Emotion Detection – Artificial Neural Networks Performance Comparison

Natural Language Processing – Final Project
HIT – 99101 - Dr. Nava Shaked

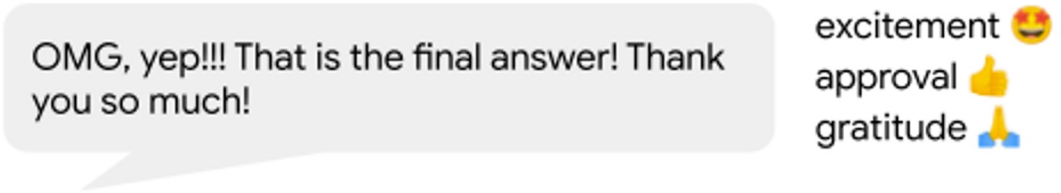
Tali Aharon 034791236
Helit Bauberg 027466002

Emotions Text Classification – Overview

- Emotions are a key aspect of social interactions, influencing the way people behave and shaping relationships.
- Especially with language — with only a few words, we're able to express a variety of subtle and complex emotions.
- Two critical areas of natural language processing are sentiment analysis and emotion recognition:
  - Sentiment analysis is a means of assessing if data is positive, negative, or neutral.
  - In contrast, Emotion detection is a means of identifying distinct human emotion types such as furious, cheerful, or depressed.

Emotions Text Classification - Challenges

- The need for an **emotion dictionary** to contain a reasonable number of emotion categories, since limited keywords can greatly affect the performance of the approach among ambiguity of keywords and the lack of linguistic information.
- **Irony and sarcasm**: In sarcastic text, people express their negative emotions using positive words. This fact allows sarcasm to easily cheat emotion classification models unless they're specifically designed to take its possibility into account.
- **Types of negations**: The simplest approach for dealing with negation in a sentence, which is used in most state-of-the-art emotion analysis techniques, is marking as negated all the words from a negation cue to the next punctuation token.
- **Word ambiguity**: The problem of word ambiguity is the impossibility to define polarity in advance because the polarity for some words is strongly dependent on the sentence context.
- The **expression of multiple emotions** in a single sentence. It is difficult to determine various aspects and their corresponding sentiments or emotions from the multi-opinionated sentence.



