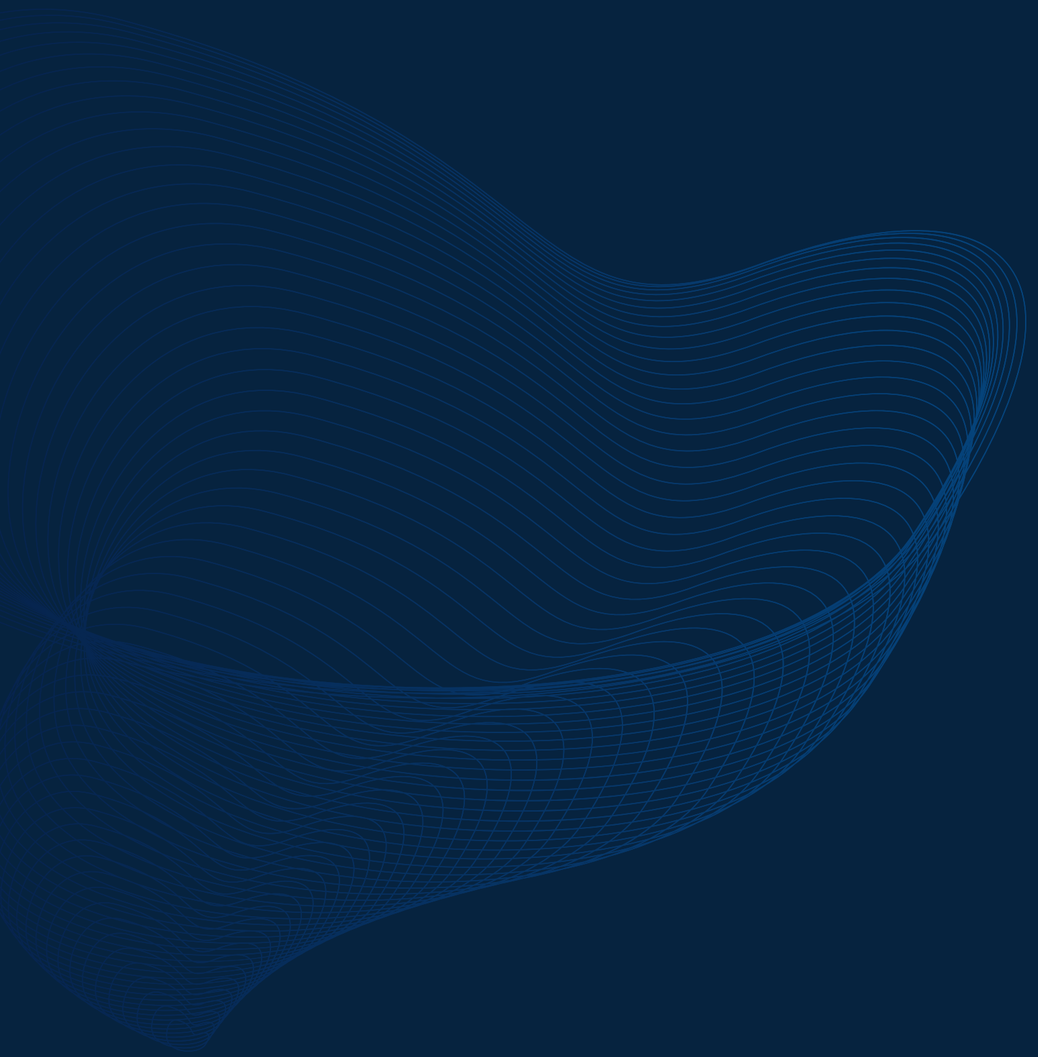
STRUCTURED QUERY VERBALIZATION
WITH LARGE LANGUAGE MODELS

By Patrik Valach & Louisa Siebel



Overview

- Recap
- Our Approach & Preliminary results
  - Loading data and mapping values
  - Using the LLM for translation
  - Manual inspection
  - Using the LLM for comparison
  - BLEU
  - BERT
- Current Issues and Next Steps
- Current results summary

