Submission Report

Team 25:

- Advaith Malladi (2021114005)
- Darur Lakshmipathi Balaji (2021114007)
- Raghav Donakanti (2021101024)

Project:

Factual Verification Over Multimodal Evidence Sources

What has been completed:

- **Statistical Entity Extraction** (extract_ntts_statistical.py)
- **Neural Entity Extraction** using Named Entity Recognition (extract_ntt_neural.py)
- **Verification of Entities** and their presence in WikiData Dump (verify_ntts.py)
- **Ranking of Entities** based on the semantic similarity between the introductory articles in the Wiki page of the entity and the Claim, **choose top k entities** (rank_ntts.py)
- **Image Extraction of all entities** which was **not present in the provided dataset and had to be implemented from scratch** (extract_images.py)
- **Ranking of all paragraphs and tables of all k entities** against the claim, selecting top i paragraphs and tables (bert_rank_paras.py)
- Architecture for **FactBERT** for factual verification, we were able to achieve an **Accuracy of 0.7004986595174263 with Cross Entropy loss: 0.9623966813087463** (bert_baseline.py)
- Baseline Implementation
- Scraping of all the images from wikipedia based on extracted entities

- Ranking of all the images using **CLIP Model** to select the top 2 relevant images as evidence
- Ranking of paragraphs and tables using TF-IDF has also been done as ranking evidence based on BERT embeddings is not scalable due to time and memory constraints.
- Implementation of model: **BERT+DINO** for multimodal feature extraction and fusing evidence by a model built and trained by us and factual verification
- Fine-tuning **BridgeTower** model presented in a paper in AAAI 2023 for multimodal feature extraction by adding a classification head on top. Reference: ***Xu, X., Wu, C., Rosenman, S., Lal, V., Che, W., & Duan, N. (2023). BridgeTower: Building Bridges between Encoders in Vision-Language Representation Learning. Proceedings of the AAAI Conference on Artificial Intelligence, 37(9), 10637-10647. https://doi.org/10.1609/aaai.v37i9.26263***

Dataset Description:

- Dataset consists of 71000 claims which are either verified, or refuted or not supported by all of wikipedia articles as of 2020
- So, each sample has 3 classes:
  - 0 Supported
  - 1 Refuted
  - 2 Not enough Evidence
- Dataset also consists of a 60GB dump of all wikipedia pages which consists of tables and paragraphs, we had to scrape all the images by ourselves

Methodology?

Statistical Entity Extraction:

- Given a claim, we extracted all ngrams with n ranging from 1 to 13

- We queried the wiki articles database to check for the presence of all of these ngrams and their substrings
- We removed all the stop words and stop phrases
- We finally stored all the entities obtained this way
- **Observation:** This method of extracting entities resulted in noisy data as a lot of entities would be extracted and ranking all the introductory paragraphs across all these articles resulted in selecting the wrong entities which resulted in an **accuracy of 0.37** Even randomly selecting the class could do better

Neural Entity Extracting using Named Entity Recognition:

- Given a claim, we implemented NER to selected all the named entities.
- Using this directly resulted in information loss, for ex:
  - If the required entity was **‘2003-04 Indian Cricket Team’,** the entity extracted would be **‘Indian Cricket Team’** clearly a sign of information loss, we would not be able to retrieve the correct evidence
- So, we had to augment the entities which we got through neural NER.
- **Augmentation**:
  - say the entity we got through NER was <N1>, let the word predecessor of <N1> in the claim be <W1> and the word successor of <N1> be <W2>
  - We considered the following entities:
    - <N1>
    - <W1> + <N1>
    - <N1> + <W2>
    - <W1> + <N1> + <W2>
- Thus extracting entities by using neural NER and the above mentioned Augmentation resulted in the best performance

Verification of Entities:

- After extracting entities from the claim, we had to ensure that Wiki Articles existed for these entities.
- We did this by querying the wiki article database
- We only chose entities which had wiki articles. This brought the entity count down from n to m

Ranking Entities:

- Now that we have a claim and related m entities, we extracted introductory paragraphs from the wikipedia articles for all of these m entities.
- Thus, we have 1 claim and m introductory paragraphs to rank.
- We used BERT base uncased to encode all of these m paragraphs and 1 claim and measured cosine similarity
- We chose the top k entities from these m entities based on the cosine similarity

Image Extraction

- We wrote to return a list of all the image urls for any wikipedia entity
- We had to augment the dataset using this

Ranking Images:

- We made use of a pre-trained **CLIP model** to select the top 2 images relevant to the current claim.
- This model works by calculating and trying to minimize the loss between the representations of the image and the claim, thus ranking the images by making use of the losss.