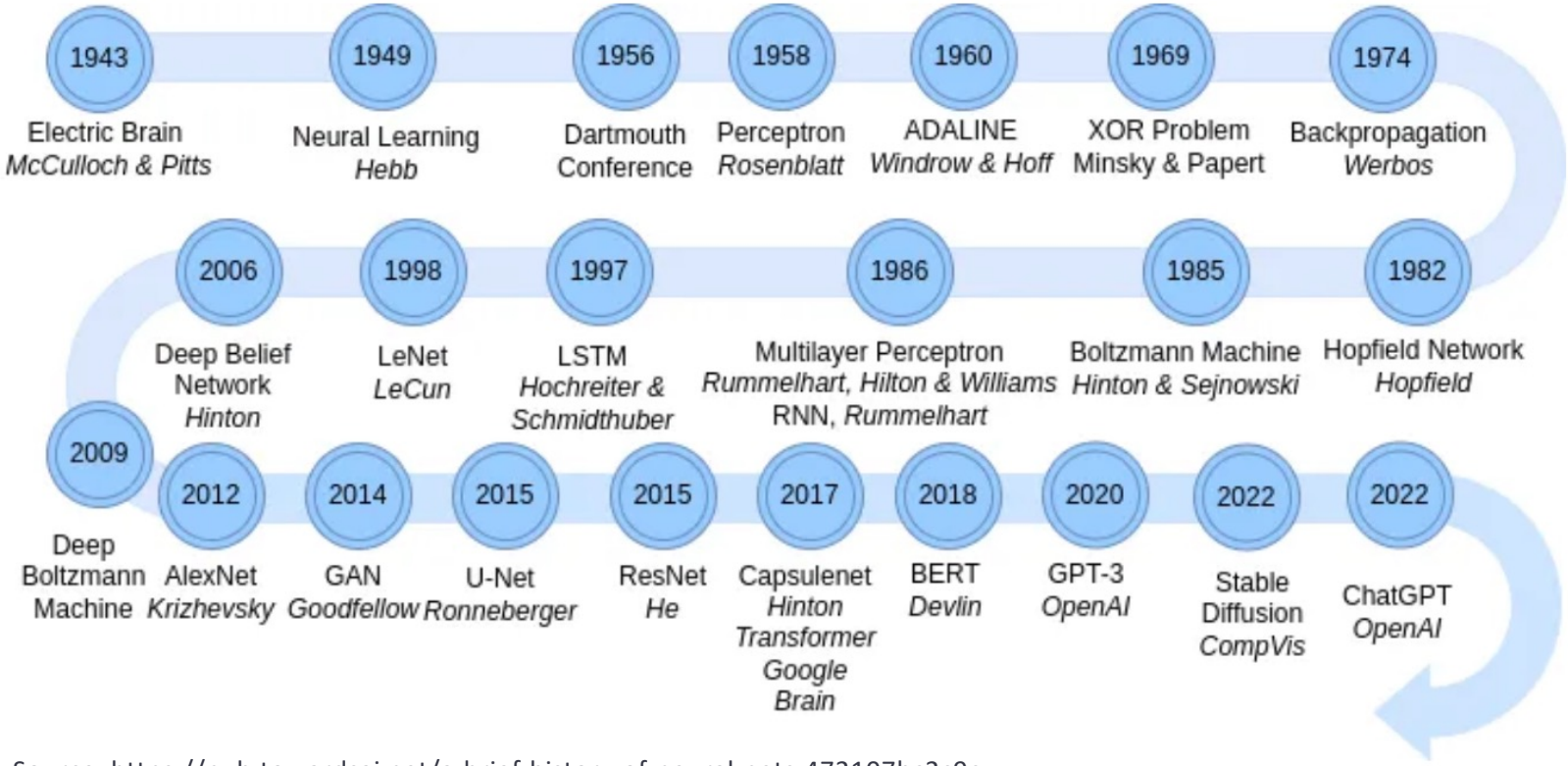
Initial hypothesis: The Bilateral Brain

- In our initial hypothesis, we wanted to explore a new hybrid approach, a brain inspired network with bilateral functionalities.
- At any given moment, our world view is a fusion of two very different world views, one produced by the right hemisphere, the other by the left.

- A recent research (https://arxiv.org/abs/2209.06862) tested the bilateral approach on **image classification** by using two different and specialized Artificial Neural Networks to achieve a better image classification.
- Our hypothesis was that we can custom two ANNs to do the same for **textual emotion classification**.

Left Hemisphere Perspective	Right Hemisphere Perspective
Narrow, detail focused attention	Wide beam attention
Simplified, explicit version of reality	Understands complexity
Black and white categorization	Understands context, implicit meaning
Fixed, isolated, static	Embodied, interconnected, dynamic
Disposition towards the mechanical	Disposition towards the living
Knowledge of the parts	Wisdom about the whole and world knowledge
Language specialization*	Indirect communication (body, intents..)*

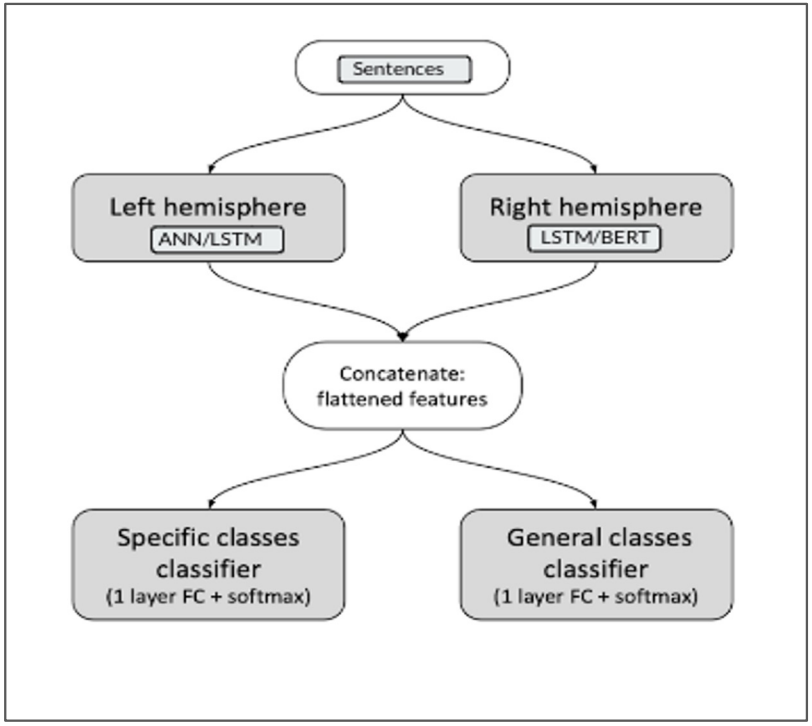
The evolution of Artificial Neural Networks:



Source: https://pub.towardsai.net/a-brief-history-of-neural-nets-472107bc2c9c

Initial hypothesis

- We started by exploring (online and by scripts) some potential candidate ANNs on the **IMDB movie reviews – sentiment classification dataset.**
- IMDB dataset contains a training set of 25,000 Reviews labelled with Pos/ Neg sentiments.
- For the **left hemisphere** candidates, we got the following accuracies -
  - Simple ANN - 86.57%
- Network with **both left and right** properties -
  - LSTM - 82.27%
- For the **right hemisphere** -
  - BERT - 94.2



⇒ This surprised us, as LSTM should have scored better than a simple ANN.
We understood that it had to do with the size of the data and looked for a larger training dataset.
⇒ We also understood that we need an hierarchical dataset (which we may classify to multiple or to binary classes, as in the original model).

GoEmotions Dataset

- A corpus of 58k carefully curated comments extracted from Reddit, with human annotations to **27 emotion categories + Neutral** (The Neutral category makes up 26% of all emotion labels)
- Most of the examples (83%) have a single emotion label and have at least two raters agreeing on a single label (94%).

Positive		Negative		Ambiguous
admiration 👏	joy 😃	anger 😡	grief 😢	confusion 😕
amusement 😂	love ❤️	annoyance 😒	nervousness 😬	curiosity 🤔
approval 👍	optimism 🤞	disappointment	remorse 😔	realization 💡
caring 🤗	pride 😌	disapproval 👎	sadness 😞	surprise 😲
desire 😍	relief 😅	disgust 🤮		
excitement 🤩		embarrassment 😳		
gratitude 🙏		fear 😨		

GoEmotions taxonomy: Includes 27 emotion categories, plus “neutral”.

Classifying emotions using GoEmotions Dataset

ANNs performance on GoEmotions:

- **LSTM** - 62.29% (on Pos/Neg/Neutral)
- **Bilateral LSTM** - 66.19% (on Pos/Neg/Neutral)
- **Pretrained BERT** - **96.21% (28 classes)**

⇒ Bidirectional LSTM showed slightly better results than LSTM.
⇒ BERT won by a "knockout" !!
⇒ There is no benefit in exploring additional ANNs or combining BERT with another ANN.

Revised hypothesis: Improving right brain capabilities

As Best classification was achieved using BERT, we revised our hypothesis.
We decided to re-scope our project by trying to simulate **right** hemisphere capabilities using different types of layers on top of BERT

➔ Our new hypothesis:

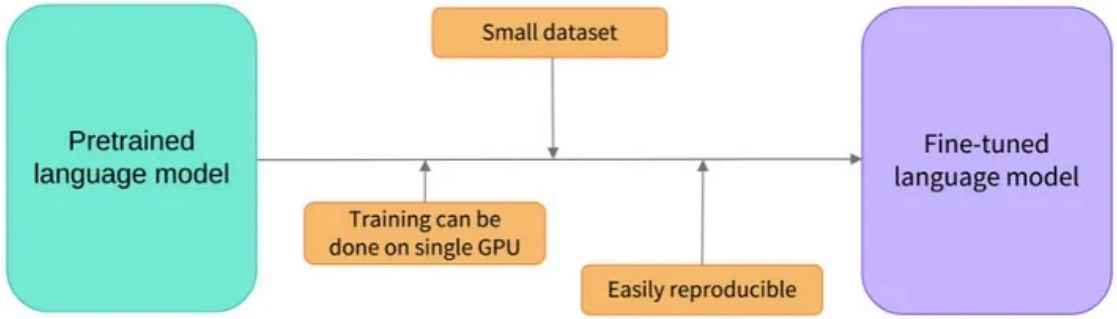
Fine-tuning Pre-trained BERT with additional layers would yield the best accuracy on textual emotion classification.

- We used GoEmotions to check our hypothesis.