Recap 1

- Problem:
  - There exists a lot of research on the task of transforming natural language queries to SQL queries using LLM, BUT there is not enough training data
- Goal:
  - Create this training data by getting an LLM to take a SPARQL query and generate it’s natural language counterpart
- Dataset:
  - Created to answer natural language questions
  - 30 000 Pairs

question string · *lengths* 1 976 ⌀	sparql_wikidata string · *lengths* 37 355
What periodical literature does…	select distinct ?obj where { wd:Q188920 wdt:P2813 ?obj . ?obj…
Who is the child of Ranavalona I's…	SELECT ?answer WHERE { wd:Q169794 wdt:P26 ?X . ?X wdt:P22 ?answer}
Is it true Jeff_Bridges…	ASK WHERE { wd:Q174843 wdt:P106 wd:Q1804811 . wd:Q174843 wdt:P106…

paraphrased_question string · *lengths* 1 43k
What is Delta Air Line's periodical…
What is the name of Ranavalona I's…
Are Jeff Bridges and Lane Chandler both…

Recap 2

- Models to consider:
  - llama: 8 and 70B sizes, Optimized for dialogue, Text only
- Evaluation Methods:
  - Manual Inspection:
    - 0/1 matrix
    - Either the SPARQL query is converted to the NL question correctly or not
  - Bert:
    - converts text to vector
    - semantically indifferent text will give vectors that are close to each other
  - Bleu:
    - how many transformations are needed to do to get the exact same text

Loading data and mappping values

- Access the lc_quad dataset to work on
- Find ambiguous entities and properties

sparql wikidata:

select distinct ?obj where { wd:Q188920 wdt:P2813 ?obj . ?obj wdt:P31 wd:Q1002697 }

How can we find these ambiguities?

```
def map_wikidata_to_natural_language(sparql_query:str) -> str:
    client = Client()

    translatable_parts = find_translatable_parts(sparql_query)

    for i in translatable_parts:
        try:
            entity = client.get(i.split(":")[1], load=True)
            sparql_query = sparql_query.replace(i, str(entity.label))
        except Exception as e:
            print(e, end=" ")
            print(i)
            return None
    return sparql_query
```

- Query the knowledge graph of the dataset for these entities and properties
- Map them trough the Wikidata client into their natural language form

