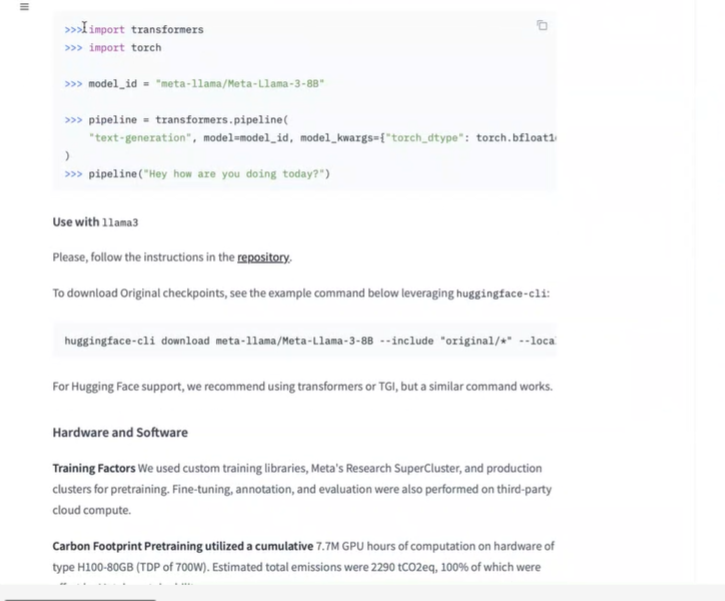
**Midterm Webinars**

* About topics 3 and 4, language modelling.
* **Main NLP task:** this is a *text classification* task. The midterm is all about THIS task. You need to select an **annotated dataset with categories and classes**, e.g. spam, fake news, humour, irony, sentiment/emotion analysis.
* **Main Things to Takeaway from the Course:** main NLP trends are black-box, but we need the results to be **explainable and interpretable**, which is the *statistical, traditional way*! Generative AI = not interpretable! Traditional models have an interpretable aspect to the results. Should be able to *figure out which is better* 🡺 manual feature extraction or AI-black box way? Is there a middle ground we can create/?
* The main marks are for:
  + Text processing
  + Features
  + Embedding
  + Model training is just a small component!
* E.g. *do not choose an SVM if there is a clear class imbalance!* This would be incorrect.
* Selecting the right text, cleaning, tokenizing, normalizing, removing/stopwords, taking the right features, then training the model 🡺 then the evaluation + feedback, correct justification for metrics is what is important.
* Once you take the dataset (e.g. tweets) with annotations for categories (supervised learning).
* In the NLP pipeline, you need to do 2 things:
  + Compare **traditional statistical methods (manual feature-extraction method), e.g. n-grams** to deep learning/embeddings.
  + With automated feature-extraction methods (embeddings), traditional statistical classifier. Justify which classifier (e.g. Naïve-Bayes).
  + Like comparing machine learning to deep learning.
  + But more interested in extracting **textual features** and **feature extraction** (manually vs DL/automated using embeddings) than traditional ML, must be NLP focused.
* Main takeaway: *justify all the selections!* Justify why TF-IDF, look at the dataset, which features can be used, etc. Decide “I think a more static embedding would work better…”, discussion of computational resources etc.
* **Baseline vs SOTA:** 
  + SOTA = best performing model we have
  + Baseline = a seen, validated result that is the **benchmark for the moment**. A baseline for a new take, e.g. LIAR approach for TF-IDF, trying to implement the same.
  + A baseline is related to the problem statement/same dataset, e.g. “the baseline for fake news, for sentiment analysis etc.”
  + A ***baseline on the same dataset for the same domain*.** Some domains don’t have the same dataset being used, ideally the same DOMAIN AND DATASET. The baseline needs to be something on the same dataset that has been done and published.
  + We do not need SOTA/99% state of the art! Baseline is the basic e.g. 75% accuracy.
* Bare minimum: binary text classification text, e.g. fake news detection, sentiment analysis, spam or not, rumor/not rumor detection, and **document tagging** (is the document related to a domain or not?) 🡪 *trying to categorize the text into 2 domains.*
* Kaggle, HuggingFace etc. 🡺 we are free to choose these but we need to cite the source!
* **Text Normalization:** tokenization, lemmatization etc.
* **A baseline approach is required: a published article/project** is required!
* “Statistical” model vs embedded:
  + Statistical models:
  + SVM
  + RNN is a **deep-learning technique which requires an embedding**, even your own with 1-hot encoding, but can use GloVe or FastText.