Right Hemisphere Characteristics – mapping to ANN type/configuration:

Right Hemisphere characteristics	Mapping to ANN characteristics
Wide beam attention	BERT – multiple attention heads, long input size; LSTM, bi-LSTM - capable of learning long-term dependencies.
Understands complexity	Pre-trained, large corpus, multiple epochs
Understands context, **implicit meaning**	BERT – multiple attention heads, long input size; LSTM, bi-LSTM - LSTM can derive context from previous words in a sentence. Here bi-LSTM is better since it has context understanding from the following words as well. **multiclass emotion classification**
Wisdom about the whole and world knowledge	Pre-trained, large corpus
Indirect communication (body, intents)	**multiclass emotion classification**

Classifying emotions – Transfer Learning



Pre-trained models are models that have already been trained on large amounts of data) for some other task. Training pre-trained model on a new task, related to the original task the model was trained for, significantly reduces required training resources and has proven to produce good accuracy.

GoEmotions Exploratory Data Analysis

Jupiter notebook – BERT baseline model: download and view the dataset:

```
[ ] from datasets import load_dataset
    emotions = load_dataset("go_emotions", "raw")

    WARNING:datasets.builder:Found cached dataset go_emotions (/root/.cache/huggingface/datasets/go_emoti
    100% 1/1 [00:00<00:00, 20.26it/s]

[ ] emotions

    DatasetDict({
        train: Dataset({
            features: ['text', 'id', 'author', 'subreddit', 'link_id', 'parent_id', 'created_utc', 'rater
    'caring', 'confusion', 'curiosity', 'desire', 'disappointment', 'disapproval', 'disgust', 'embarrassm
    'pride', 'realization', 'relief', 'remorse', 'sadness', 'surprise', 'neutral'],
            num_rows: 211225
        })
    })

    df = emotions['train'].to_pandas()
    df.describe

[ ] label_cols = ['admiration', 'amusement', 'anger', 'annoyance', 'approval', 'caring', 'confusion',
                  'curiosity', 'desire', 'disappointment', 'disapproval', 'disgust', 'embarrassment',
                  'excitement', 'fear', 'gratitude', 'grief', 'joy', 'love', 'nervousness', 'optimism',
                  'pride', 'realization', 'relief', 'remorse', 'sadness', 'surprise', 'neutral']
    len(label_cols)

    28
```

Tokenization and Preprocessing

Tokenize and encode

```
[ ]  model_ckpt = "distilbert-base-uncased"
     tokenizer = AutoTokenizer.from_pretrained(model_ckpt)
```

```
Downloading (…)okenizer_config.json: 100% 28.0/28.0 [00:00<00:00, 1.30kB/s]
Downloading (…)lve/main/config.json: 100% 483/483 [00:00<00:00, 25.0kB/s]
Downloading (…)solve/main/vocab.txt: 100% 232k/232k [00:00<00:00, 1.42MB/s]
Downloading (…)/main/tokenizer.json: 100% 466k/466k [00:00<00:00, 1.90MB/s]
```

```
[ ]  train_encodings = tokenizer(df_train["text"].values.tolist(), truncation=True)
     test_encodings = tokenizer(df_test["text"].values.tolist(), truncation=True)
```

```
[ ]  train_labels = df_train["labels"].values.tolist()
     test_labels = df_test["labels"].values.tolist()
```

```
class GoEmotionDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):
        item = {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}
        item['labels'] = torch.tensor(self.labels[idx])
        return item

    def __len__(self):
        return len(self.labels)
```

GoEmotion Dataset Loading

```
train_dataset = GoEmotionDataset(train_encodings, train_labels)
test_dataset = GoEmotionDataset(test_encodings, test_labels)
```

```
[ ] train_dataset[0]
```

```
{'input_ids': tensor([ 101, 2008, 2208, 3480, 1012,  102]),
 'attention_mask': tensor([1, 1, 1, 1, 1, 1]),
 'labels': tensor([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 1, 0, 0])}
```

```
[ ] # sanity check
tokenizer.decode(train_dataset[0]["input_ids"])
```

```
'[CLS] that game hurt. [SEP]'
```