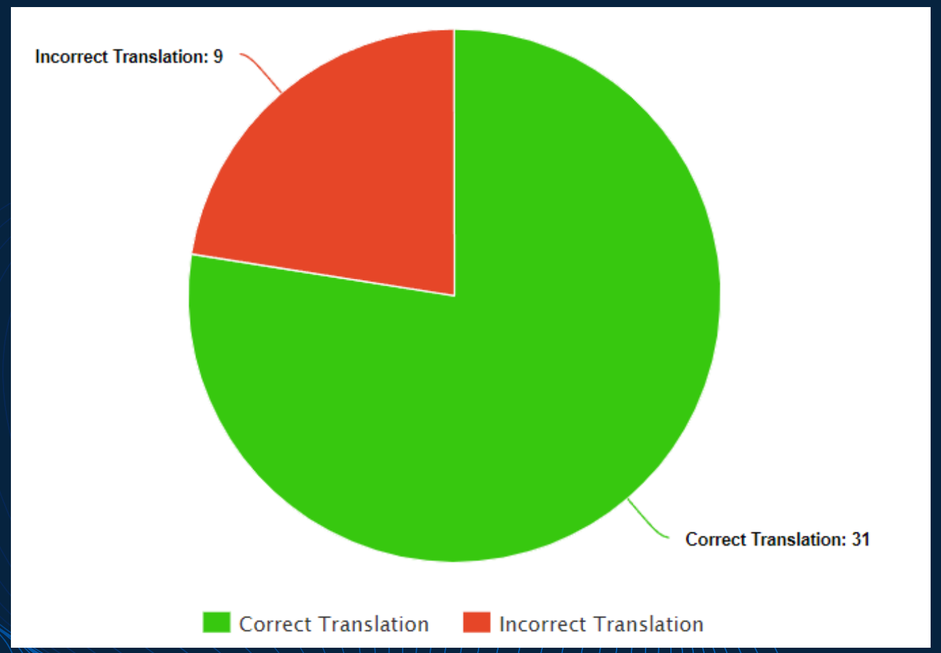
Using the LLM for translation

- Model is llama3 (smaller one 8B)
  1. Give the model an explanation of the exact assignment
  2. Provide a combination of the SPARQL query and the paraphrased question as examples
  3. Prompt llama to translate the SPARQL query input into a natural language question


```
prompt_translate = """<|begin_of_text|><|start_header_id|>          system                                  <|end_header_id|>
You are a AI translator for converting sparql queries into normal natural language questions          <|eot_id|><|start_header_id|>user<|end_header_id|>
Translate the sparql query into natural language; formulate the response as a question and respond in one sentence only with the translation itself:
PROMPT                    SELECT DISTINCT ?sbj ?sbj_label WHERE ....                                  <|eot_id|><|start_header_id|>assistant<|end_header_id|>
Paraphrased question      Let me know physical marvel whose title has the word surface in it.         <|eot_id|><|start_header_id|>user<|end_header_id|>
......Repeat multiple times
Actual prompt to translate
"""
```

Manual Inspection

- Do a manual comparison to check the translation
- Translation: 31 correct out of 40



Dataset:
Did Brittany Murphy have USA citizenship?

Translation:
Was Brittany Murphy a citizen of the United States of America?

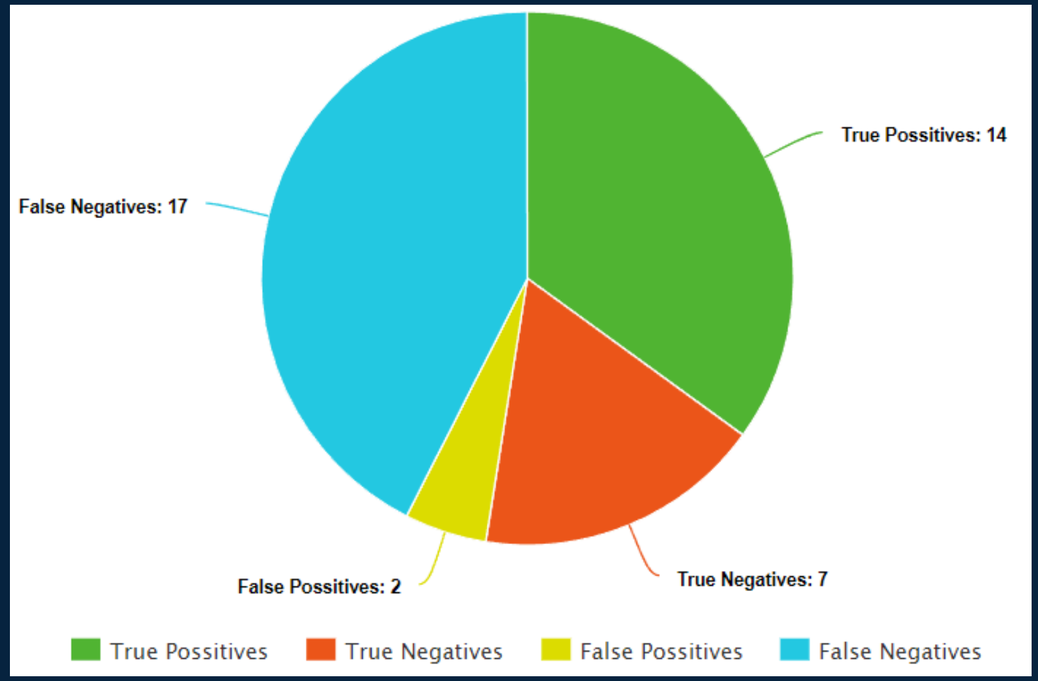
Are these Semantically the same?

