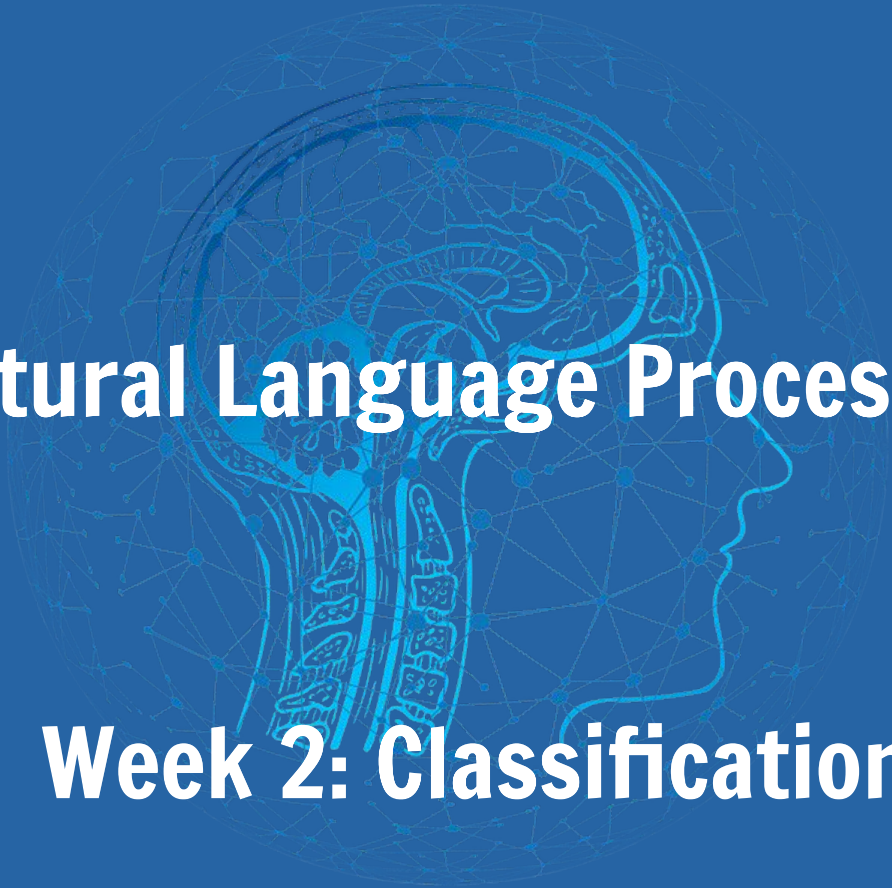
Imperial College
London



Natural Language Processing

Week 2: Classification

Classification
Lecture 2

Outline

1. De-biasing

2. Coursework specification

I have moved evaluation to next Monday’s lecture….

Model de-biasing

(preventing a model learning from shallow heuristics)

Motivation

Self driving cars:

- Humans can learn to drive from just 80 hours
- Machine learning models learn from millions of hours of driving data - but still can fail.

How can we make AI (and NLP) models generalise better?

Natural Language Inference

- **Premise**: The kitten is climbing the curtains again
- **Hypothesis**: The kitten is sleeping

Labels:

- *Entailment*
- *Contradiction*
- *Neutral*

Debiasing

Possible strategies:

- Augment with more data so that it ‘balances’ the bias
- Filter your data
- Make your model predictions different from predictions based on the bias (robust & shallow models)
- Prevent a classifier finding the bias in your model representations

Debiasing

We combine together:

- The probabilities of each class given our bias (b_i)...
  - I will call these the “bias probabilities”

- … and the probabilities from our model (p_i)

Is this notation clear?

The next slide shows the Product of Experts method for de-biaisng

It looks simple, but it’s really interesting

Debiasing

We combine together:

- The probabilities of each class given our bias (b_i)...
  - I will call these the "bias probabilities"

- … and the probabilities from our model (p_i)

When training your robust model:

$$\hat{p_i} = softmax(\log(p_i) + \log(b_i))$$

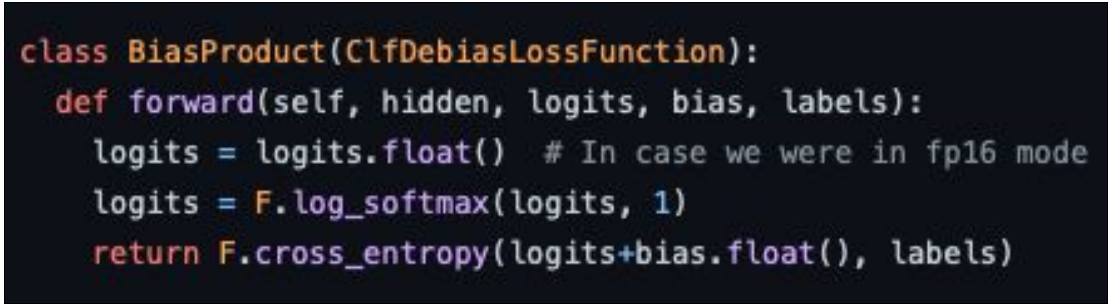
Debiasing

We combine together:

- The probabilities of each class given our bias (b_i)...

- … and the probabilities from our model (p_i)

$$\hat{p}_i = softmax(\log(p_i) + \log(b_i))$$



```
class BiasProduct(ClfDebiasLossFunction):
  def forward(self, hidden, logits, bias, labels):
    logits = logits.float()  # In case we were in fp16 mode
    logits = F.log_softmax(logits, 1)
    return F.cross_entropy(logits+bias.float(), labels)
```

Code from Clark et al (2019):
“Don’t Take the Easy Way Out Ensemble Based Methods for Avoiding Known Dataset Biases”

Debiasing

How can you get the ‘bias probabilities’ b_i:

- Target a specific known ‘biased’ feature

Debiasing

How can you get the ‘bias probabilities’ b_i:

- Target a specific known ‘biased’ feature

Or creating a ‘biased’ model…

Debiasing

Creating a ‘biased’ model:

1. Use a model not powerful enough for the task (e.g. BoW model, or TinyBERT)
2. Use incomplete information (e.g. only the hypothesis in NLI)
3. Train a classifier on a very small number of observations

Debiasing

Another approach is to weight the loss of examples based on the performance of our biased model:

- When training the robust model, multiply the loss by:
  $1 - b_i$ (now b_i refers to the bias model probabilities for the correct class)

Clark et al (2019):
“Don’t Take the Easy Way Out Ensemble Based Methods for Avoiding Known Dataset Biases”

In this implementation, the loss is normalized across each minibatch so the total minibatch loss remains the same

Debiasing

When is it desirable to stop a model learning from shallow-heuristics in the dataset?

Debiasing

- You will find results something like

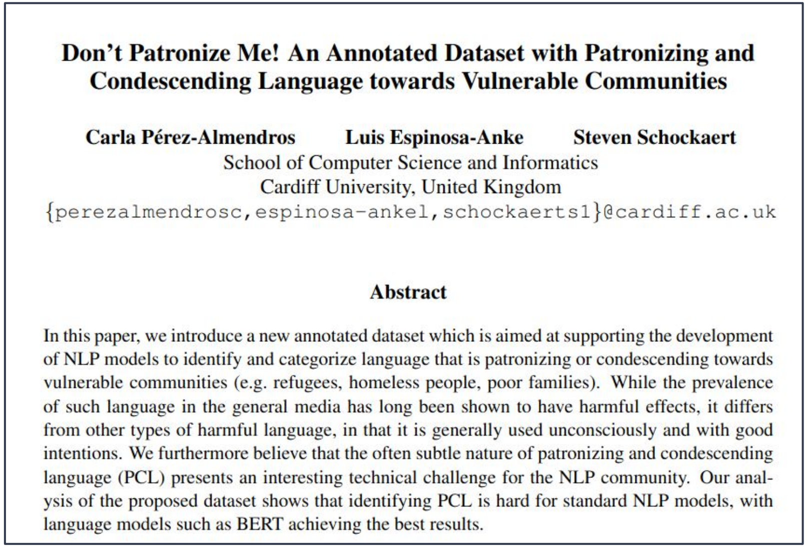
	In-distribution test set	**out-of-distribution test set**
Normal training	Model performs great	Not so great
Training with PoE	Little bit worse than normal training	Better than normal training