  + DL = Deep Learning = requires fine-tuning! Not supposed to **beat** the state-of-the-art or baseline, but we need a **comparable result** to the baseline. Results need to be reproducible.
* **Tips for midterm:**
  + In the rubrics!
  + Basic level: understand what is text classification, show text pre-processing (tokens, techniques and justification), statistical (term frequency, can be very basic), then create a Word2Vec.
  + Superior results: expected you give a **good comparative analysis** and **why things work in favour of embeddings OR in favour of statistical approaches**, based on the particularities of the dataset and why things happen.
  + A consideration of which text you have, and how you apply the techniques on the use-case.
  + **Cross-validation is a good idea if there is a clear imbalance e.g. Fake News**. Must check against *overfitting*.
  + Higher marks: more about how you present/explain your comparison of different results. Statistical analysis VS embeddings are **black box**, so selecting the right *embedding* (and explain the black box and **why** you selected this dataset, why this interested you, *why* the evaluation metrics were taken *for this dataset*), *why not doing something like an SVM or Naïve-Bayes, why which embedding, a systematic build-up on WHY and HOW*.
  + **Which embedding layer are you using for the RNN?**
  + All about *which embedding and why*, the **features** you use from the natural text to process into the ML model.
  + The more you elaborate *why*, the higher marks.
  + Binary vs multi-class is just the first step, it’s how you evolve it that matters.
  + **Use transformers AND EXPLAIN WHY!** If you’re using totally transformers, it will be very difficult to explain the results, as they’re totally black-box, it’s not a *static or dynamic embedding* like GloVe or FastText.
  + **Do not use transformers only (a GPT)**, but you could use a BERT embedding.
  + Can use BERT embedding and train a model on that.
  + The core idea is **to explain how the language is being modelled.**
  + E.g. BERT embeddings and DNN is a good idea for one of the extensions 🡺 use a transformer and explain how they work. This is an “advanced approach”, and should be reasoned correctly.
* Main place students lose marks:
  + When you use *all the metrics* even when they are NOT required for the problem statement! That shows you are **not clear about the problem statement** and what would check the efficiency of the classifier.
  + Confusion Matrices should be there, explain WHY using micro/macro average etc.
  + The comparison cannot be “theoretical” in all 🡺 we knew embeddings were an improvement on statistical approaches, but we *need to prove it practically that as we move on from statistical approaches (TF-IDF) to embeddings, the comparison can be SEEN EVERYWHERE*, this example of empirical results and justifying evaluation metrics is needed for high marks.
* General data pre-processing should not change on the model, but on the problem statement. E.g. *should the caps and punctuation remain? If classifying sarcasm, then YES, you’d want to keep punctuation, emojis not removed, keep caps lock etc.*
* Be **very clear** on the problem statement!
* RNN and Transformers are **automated-feature-learning**, then you need to fine-tune the “transfer learning”.
* Use GloVe, Word2Vec, FastText etc.
* **Baselines:** if you don’t have a baseline or SOTA, you cannot validate the results, data-result validation is *also important*, you need a BASELINE REQUIREMENT**, WHETHER IT’S OFFICIALLY PUBLISHED OR NOT,** THERE SHOULD BE A BASELINE ONTO THAT.
* Language modelling topic is related to embeddings + building **contextual embeddings** 🡺 related to medical/legal/financial domain, where you build the *language/contextual embeddings* for medicine or finance only. Like building a *finance GPT*, e.g. “finGPT” related to language-modelling only for a **specific field**.
* Baselines/SOTA = “state of the art technique”, has to be **the same domain and same dataset.**
* Imbalanced datasets 🡺 whatever metrics you apply should be *appropriate* for the use case.
* Challenges with Jupyter Notebooks: the model is deployed as an API on the server
* Extracting Jupyter code into Python script is essential: should be shared code between the inference API and your code
* Evaluation metrics: look at your use case 🡪 an imbalanced dataset always has a problem with accuracy, so the use case dictates whether to look at precision/recall etc. Is it better to have high recall? You can weight the F1 score in either direction.
* Very powerful classifiers are based on techniques developed in TOPIC 2 (nltk and sentence segmentation, word tokenization, extracting lemmas/stems etc)
* When to remove stopwords and when not to?
* Regex: could for parsing LOGS (e.g. for IoT devices, to check if they’re doing alright or not = look for regexes to filter the log message u want)
* Stopwords 🡪 most apps end up removing stopwords, but consider your use case!