Yes

No

Using the LLM for comparison

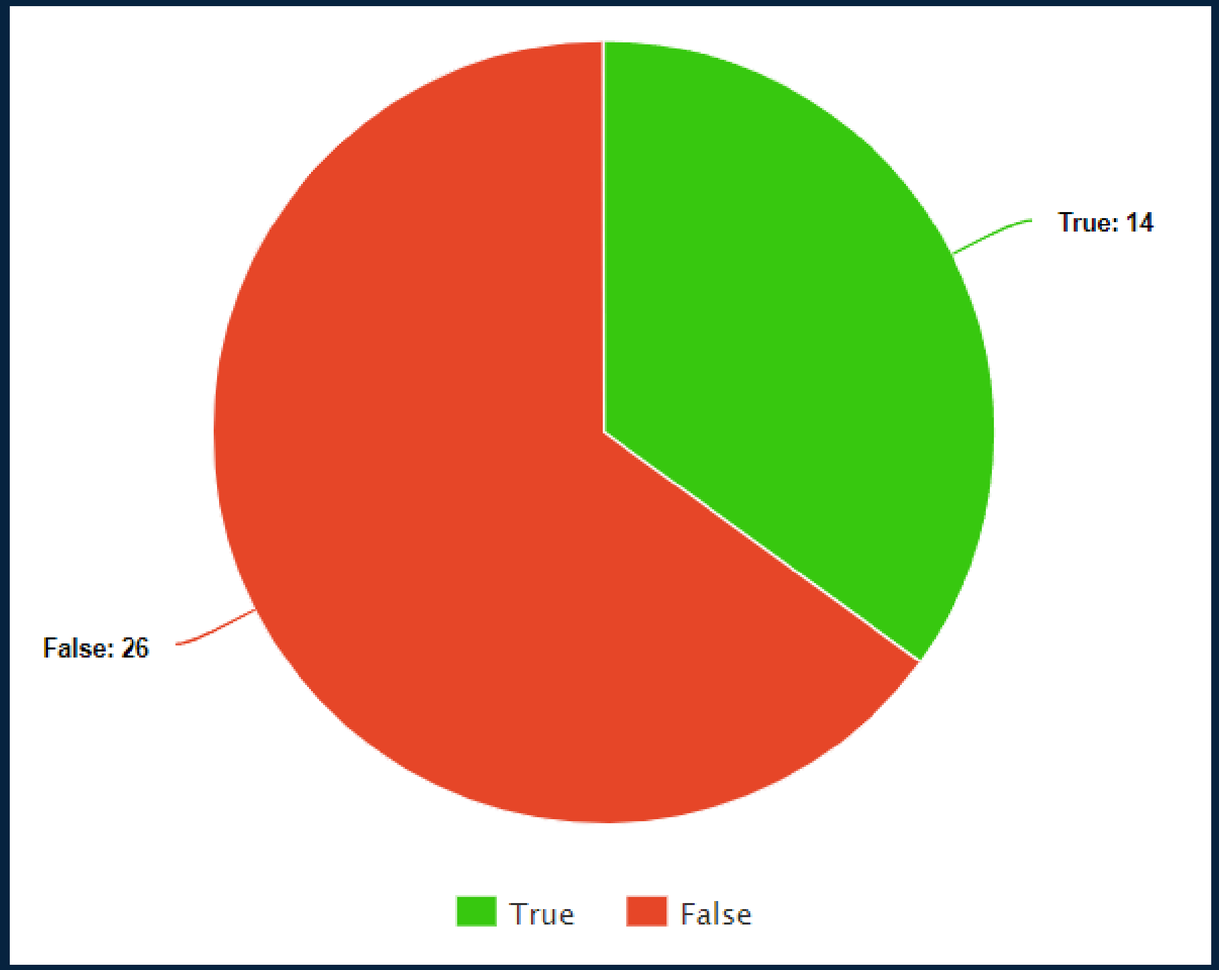
- Using llama3 8B
- Ask llama if the translated sentence is semantically the same as the paraphrased one
- Comparison: 19 mistakes out of 40
  - Mistakes include False Positives and False Negatives