Coursework spec

25 mins

Coursework introduction

The task is a binary classification task, to classify whether a text contains condescending or patronising language:



Don't Patronize Me! An Annotated Dataset with Patronizing and Condescending Language towards Vulnerable Communities

Carla Pérez-Almendros **Luis Espinosa-Anke** **Steven Schockaert**
School of Computer Science and Informatics
Cardiff University, United Kingdom
{perezalmendrosc,espinosa-ankel,schockaerts1}@cardiff.ac.uk

Abstract

In this paper, we introduce a new annotated dataset which is aimed at supporting the development of NLP models to identify and categorize language that is patronizing or condescending towards vulnerable communities (e.g. refugees, homeless people, poor families). While the prevalence of such language in the general media has long been shown to have harmful effects, it differs from other types of harmful language, in that it is generally used unconsciously and with good intentions. We furthermore believe that the often subtle nature of patronizing and condescending language (PCL) presents an interesting technical challenge for the NLP community. Our analysis of the proposed dataset shows that identifying PCL is hard for standard NLP models, with language models such as BERT achieving the best results.

Coursework introduction

Category:	**County code**:	**Paragraph:**	**Label:**
homeless	nz	Call to restore hope for homeless through inquiry	4
poor-families	gb	More than 100,000 poor families have already been evicted , the report says	0

Test / Official dev / Internal dev

Coursework introduction

Annotations:

2 annotators have been asked to give a score of:

- 0 (no PCL)
- 1 (borderline PCL), or
- 2 (contains PCL)

Training labels*:

These scores are converted to:

- 0 (0 from both annotators)
- 1 (0 and 1 scores)
- 2 (1 from both annotators)
- 3 (1 and 2 scores from annotators)
- 4 (2 from both annotators)

Test labels:

These scores are converted to:

- No PCL (0 and 1)
- PCL (2, 3, 4)

***Scores of 0 and 2 used a 3rd annotator**

Coursework introduction

Ultimately:

- Your task is to implement a transformer model which outperforms a RoBERTa-base baseline

- We ask you to analysis the training data, and answer 3 analysis questions

- You will be marked based on your report, using the spec provided.

- For your model improvements, as long as you are outperforming the baseline, it does not matter if these ideas work

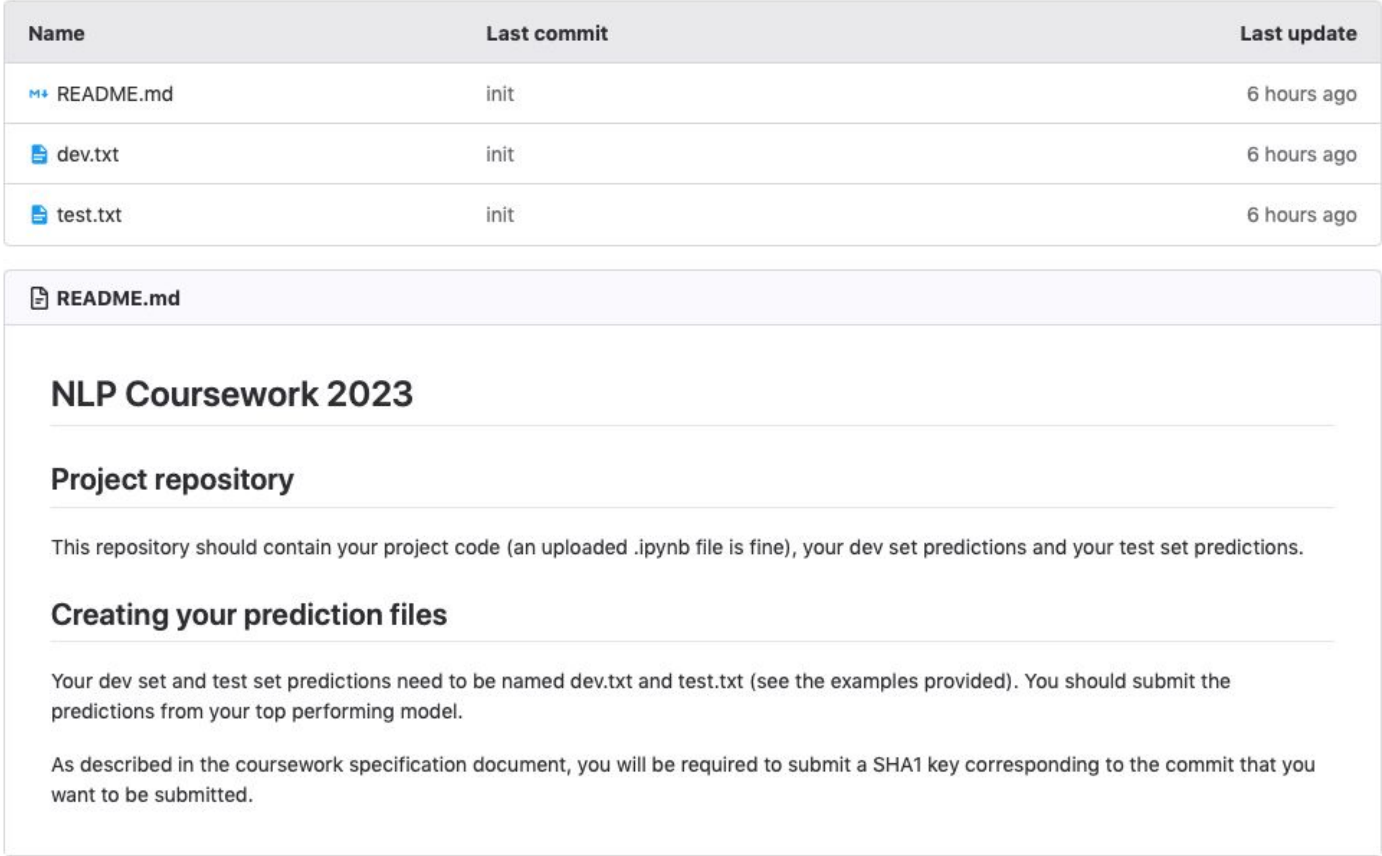
Coursework introduction

A few more things:

- Maximum 5 page count on a template we provide
- Coursework counts for 30%, and the deadline is 7pm on 7th March
- Outputs: your report, with your LabTS submission (for dev and test predictions). You also need to upload your code.
- You should request the data today
  - please use this link:

  https://docs.google.com/forms/d/e/1FAIpQLSe5KyzXgpnEOjS-Y6Gb8TTKiWxh4_qLuPL-NGiqKCyF41ALlg/viewform

Student repo



Name	Last commit	Last update
README.md	init	6 hours ago
dev.txt	init	6 hours ago
test.txt	init	6 hours ago

README.md

NLP Coursework 2023

Project repository

This repository should contain your project code (an uploaded .ipynb file is fine), your dev set predictions and your test set predictions.

Creating your prediction files

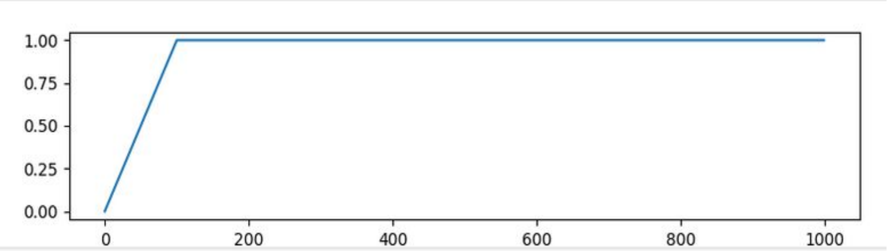
Your dev set and test set predictions need to be named dev.txt and test.txt (see the examples provided). You should submit the predictions from your top performing model.

As described in the coursework specification document, you will be required to submit a SHA1 key corresponding to the commit that you want to be submitted.