Classifying emotions - pretrained distilBERT

Creating a Custom Model

```
class DistilBertForMultilabelSequenceClassification(DistilBertForSequenceClassification):
    def __init__(self, config):
      super().__init__(config)

    def forward(self,
        input_ids=None,
        attention_mask=None,
        head_mask=None,
        inputs_embeds=None,
        labels=None,
        output_attentions=None,
        output_hidden_states=None,
        return_dict=None):
        return_dict = return_dict if return_dict is not None else self.config.use_return_dict

        outputs = self.distilbert(input_ids,
            attention_mask=attention_mask,
            head_mask=head_mask,
            inputs_embeds=inputs_embeds,
            output_attentions=output_attentions,
            output_hidden_states=output_hidden_states,
            return_dict=return_dict)

        hidden_state = outputs[0]

        return SequenceClassifierOutput(loss=loss,
            logits=logits,
            hidden_states=outputs.hidden_states,
            attentions=outputs.attentions)
```

```
[ ] num_labels=28
    model = DistilBertForMultilabelSequenceClassification.from_pretrained(model_ckpt, num_labels=num_labels).to('cuda')
```

```
Downloading model.safetensors: 100% 268M/268M [00:03<00:00, 82.7MB/s]
Some weights of the model checkpoint at distilbert-base-uncased were not used when initializing DistilBertForMultilabelSequenceClassification:
- This IS expected if you are initializing DistilBertForMultilabelSequenceClassification from the checkpoint of a model trained on another task
- This IS NOT expected if you are initializing DistilBertForMultilabelSequenceClassification from the checkpoint of a model that you expect to
Some weights of DistilBertForMultilabelSequenceClassification were not initialized from the model checkpoint at distilbert-base-uncased and are
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

model	additional layers/activation layers	epochs	data size	Accuracy Pre training	Accuracy Post training
distilbert-base-uncased	None	3	1%	50.05%	96.19%
distilbert-base-uncased	None	3	100%	51.05%	96.16%
*distilbert-base-uncased	1 x bidirectional LSTM layer	12	1%	57.66%	95.94%
*distilbert-base-uncased	1 x bidirectional LSTM layer	12	100%	47.1%	95.93%
distilbert-base-uncased	1 x dense (fully connected) layer 1 x RELU	3	1%	49.37%	96.20%
distilbert-base-uncased	1 x dense (fully connected) layer 1 x Sigmoid	3	1%	40.85%	96.20%
distilbert-base-uncased	**1 x dense (fully connected) layer 1 x tanh**	**3**	**1%**	**60.12%**	**96.22%**
** distilbert-base-uncased	1 x dense (fully connected) layer 1 x tanh	10	100%	52.79%	96.20%
distilbert-base-uncased	1 x dense (fully connected) layer 1 x tanh	3	100%	56.98%	96.19%

Results:

Our models vs. GoEmotions official Benchmark model

Benchmark - https://ai.googleblog.com/2021/10/goemotions-dataset-for-fine-grained.html
Best benchmark model:
Pretrained BERT + dense layer + sigmoid activation

Our best model:
Pretrained DistilBERT + dense layer + tanh activation

⇒ We were able to improve the benchmark model by 0.02%

Overall Conclusions

- In our process, we followed the evolution of Artificial Networks for NLP tasks.
- We have seen that there's no benefit in combining two ANNs (simulating two hemispheres) together into one 'bilateral' model.
- The theoretical advantages of the newer bi-directional transformer architectures over the sequential processing of the LSTM network architectures were empirically established through a set of comparative experiments.
- biLSTM showed somewhat better results compared to LSTM model. This is due to its ability to derive bi-directional context (from sentence beginning to end, and from sentence end to beginning).
- Data size – we saw that LSTM and biLSTM require massive amount of training data to provide accurate classification. BERT, on the other hand, achieved superior results with far less training data. This is probably due to both factors – multi attention architecture as well as the use of pre-trained models for our experiments.
- There is a big variety of open source pre-trained models for various NLP tasks freely available online. One of our key observations is how significant *Transfer Learning* is to the process: whether it is in the short training time, or the small amount of data that is required to train a pre-trained model.
- We have managed to produce a model with slightly better classification accuracy: distilBERT + dense layer + tanh activation layer showed 96.22% compared to our benchmark model (BERT + dense layer + sigmoid) that produced 96.20%

Future Work

- Alternatives to Human-Labeling of datasets for ML.
- Automating Social Media emoji tags (See https://www.researchgate.net/publication/321057905_Emoji_as_Emotion_Tags_for_Tweets)
- Application to identify and recognize emotions for children with autism (See https://www.frontiersin.org/articles/10.3389/frcha.2023.1118665/full)

Thank you !! 😊