Ranking Paragraphs and Tables using BERT:

- For the top k entities we had a list of all the paragraphs and tables represented in a linear fashion.
- We ranked all the tables and paragraphs against the claim using BERT based semantic similarity
- We chose the top i paragraphs and tables

Ranking Paragraphs and Tables using TF-IDF:

- As ranking paragraphs and tables using a BERT encoder is not scalable due to time and VRAM requirements, we made use of TF-IDF to rank the paragraphs much faster as these claims were directly extracted from the wiki articles
- We notices there was not much difference in these methods of ranking evidence as similar scores were achieved by the baseline model in both cases.

Factual Verification and FactBERT:

- Here, we extended the architecture of BERT and trained it further for the final task. We propose FactBERT:

```
fact_model(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (linear): Linear(in_features=1536, out_features=3, bias=True)
)
```

- Here we could do things:
  - Train the entire model (accuracy: 0.46153846153846156)
  - Train the final layer (accuracy: 0.7004986595174263)
- Clearly, training only the final layer resulted in better performance
- Here, we have a list of evidences and one claim
- All of the evidences are first passed through BERT.
- The representations of the encoded evidences have been added
- Then we pass the claim through BERT
- The added evidences and claim are concatenated and then passed through a linear layer for final classification.

Factual Verification and Fact(BERT+DINO):

- Here we fine tuned a fusion of two models, DINO and BERT. DINO for image representations, BERT for textual representations
- Here we tried to learn fused image+text representations by passing the image and text outputs through non linear layers and trying to align the spaces learnt by these non linear layers.
- Then, we added a classification layer on top to make the final judgement
- Here, we tried to make our model learn the fused representations by trying make the model align the spaces for these tasks

Factual Verification and BridgeTower model:

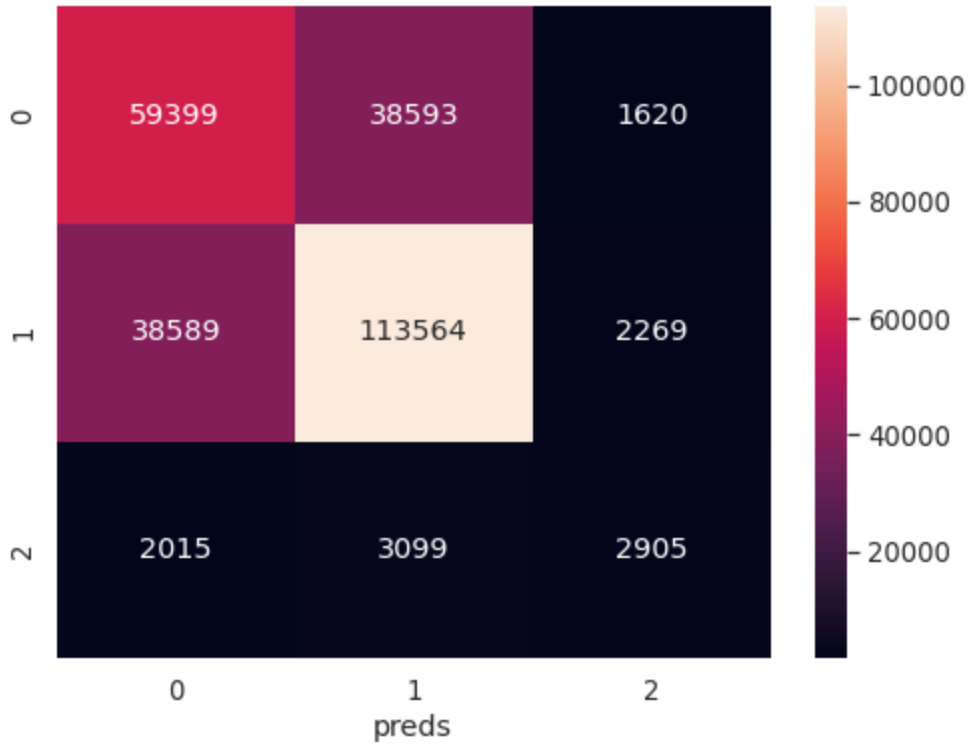
- Bridge Tower is a pre-trained model presented in AAAI’23 which can be used to acquire fused image and text representations
- We fine tuned Bridge Tower model to suit our use case here.

- We passed the evidence through the model and through classification head attached to the model to make the final judgement.