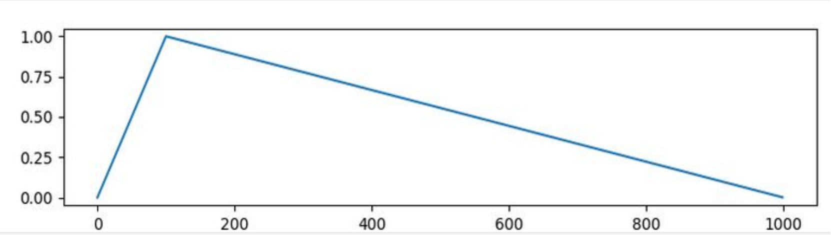
Coursework introduction

Learning schedules:

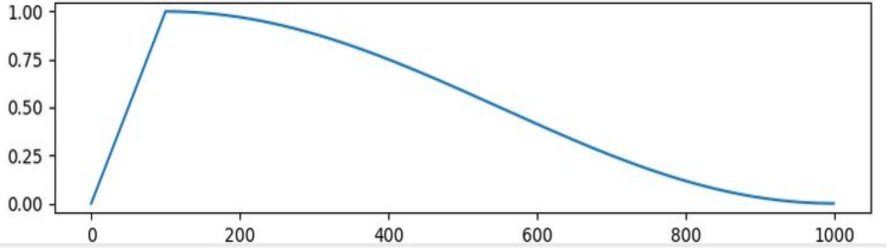
Warm-up period



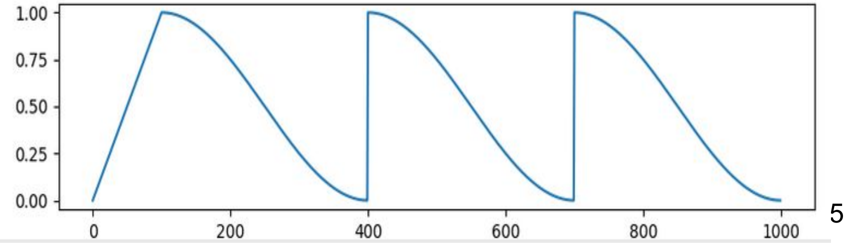
Linear schedule with warm-up



Cosine with warm-up



Cosine schedule with hard resets

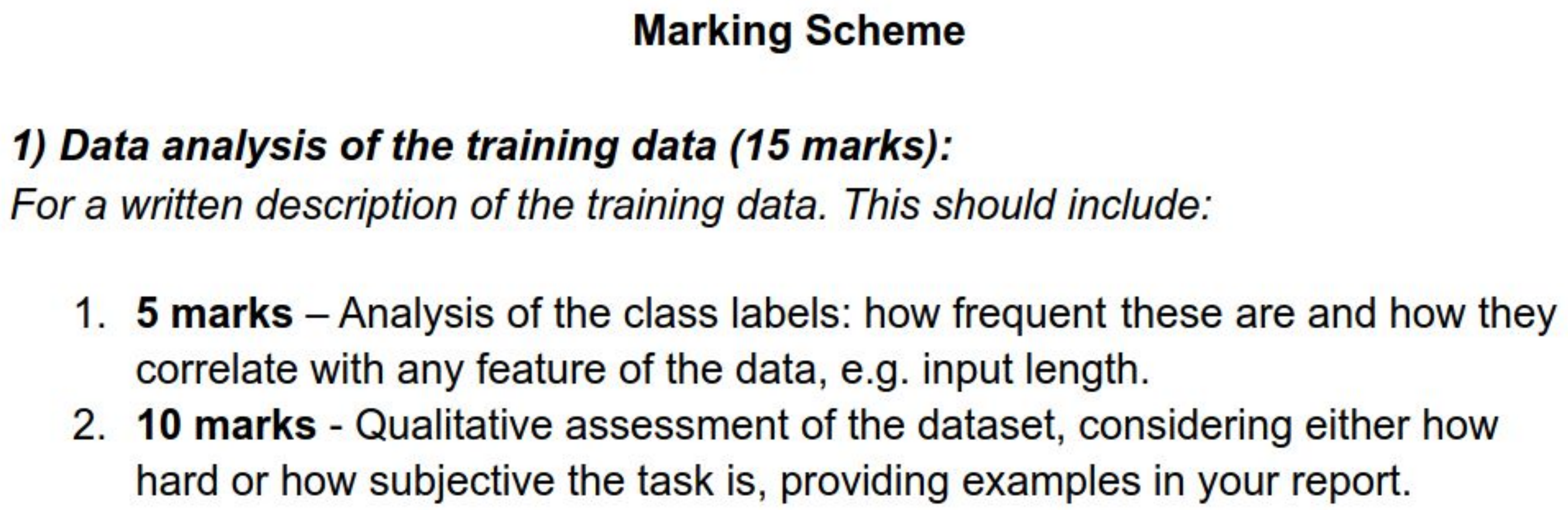


The detailed mark scheme

40 mins

Coursework introduction:

Spec part 1:



Marking Scheme

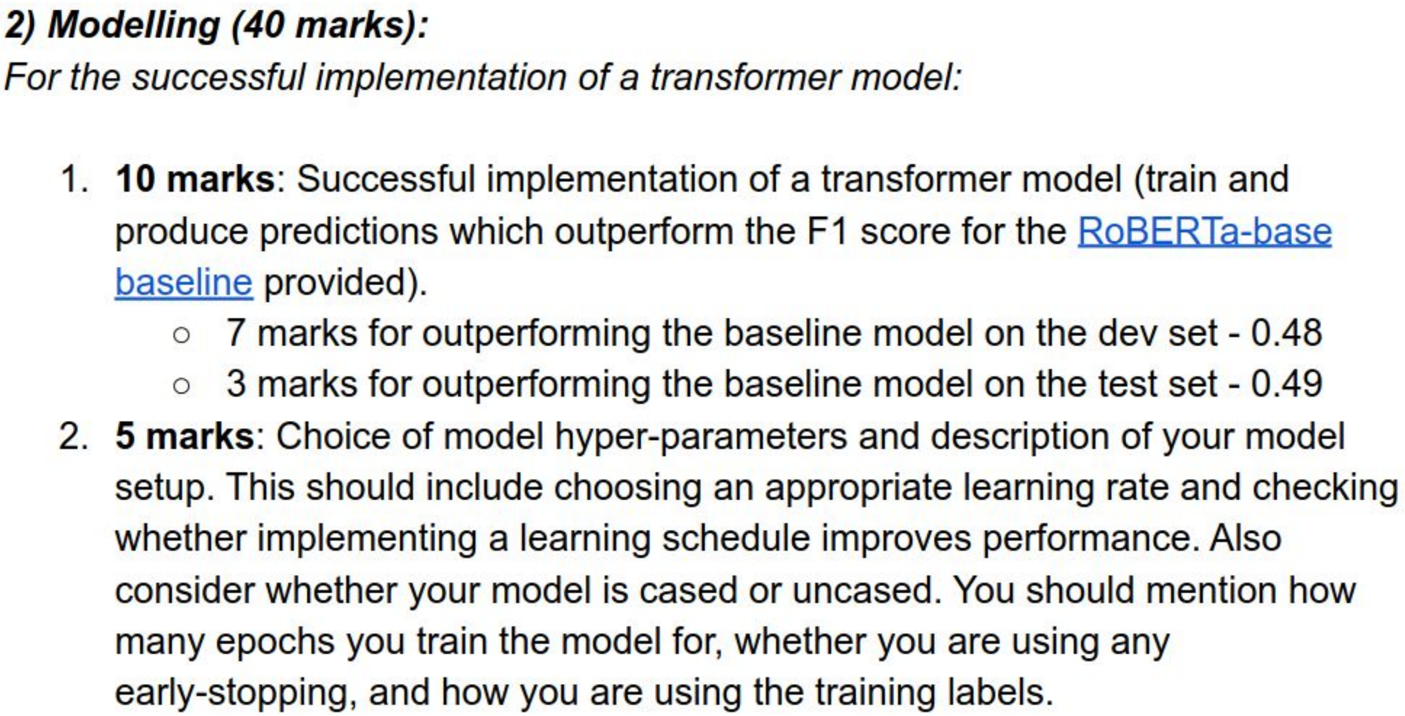
1) Data analysis of the training data (15 marks):

For a written description of the training data. This should include:

1. **5 marks** – Analysis of the class labels: how frequent these are and how they correlate with any feature of the data, e.g. input length.
2. **10 marks** - Qualitative assessment of the dataset, considering either how hard or how subjective the task is, providing examples in your report.