Accuracy: 0.525
Precision: 0.875
Recall: 0.4516
F1 Score: 0.5957

Llama comparison

- Actually true results are 14 out of 40



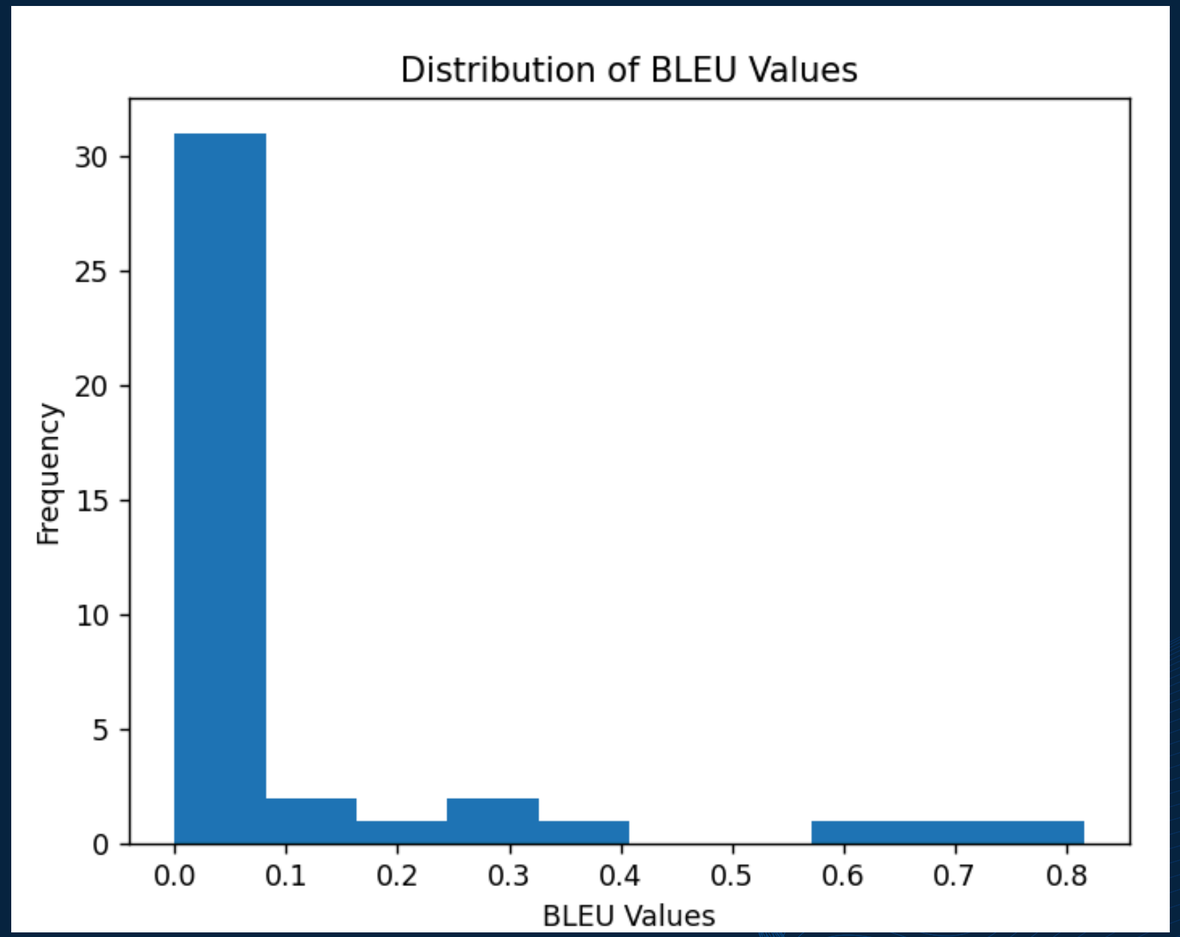
BLEU

- Got base code from library: https://huggingface.co/spaces/evaluate-metric/bleu

```python
def eval_bleu_logic(dataset1: str, dataset2: str, model_response: str):
    dataset = [dataset1, dataset2]
    reference = [question.split(" ") for question in dataset]
    predictions = model_response.split(" ")
    score = sentence_bleu(reference, predictions)
    return score
```

