Learning from Data

Lecture 7: Seq2seq, Best practices, Final project

Rik van Noord - 13 October 2025

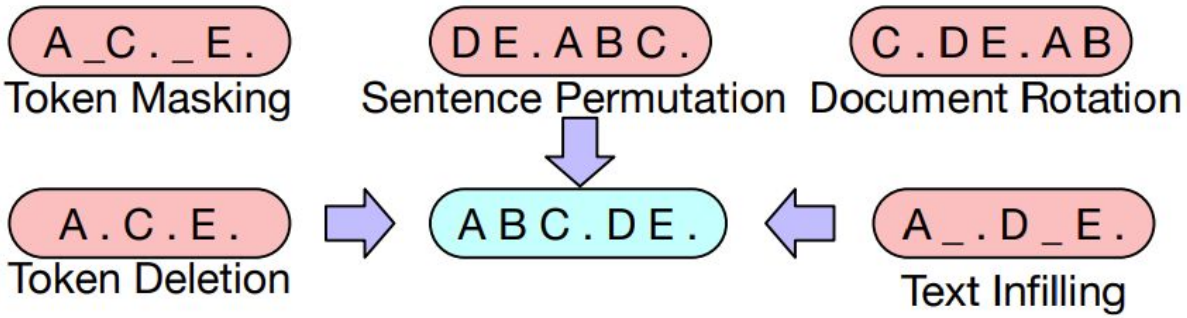
Plan for today

- Assignment and peer review
- Pretrained seq2seq models
- Machine learning best practices
- **Final project**
- Schedule for next weeks

Pretrained seq2seq models

BART (Facebook)

- Train both an encoder and a decoder
- Corrupt text with noise function, train model to reconstruct it
- Model has to output the full original text, not just the fixes



BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. Lewis et. al (2020)

BART: Text Infilling

- Best BART model only uses Text Infilling and Sentence Permutation
- A number of text spans (tokens) is replaced with a **single** <MASK> token
- Potentially multi-word masking
- Model has to figure out the number of words that are missing

Input: The capital <MASK> Paris

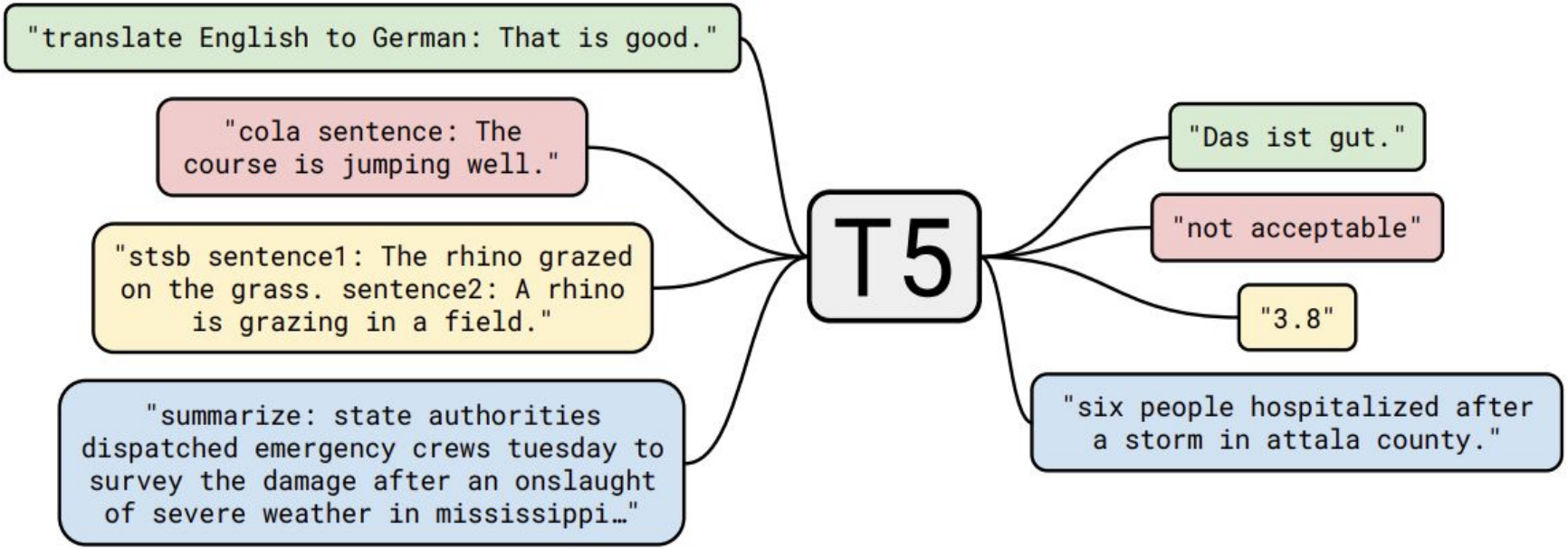
Output: The capital of France is Paris

T5 - Text to Text Transfer Transformer (Google)

- Format **every task** as a sequence-to-sequence (text to text) problem
- Release of a clean common crawl corpus (C4) of 750GB English text
- SOTA on a variety of tasks with their best model of 11B parameters
- Systematic overview of pretraining choices
  - Objective, training time, model size, corpus size
- This is actually a very good paper. You should check it out.

Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Raffel et. al (2020).

Everything is seq-to-seq!



Modelling objectives

Objective	**Inputs**	**Targets**
Language modelling	Thank you for inviting	me to your party last week .

Modelling objectives

Objective	**Inputs**	**Targets**
Language modelling	Thank you for inviting	me to your party last week .
BERT-style	Thank you for **<M>** inviting me to your party **apple** week .	Original text

Modelling objectives

Objective	**Inputs**	**Targets**
Language modelling	Thank you for inviting	me to your party last week .
BERT-style	Thank you for **<M>** inviting me to your party **apple** week .	Original text
Deshuffling	party me for your to . last fun you inviting week Thank	Original text

Modelling objectives

Objective	**Inputs**	**Targets**
Language modelling	Thank you for inviting	me to your party last week .
BERT-style	Thank you for **<M>** inviting me to your party **apple** week .	Original text
Deshuffling	party me for your to . last fun you inviting week Thank	Original text
MASS-style	Thank you **<M>** **<M>** me to your party **<M>** week .	Original text

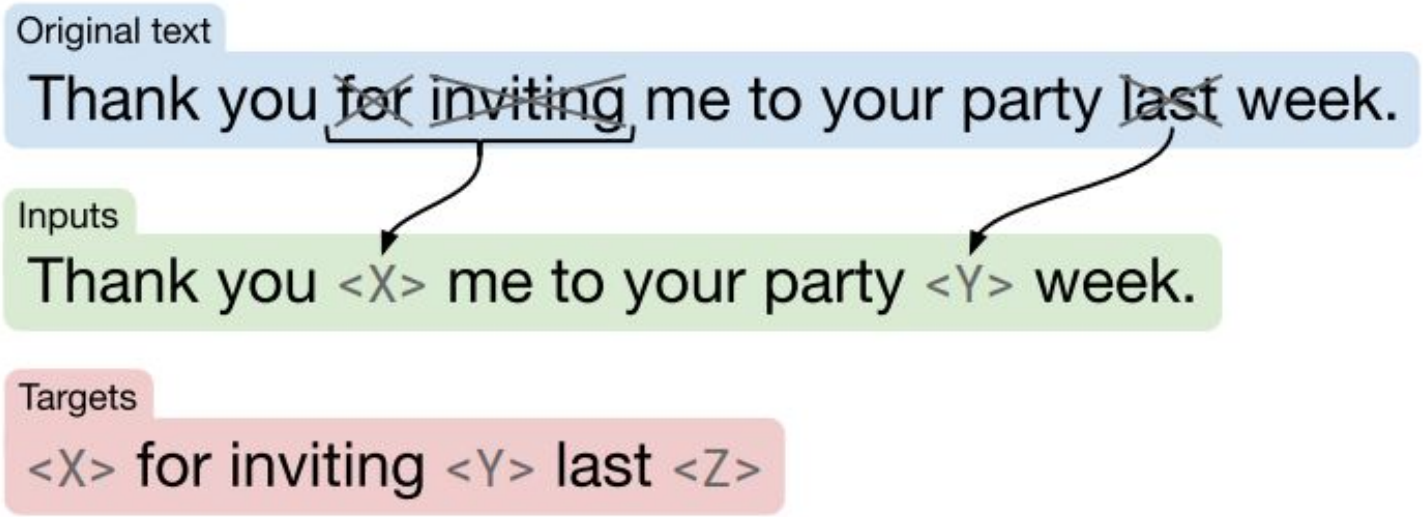
Modelling objectives

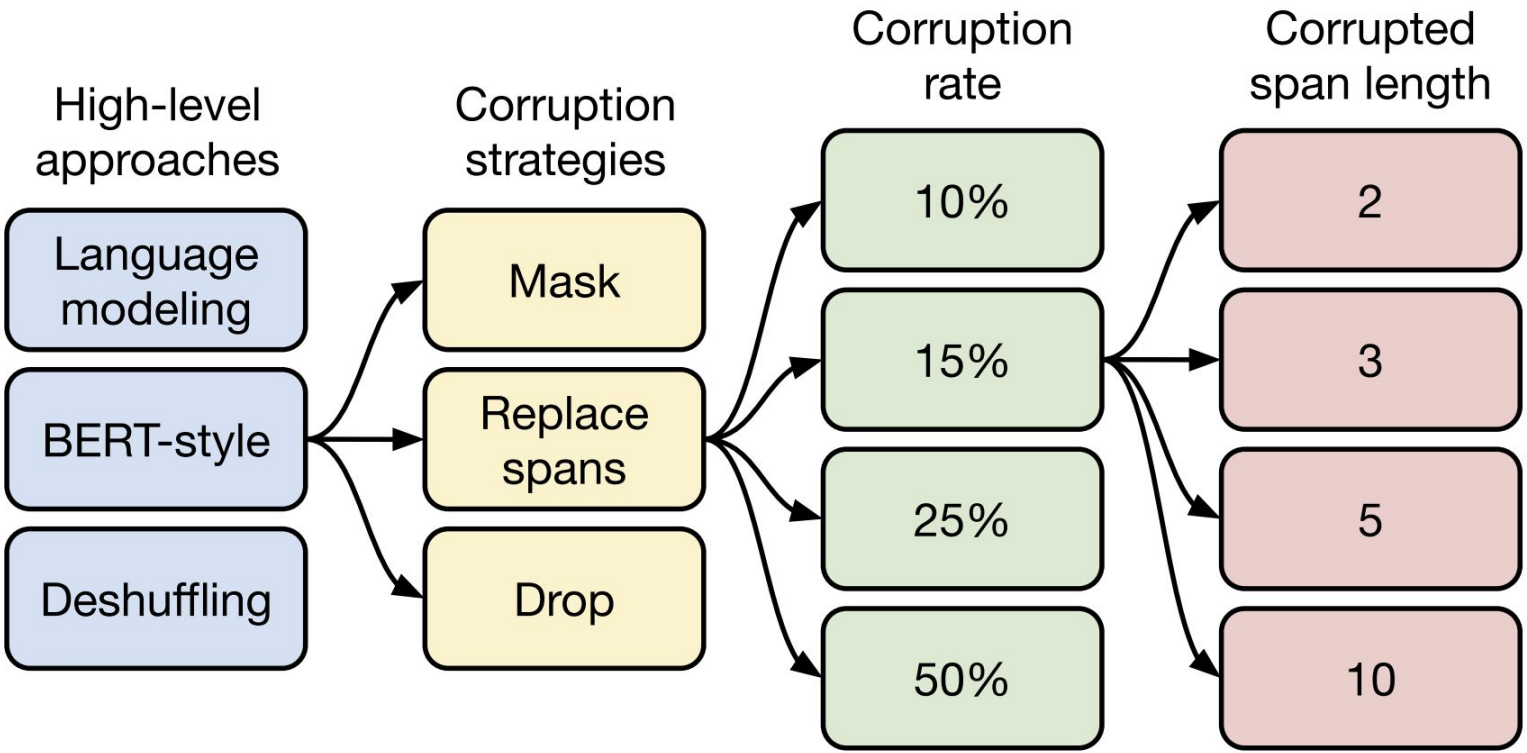
Objective	**Inputs**	**Targets**
Language modelling	Thank you for inviting	me to your party last week .
BERT-style	Thank you for **<M>** inviting me to your party **apple** week .	Original text
Deshuffling	party me for your to . last fun you inviting week Thank	Original text