Results:

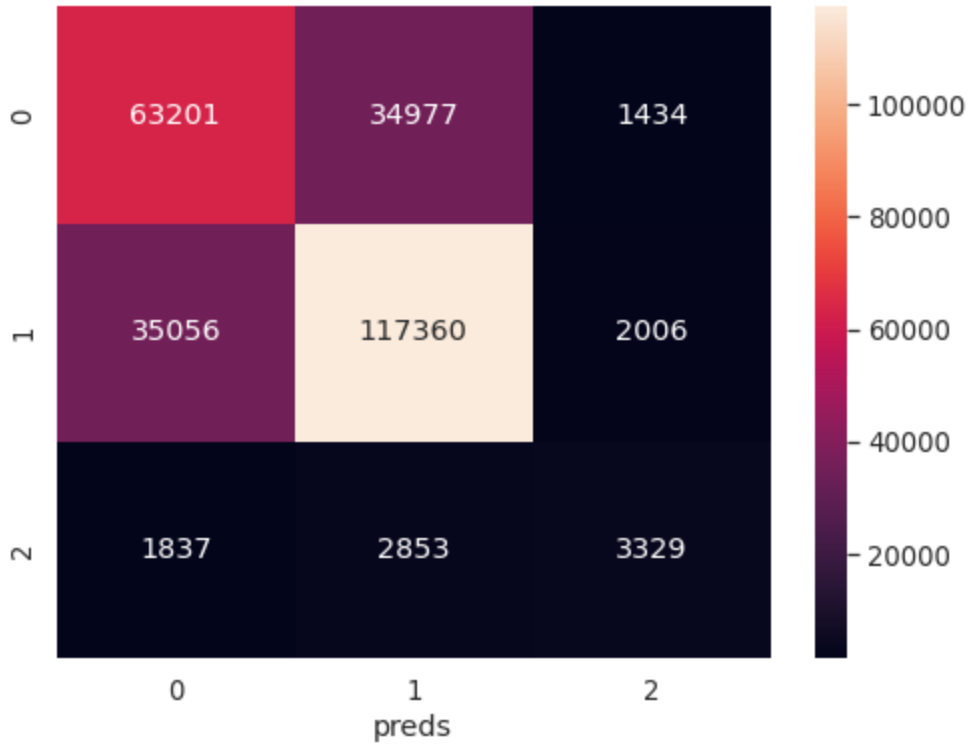
Using Vanilla BERT:

- accuracy: 0.6711161482600848 F1 Score: 0.5735971583009903



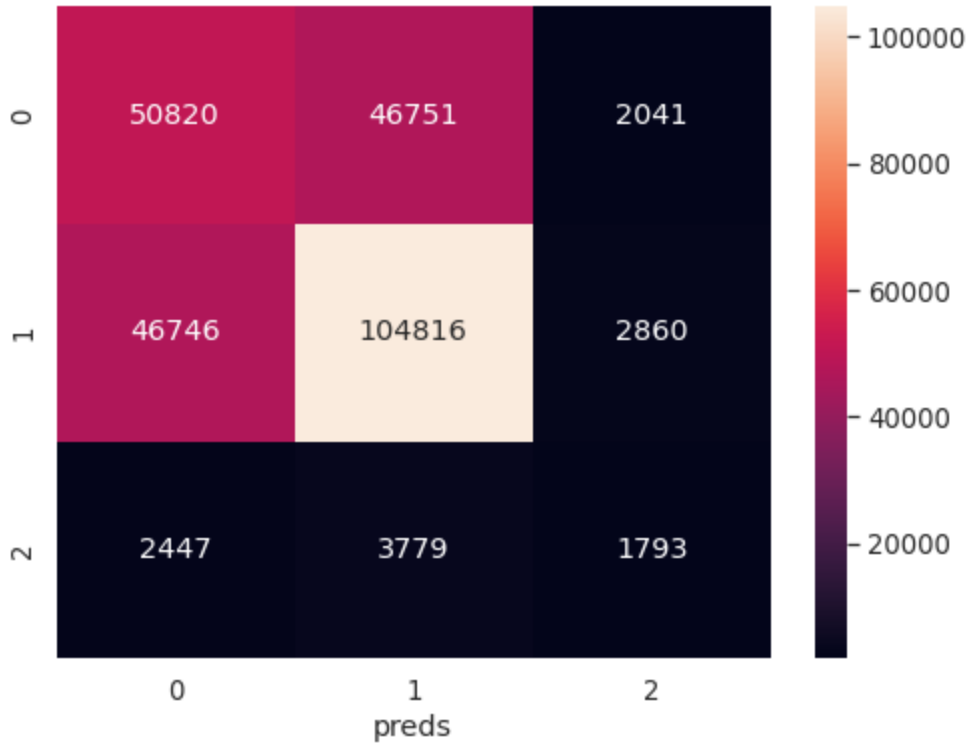
Using BERT+DINO:

- accuracy: 0.7017282763410455 F1-Score: 0.6137601631128456



Using Bridge Tower:

- accuracy: 0.6007525195284923 F1 score: 0.47654114008366194



Link to Presentation: Presentation21.pptx

Link to Onedrive: ANLP_Proj