BLEU Statistics

- Min: 0.0
- Max: 0.8153551038173115
- Average: 0.08747699091088278



The results of BLEU do not look promising

BLEU

- Problems occur when sentences are vastly different but still have the same semantic meaning
    - Take a look at the following Examples:

Dataset:	Translation:	
How is the naval artillery with the smallest firing range called?	What are the effective firing range of naval artillery pieces, listed in order from shortest to longest, give the top answer?	→ Effectively the same, but give a low BLEU score

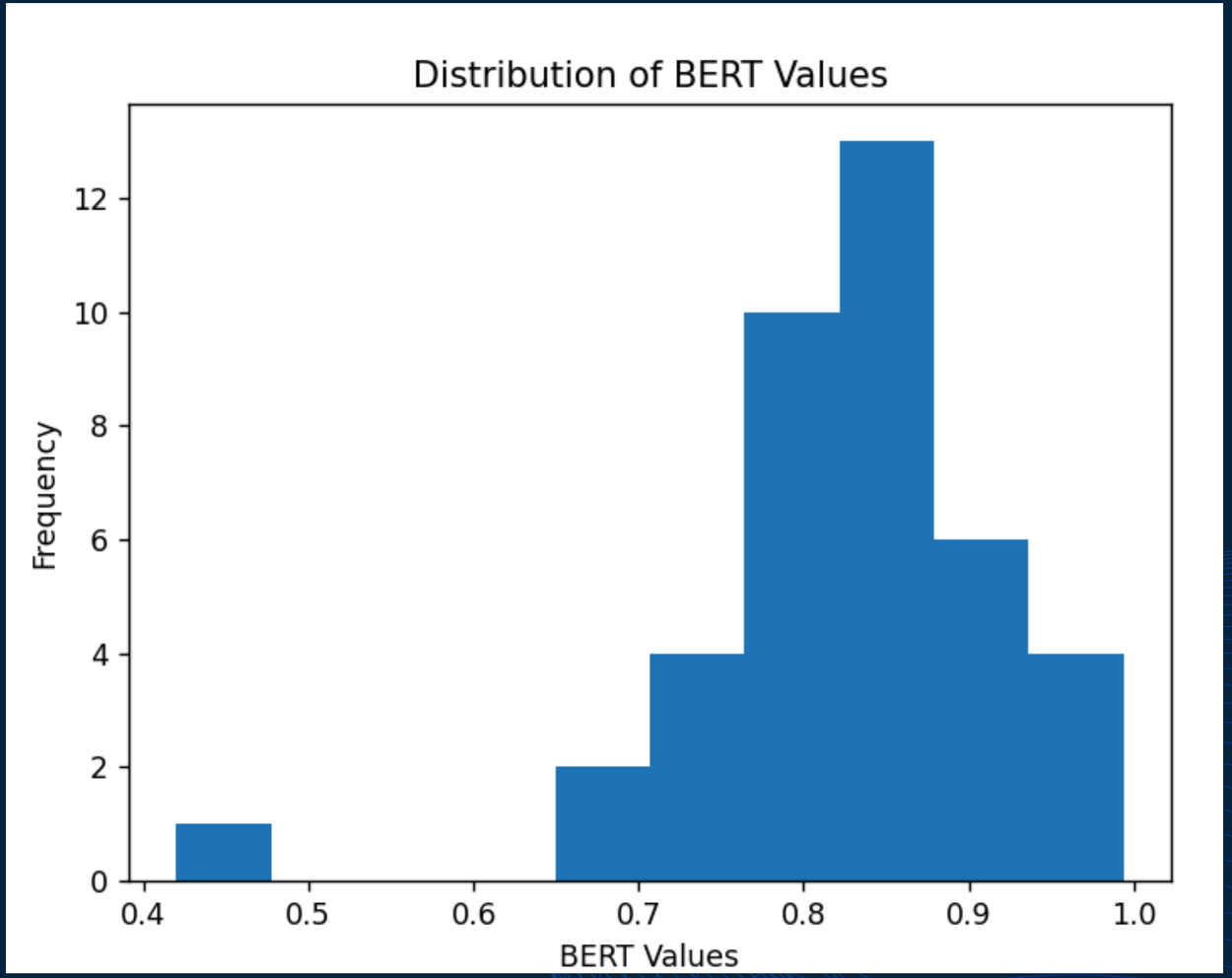
BERT

- Got base code from https://huggingface.co/docs/transformers/en/model_doc/bert
- Converts text to vector
- Semantically indifferent text will give vectors that are close to each other

```python
def eval_bert_logic(dataset: str, model_response: str):
    tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
    model = BertModel.from_pretrained("bert-base-uncased")

    inputs1 = tokenizer(dataset, return_tensors="pt", padding=True, truncation=True)