Coursework introduction:

Spec part 2:



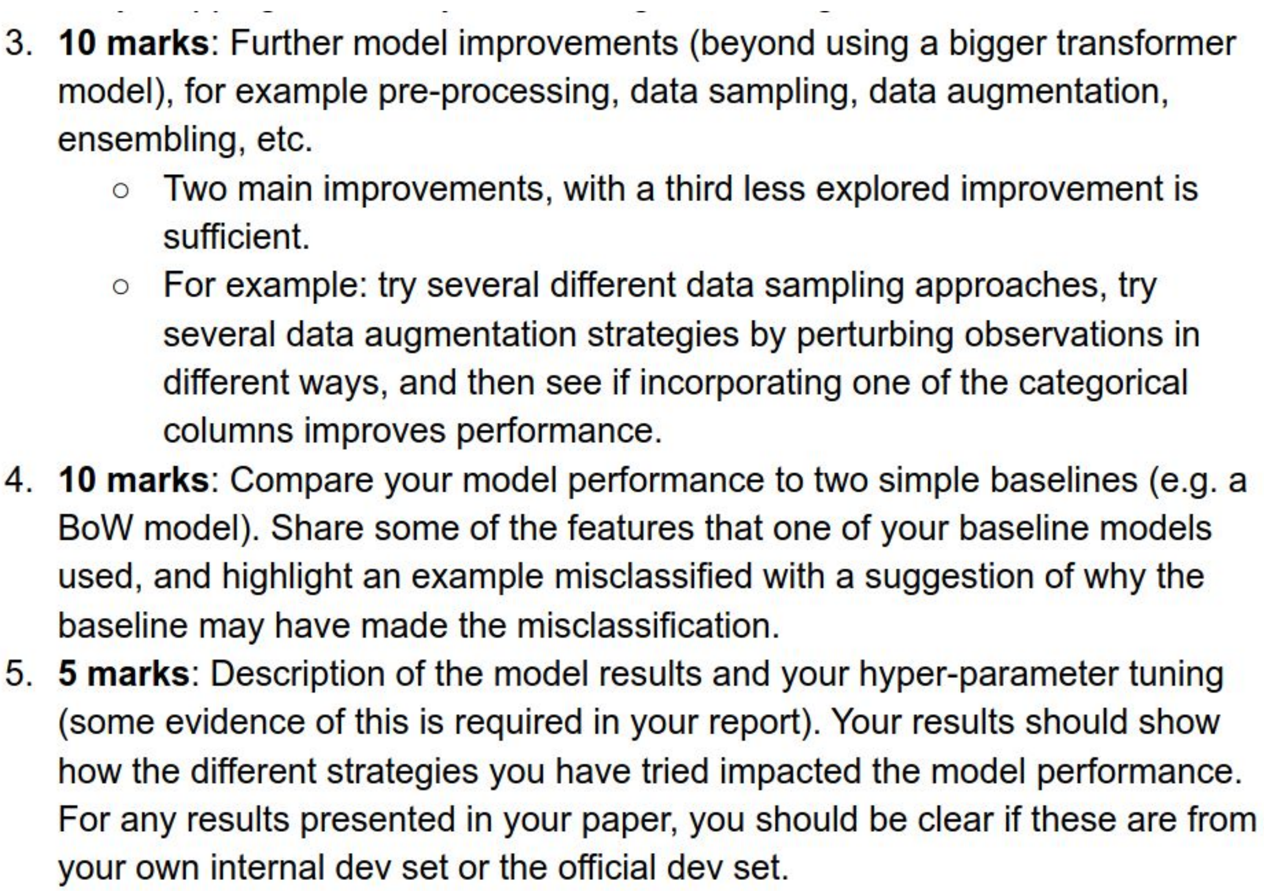
2) Modelling (40 marks):

For the successful implementation of a transformer model:

1. **10 marks**: Successful implementation of a transformer model (train and produce predictions which outperform the F1 score for the RoBERTa-base baseline provided).
    - 7 marks for outperforming the baseline model on the dev set - 0.48
    - 3 marks for outperforming the baseline model on the test set - 0.49
2. **5 marks**: Choice of model hyper-parameters and description of your model setup. This should include choosing an appropriate learning rate and checking whether implementing a learning schedule improves performance. Also consider whether your model is cased or uncased. You should mention how many epochs you train the model for, whether you are using any early-stopping, and how you are using the training labels.

Coursework introduction:

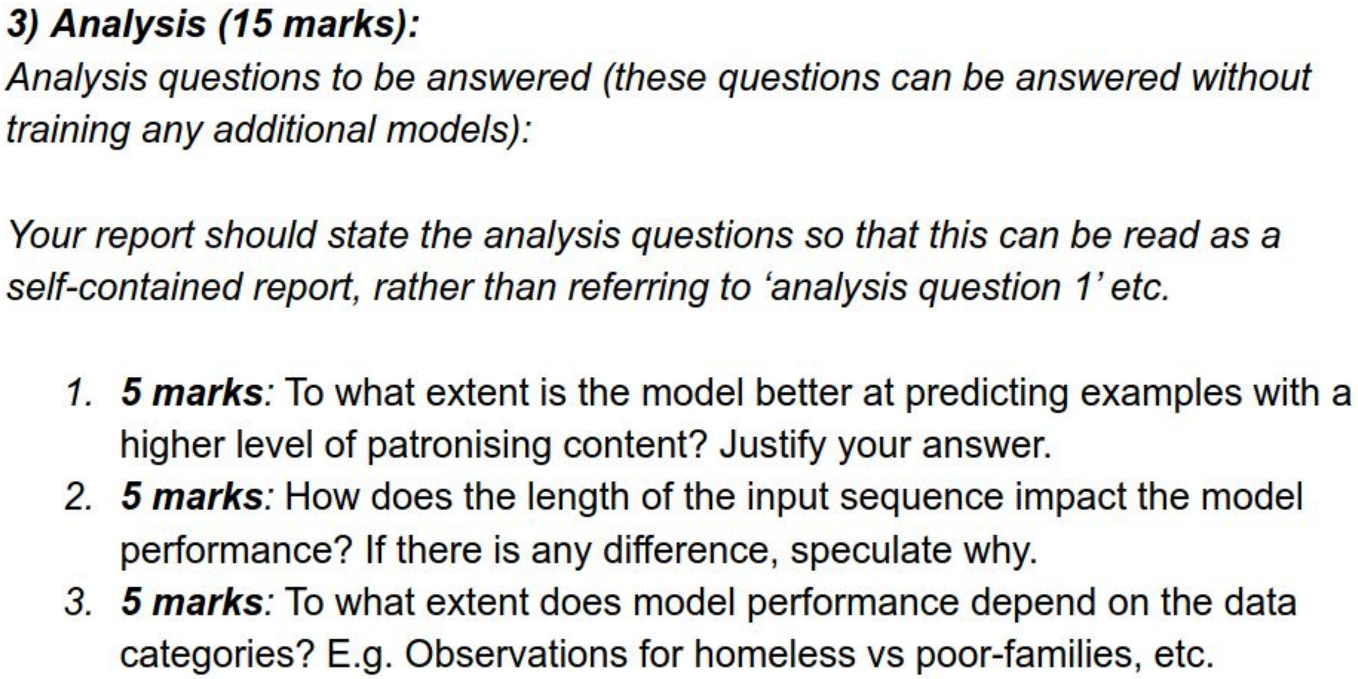
Spec part 3:



3. **10 marks**: Further model improvements (beyond using a bigger transformer model), for example pre-processing, data sampling, data augmentation, ensembling, etc.
   - Two main improvements, with a third less explored improvement is sufficient.
   - For example: try several different data sampling approaches, try several data augmentation strategies by perturbing observations in different ways, and then see if incorporating one of the categorical columns improves performance.
4. **10 marks**: Compare your model performance to two simple baselines (e.g. a BoW model). Share some of the features that one of your baseline models used, and highlight an example misclassified with a suggestion of why the baseline may have made the misclassification.
5. **5 marks**: Description of the model results and your hyper-parameter tuning (some evidence of this is required in your report). Your results should show how the different strategies you have tried impacted the model performance. For any results presented in your paper, you should be clear if these are from your own internal dev set or the official dev set.