* Count vectorizers/BoW might work better than TF-IDF! TF-IDF highlights which words are important, but might not always be the best.
* **You have to test DIFFERENT pipelines:** sometimes removing stopwords/lemmatization etc. is not better = you have to have multiple **experiments** and create tables to compare.
* **1 notebook for data processing and wrangling**
* Using the result from notebook 1 to put into new notebook for machine learning models.
* Third notebook for extra testing and evaluation?
* **Simple statistics:** collocations, lexical diversity, estimating probabilities, language modelling, topic modelling, **scikit-learn** topic modelling.
* **Topic modelling = clustering (seeing where clusters of text exist)**. Lots of different mechanisms for this! Explore them all!
* **Semantic Similarity**: **there are transformer-based methods for semantic similarity now, so look into that.**
* **Look into transformers, even if it is not covered in the course.**
* Word2Vec: really powerful training approach (taking words out of the sentence and predicting the next word while training). Also definitely try this tool! Getting the vectors of each word, add the vector of all words, then dividing by the number of words and putting it is an input to the classifier 🡺 but not that great for *long texts*! Works better if you remove words, e.g. a **TF-IDF subset of an article/doc.** First extract keywords with TF-IDF, as **stopwords will “dilute” the final vector.** **Word2Vec works best when you can compress the texts into a SUSBSET of important words, you might need to average that subset of words!** It works better on *shorter texts*, so they need a lot of pre-processing.
* Look at GloVe vectorization.
* Use **HuggingFace** to find resources.
* Look ahead and do the TF-IDF section later!
* **Sentiment Analysis:** problems with *irony*, jokes etc.
* **Summarization of books is required:** **RESEARCH THIS** and stay away from easy example dataset! Look at these book-summarization tools and compare different ones. Look for datasets on Hugging Face, Google and Kaggle. Do not reach for an LLM though! E.g. look for customer complaints, reviews on Amazon.
* **HuggingFace** is a data science “hub”. Kaggle has published datasets with solutions/competitions, and it became a place to put code and example approaches. HF = has datasets, models. Most models are **pre-trained on HF** 🡺 e.g. “meta llama”, an LLM released by Meta. We can download these chat-LLMs and look at how you could train it for your particular use case.
* **HuggingFace** is where most research is published! Models from Facebook, NVIDIA and Microsoft will be published here. They have a number of different tasks they can solve including text classification.
* **Check the models out! Existing literature/research behind it needs to be checked out**.
* **E**.g. Meta Llama on HF. Also *mini-gpt-v4*. VisionCAIR from Berkeley. Sometimes you can even try the LLMs/models before you download them, but some of them are HUGE – so look for *smaller versions* of the LLMs. You might not be able to train it on time/too big = just drop it in your coursework, as it might just take too long.
* **Start with a sample of the dataset = if it works, then continue, otherwise change**!
* Start with a simple solution (e.g. Naïve Bayes), if it works well, then look for other models! This is the data science process.
* Choose a difficult dataset! Say **WHY** you are trying different approaches and how they work!

**My Questions**

* Dataset selection 🡪 can we use Kaggle (yes/no)? *Was the organization allowed to use the dataset? Were they* ***allowed*** *to share it?* *You have to be LICENSED to use them!* *Look at* ***HuggingFace*** *as well. Find a challenging dataset, not very easy (don’t just do spam vs not spam and then logistic regression).*
* Do we have to build the classifier (e.g. Naïve Bayes) from scratch? Or can use libraries? Where are the marks allocated? *No, not coding the algorithm from scratch, use scikit-learn.*
* Are there marks allocated for collecting your own dataset? *At least 300 documents labelled, if clustering algorithm might not need any. Please be aware that we have to train a CLASSIFIER, not k-means or unsupervised classifiers.*
* Are we expected to have multiple notebooks? Do we have to construct Python classes in separate *.py* scripts? *Use markdown as much as poss, use 1 jup notebook here. Multiple HASHTAGS to create subtitles + sections.*
* Can we use BERT and these more advanced transformer-based models even if they have not been covered in the course material so far? *Yes, but we have to TRAIN them 🡪 as “extra work”, explain* ***how you trained it****, what examples you used. Explain the results*.
* Do we need to use English texts?
* Negation: construct underscores between the *no, not* words and the word they negate.
* You have to **try all approaches** for text processing, there is no right answer.
* Finding the right model + right data combinations/preprocessing techniques and comparing them.
* *Collecting the dataset: it depends on your classifier.*