    inputs2 = tokenizer(model_response, return_tensors="pt", padding=True, truncation=True)

    outputs1 = model(**inputs1)
    outputs2 = model(**inputs2)

    embeddings1 = outputs1.last_hidden_state.mean(dim=1).detach().numpy()
    embeddings2 = outputs2.last_hidden_state.mean(dim=1).detach().numpy()

    similarity = np.dot(embeddings1, embeddings2.T) / (np.linalg.norm(embeddings1) * np.linalg.norm(embeddings2))

    return similarity[0][0]
```

BERT Statistics

- Min: 0.41958645
- Max: 0.9939645
- Average: 0.8257805704999999
- Median: 0.8335725700000001
- Mode: 0.41958645



Overall, the results of BERT look promising

Current Issues and Next Steps

LLM Compaison

- LLama comparison is not reliable
- Adding more examples to the prompt does not help considerably

→

- Try other models
  - llama 70B size,
  - llama Optimized for dialogue

BLEU

- Bleu is unreliable due to the many different possible wordings of the same question

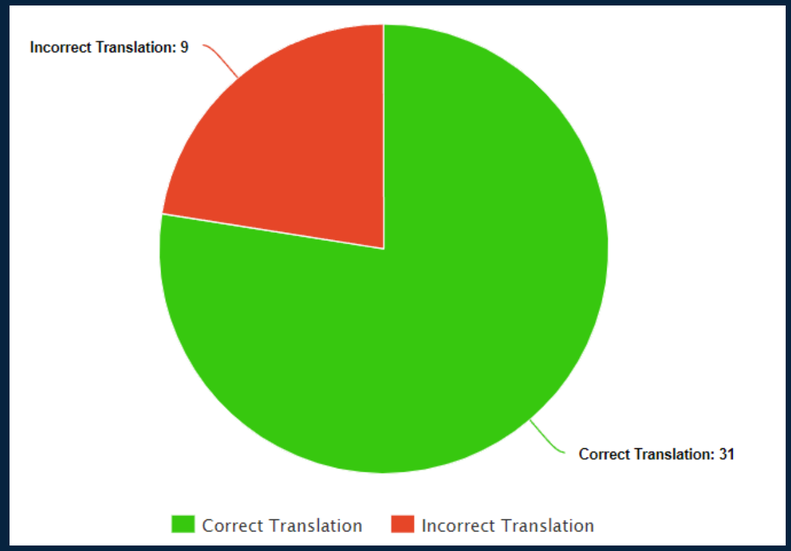
→

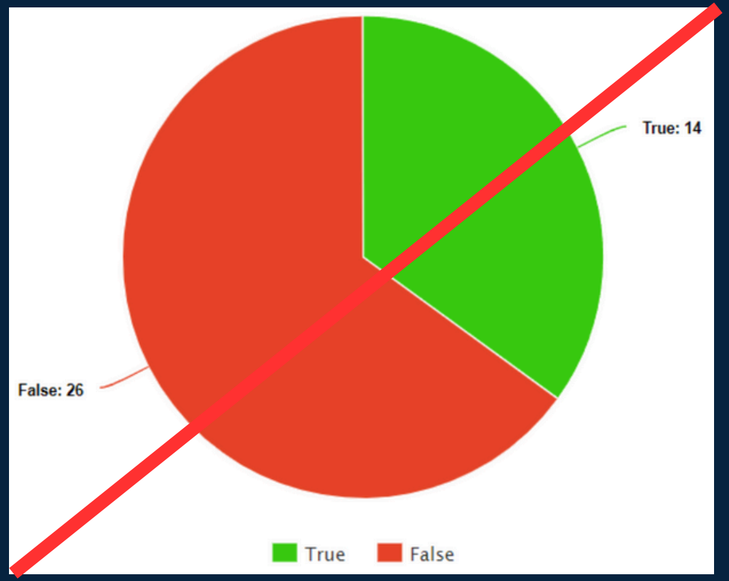
- Try adding weights
  - e.g: weights=[1]