Coursework introduction:

Spec part 4:



3) Analysis (15 marks):

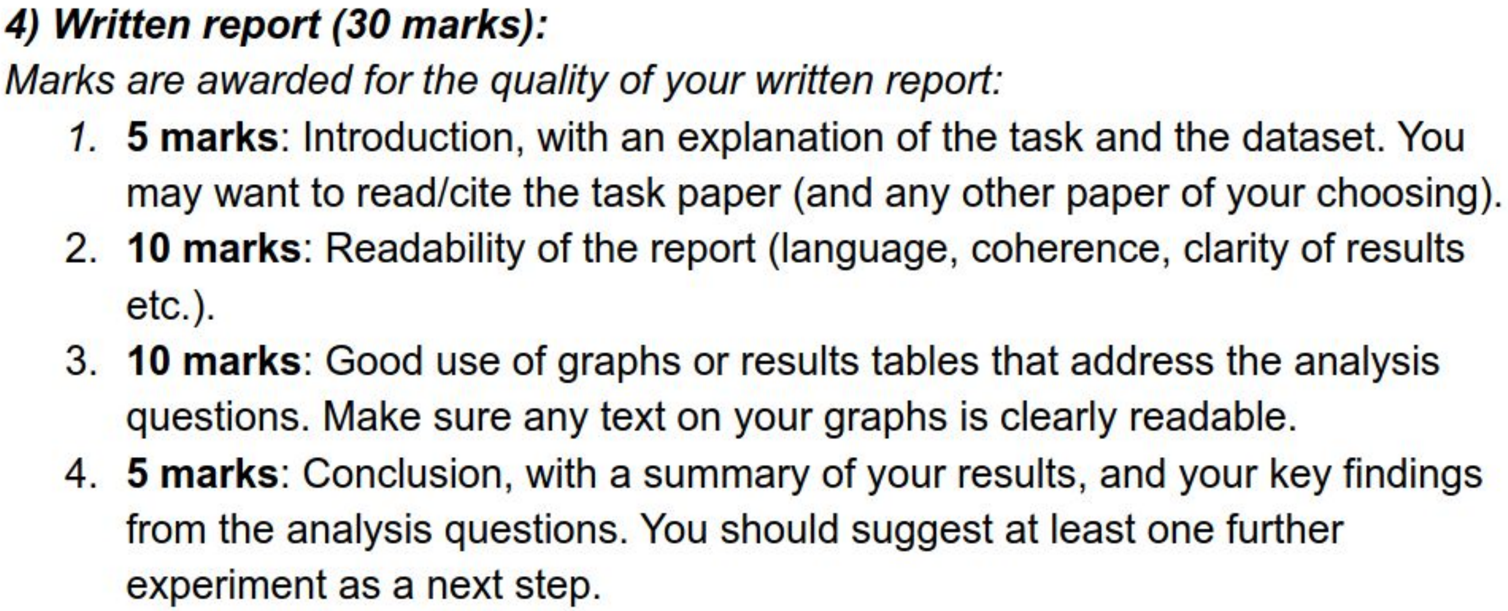
Analysis questions to be answered (these questions can be answered without training any additional models):

Your report should state the analysis questions so that this can be read as a self-contained report, rather than referring to 'analysis question 1' etc.

1. ***5 marks***: To what extent is the model better at predicting examples with a higher level of patronising content? Justify your answer.
2. ***5 marks***: How does the length of the input sequence impact the model performance? If there is any difference, speculate why.
3. ***5 marks***: To what extent does model performance depend on the data categories? E.g. Observations for homeless vs poor-families, etc.

Coursework introduction:

Spec part 5:



4) Written report (30 marks):

Marks are awarded for the quality of your written report:

1. **5 marks**: Introduction, with an explanation of the task and the dataset. You may want to read/cite the task paper (and any other paper of your choosing).
2. **10 marks**: Readability of the report (language, coherence, clarity of results etc.).
3. **10 marks**: Good use of graphs or results tables that address the analysis questions. Make sure any text on your graphs is clearly readable.
4. **5 marks**: Conclusion, with a summary of your results, and your key findings from the analysis questions. You should suggest at least one further experiment as a next step.

Coursework introduction

Feedback survey after the coursework:

- When you receive your marks, you will also have a brief survey to say what you liked and what you didn’t like about the coursework

- This will help us to improve the coursework further for future years

Coursework introduction

We hope you enjoy the coursework!

Using HuggingFace

50 mins

Classification with transformer models

Input:

[CLS]
This
is
an
NLP
lecture
[SEP]

A transformer model

(a black box …for now!)

Bert For Sequence Classification

We have a representation that we can apply a classifier to

(-0.23, 0.6, 0.3, -0.67, 0.12, ….., 0.13, 4.2, -6.2)

↓

Drop out applied

↓

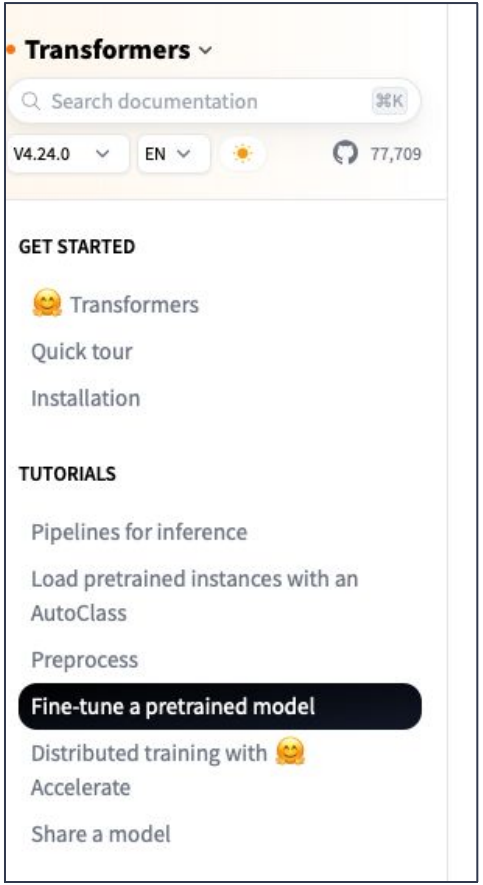
Linear layer creates logits

↓

$$l_n = -\sum_{c=1}^{C} \log \frac{\exp(x_{n,c})}{\sum_{i=1}^{C} \exp(x_{n,i})} y_{n,c}$$

torch.nn.CrossEntropyLoss

Classification with transformer models



Transformers

Search documentation ⌘K

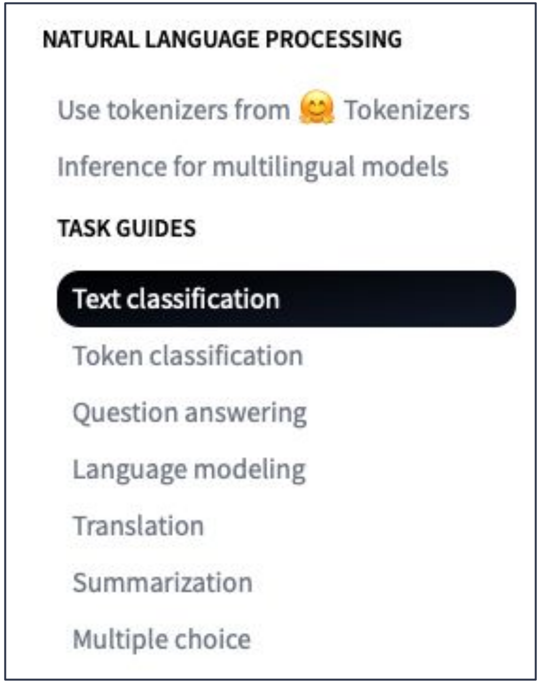
V4.24.0 EN 77,709

GET STARTED

- 🤗 Transformers
- Quick tour
- Installation

TUTORIALS

- Pipelines for inference
- Load pretrained instances with an AutoClass
- Preprocess
- **Fine-tune a pretrained model**
- Distributed training with 🤗 Accelerate
- Share a model

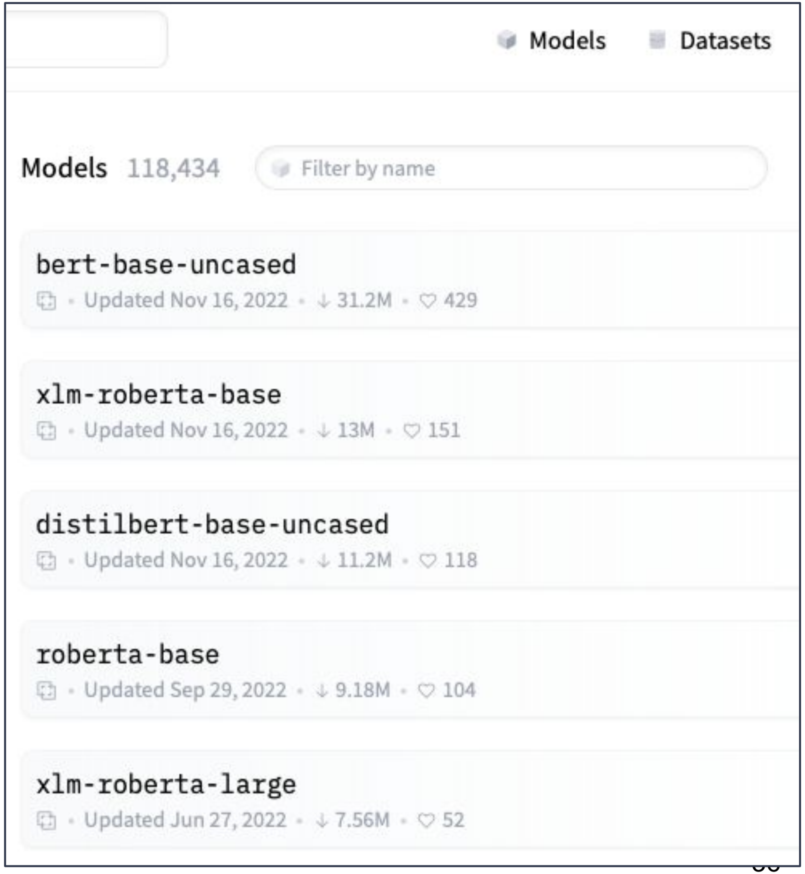


NATURAL LANGUAGE PROCESSING

- Use tokenizers from 🤗 Tokenizers
- Inference for multilingual models

TASK GUIDES

- **Text classification**
- Token classification
- Question answering
- Language modeling
- Translation
- Summarization
- Multiple choice



Models Datasets

Models 118,434 Filter by name

- bert-base-uncased
  Updated Nov 16, 2022 · ↓ 31.2M · ♡ 429
- xlm-roberta-base
  Updated Nov 16, 2022 · ↓ 13M · ♡ 151
- distilbert-base-uncased
  Updated Nov 16, 2022 · ↓ 11.2M · ♡ 118
- roberta-base
  Updated Sep 29, 2022 · ↓ 9.18M · ♡ 104
- xlm-roberta-large
  Updated Jun 27, 2022 · ↓ 7.56M · ♡ 52

Any questions?