Current results summary 1

Translation with Manual inspection:

- ~75% correct
- LLama seems to be performing the translation well but there is room for improvement





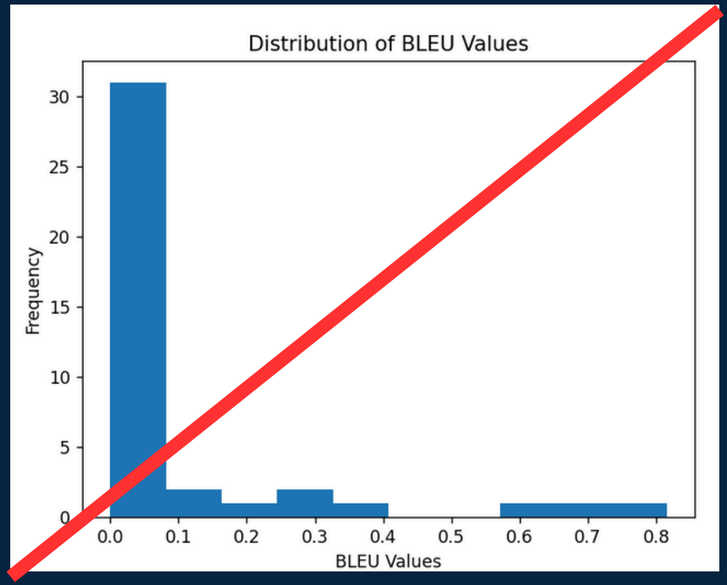
Translation with LLama comparison:

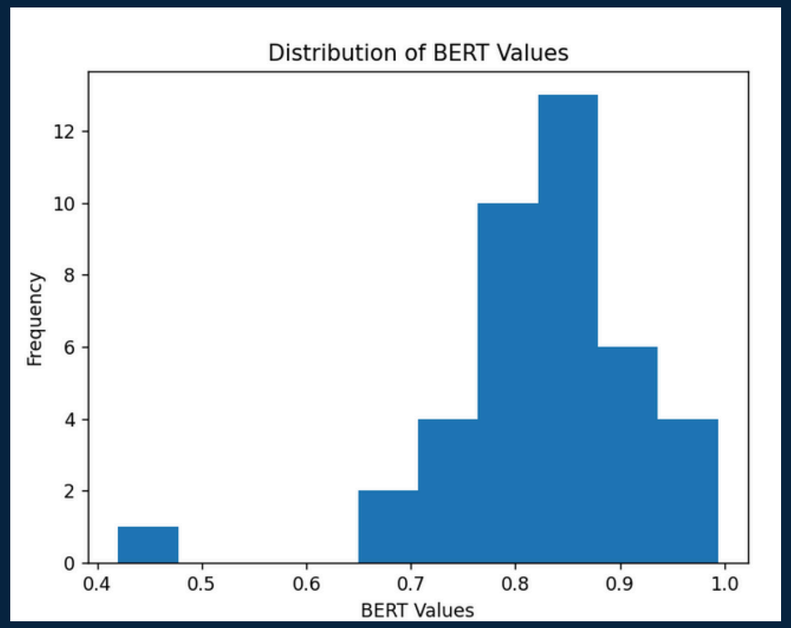
- Only ~35% evaluated as correct
- LLama does is not performing the comparison well
- Experiments with other models should be performed

Current results summary 2

BLEU

- Values show more negative translation results
- Inconsistent with translation and manual comparison results
- Seems to be unreliable in comparison to manual inspection and BERT





BERT

- Values show more positive translation results
- Consistent with translation and manual comparison results
  - More reliable

Questions?