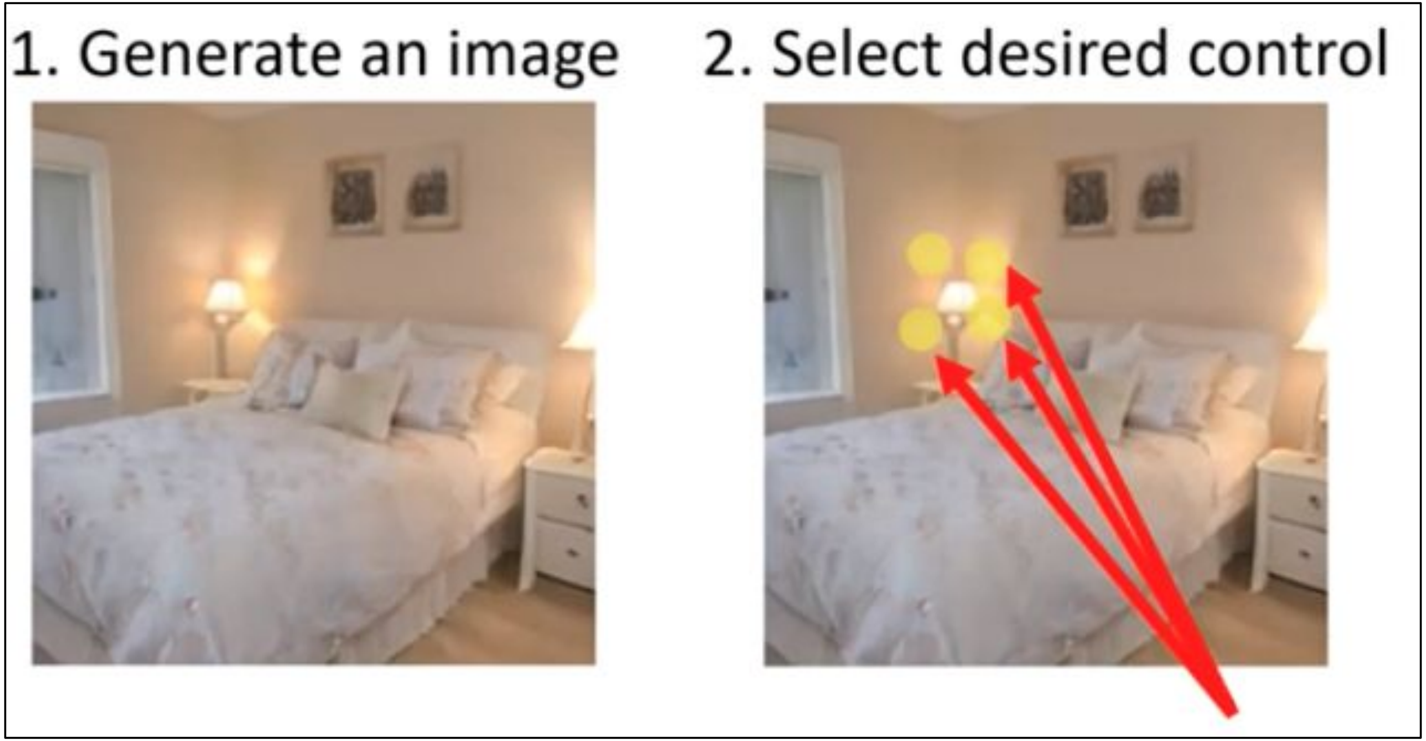
Appendix

40 mins

Something that recently inspired me

For interest only, not assessed

Switching off a lamp by turning off neurons



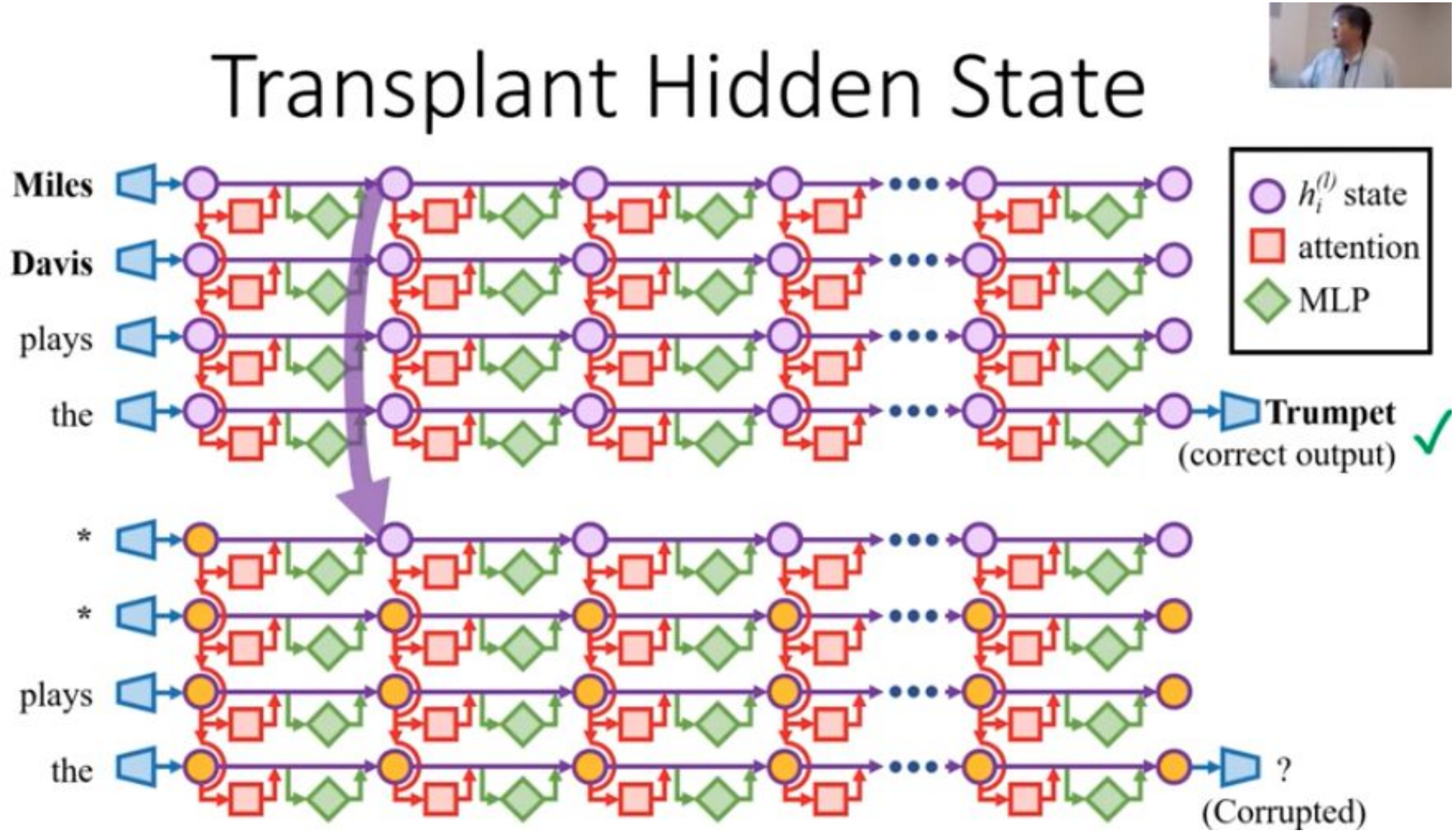
1. Generate an image

2. Select desired control

Based on a talk by David Bau (link to follow)

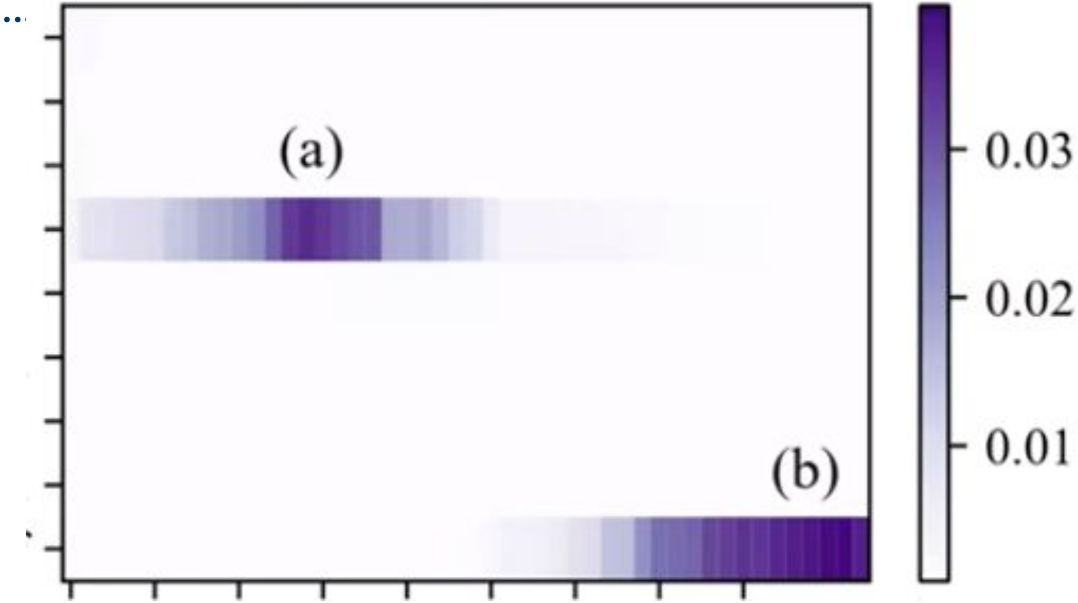
Finding where knowledge is stored...

...



Transplant Hidden State

Finding where knowledge is stored...



Notice: **two** decisive sites!

(a) early
(b) late

Finding where knowledge is stored…

Want to see a talk about this work?

- It’s an hour, but engaging and easy to follow
- The time will fly by…

https://www.youtube.com/watch?v=I1ELSZNFeHc

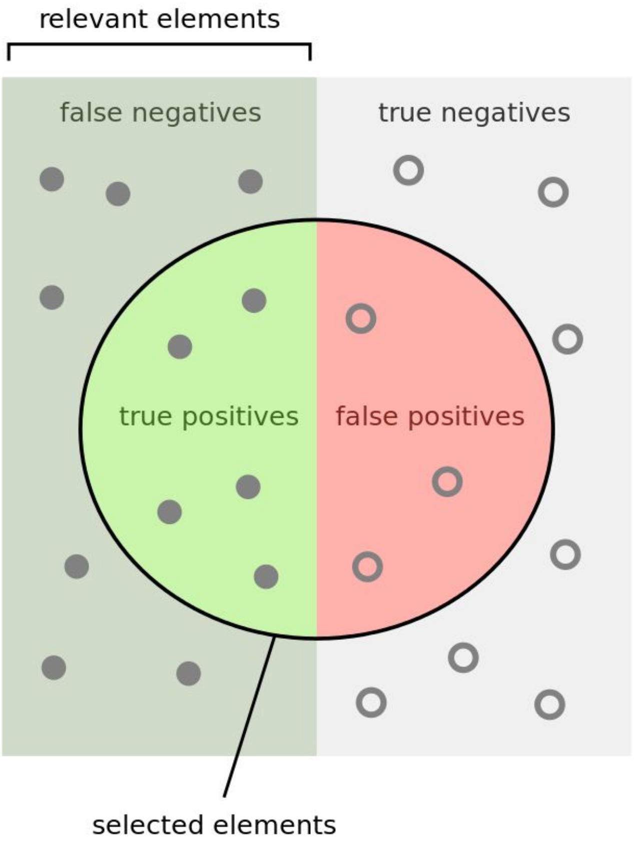
Questions so far?

Evaluation metrics for classification

- To be covered next week

10 mins

Evaluation

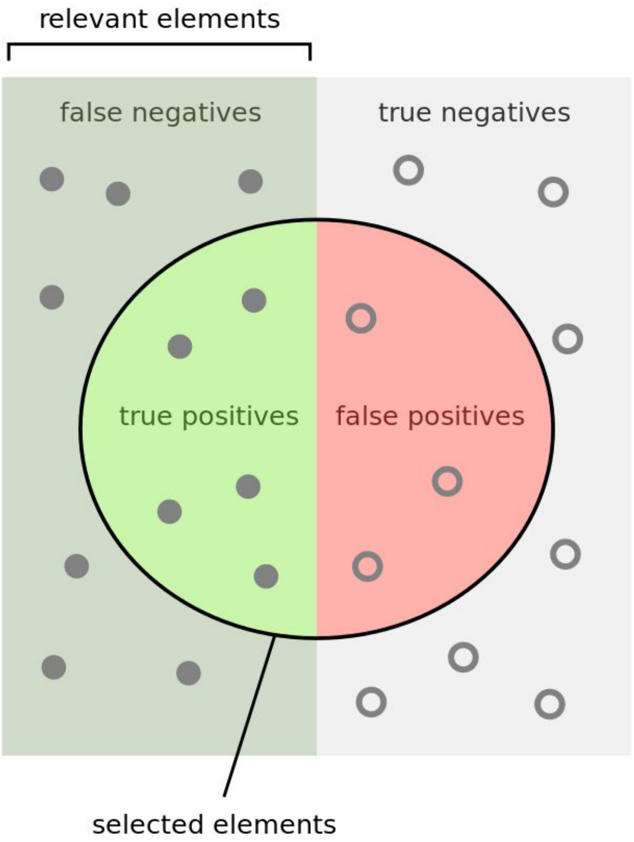


F-measure:

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

Evaluation



For two classes:

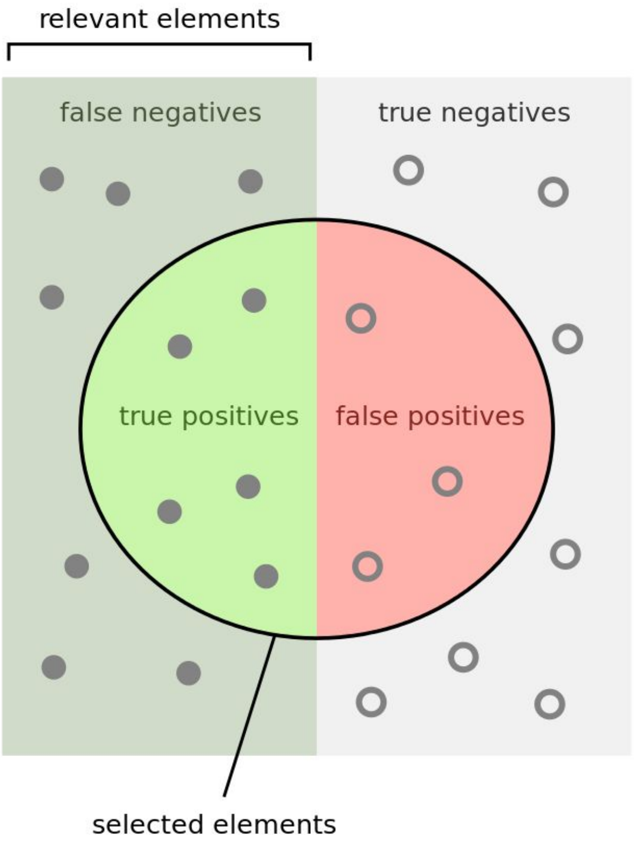
$$accuracy = \frac{TP+TN}{TP+FP+TN+FN}$$

$$TN = 999900$$

$$FN = 100$$

$$accuracy = \frac{999900}{1000000} = 99.99\%$$

Evaluation



For two classes:

$$accuracy = \frac{TP+TN}{TP+FP+TN+FN}$$

$$TN = 999900$$

$$FN = 100$$

Micro vs Macro F1:

- Macro averaging:
  - Increases the emphasis on less frequent classes

Let’s now see something really cool

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$$

$$Accuracy = \frac{\sum_i^C TP_i}{|Dataset|}$$

$$F1 = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$$

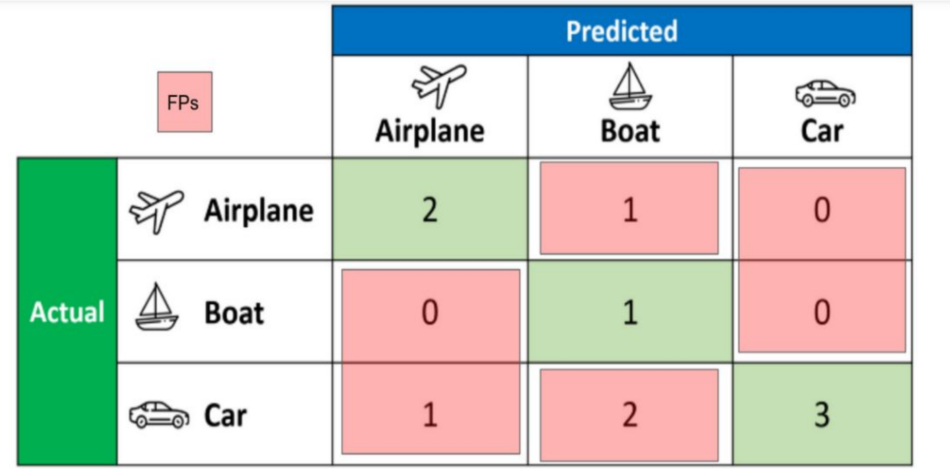
Micro averaged

$$F1 = \frac{\sum_i^C TP_i}{\sum_i^C TP_i+\frac{1}{2}(\sum_i^C FP_i+\sum_i^C FN_i)}$$

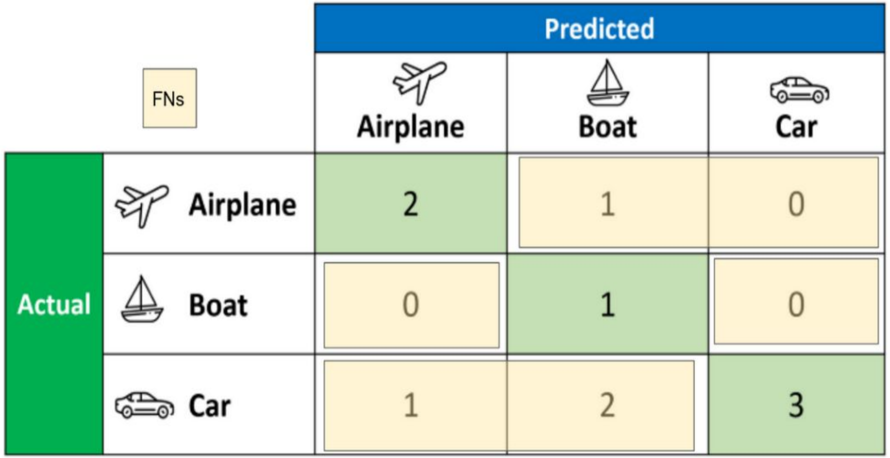
*Assuming each observation has one label

Let’s now see something really cool

$$\textbf{(Micro)}\ F1 = \frac{\sum_i^C TP_i}{\sum_i^C TP_i + \frac{1}{2}(\sum_i^C FP_i + \sum_i^C FN_i)}$$



FPs		Predicted		
		Airplane	Boat	Car
Actual	Airplane	2	1	0
	Boat	0	1	0
	Car	1	2	3



FNs		Predicted		
		Airplane	Boat	Car
Actual	Airplane	2	1	0
	Boat	0	1	0
	Car	1	2	3

$$\textbf{(Micro)}\ F1 = \frac{\sum_i^C TP_i}{\sum_i^C TP_i + \frac{1}{2}(\sum_i^C FP_i + \sum_i^C FN_i)} = \frac{\sum_i^C TP_i}{|Dataset|} = Accuracy$$

Explanation from stackexchange:
https://stats.stackexchange.com/questions/571716/why-true-positives-equals-to-true-negatives-while-calculating-micro-f1-or-accura

Appendix 2

40 mins

Classification with transformer models

Input:

[CLS]
This
is
an
NLP
lecture
[SEP]

A transformer model

(a black box …for now!)

We have a representation that we can apply a classifier to

(-0.23, 0.6, 0.3, -0.67, 0.12, ….., 0.13, 4.2, -6.2)

Classification with transformer models

Input:

[CLS]
This
is
an
NLP
lecture
[SEP]

A transformer model

(a black box …for now!)

You also have representations for each **token** if you wanted to do classification at a token level

(-0.23, 0.6, 0.3, ….., 0.13, 4.2, -6.2)

(-0.13, -0.3, 0.1, ….., 1.3, 9.7, 1.2)

(0.31, -0.1, 3.0, ….., 4.3, 1.2, -1.1)