MASS-style	Thank you **<M>** **<M>** me to your party **<M>** week .	Original text
Drop tokens	Thank you me to your party week .	for inviting last

Modelling objectives

Objective	Inputs	Targets
Language modelling	Thank you for inviting	me to your party last week .
BERT-style	Thank you for **<M>** inviting me to your party **apple** week .	Original text
Deshuffling	party me for your to . last fun you inviting week Thank	Original text
MASS-style	Thank you **<M>** **<M>** me to your party **<M>** week .	Original text
Drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you **<X>** to **<Y>** week .	**<X>** for inviting me **<Y>** your party last **<Z>**

Pretraining objective




High-level approaches
Corruption strategies
Corruption rate
Corrupted span length
Language modeling
BERT-style
Deshuffling
Mask
Replace spans
Drop
10%
15%
25%
50%
2
3
5
10

Multilingual Models

mBART

- BART model trained on 25 languages from Common Crawl
- Works well for MT tasks

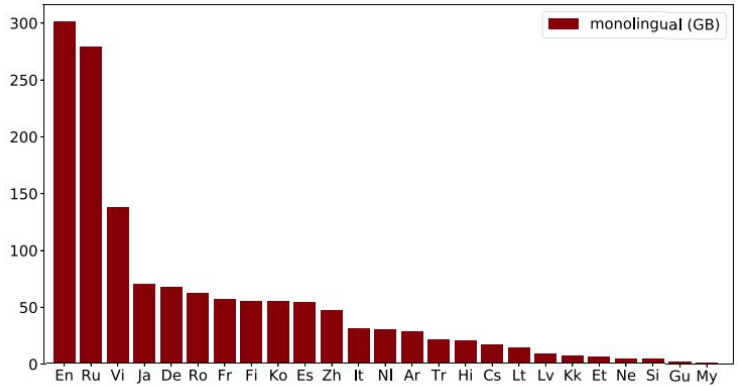


Figure 1: Sizes of the CC25 Corpus. A list of 25 languages ranked with monolingual corpus size.

Multilingual Denoising Pre-training for Neural Machine Translation. Liu et. al (2020)

mT5

- Same as T5, but now multi-lingual
- “Our goal [...] is to create a massively multilingual model that follows T5’s recipe [..] We develop an extended version of the C4 pre-training dataset that covers 101 languages”
- 32,000 -> 250,000 WordPieces as vocab

mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. Xue et al. (2021)

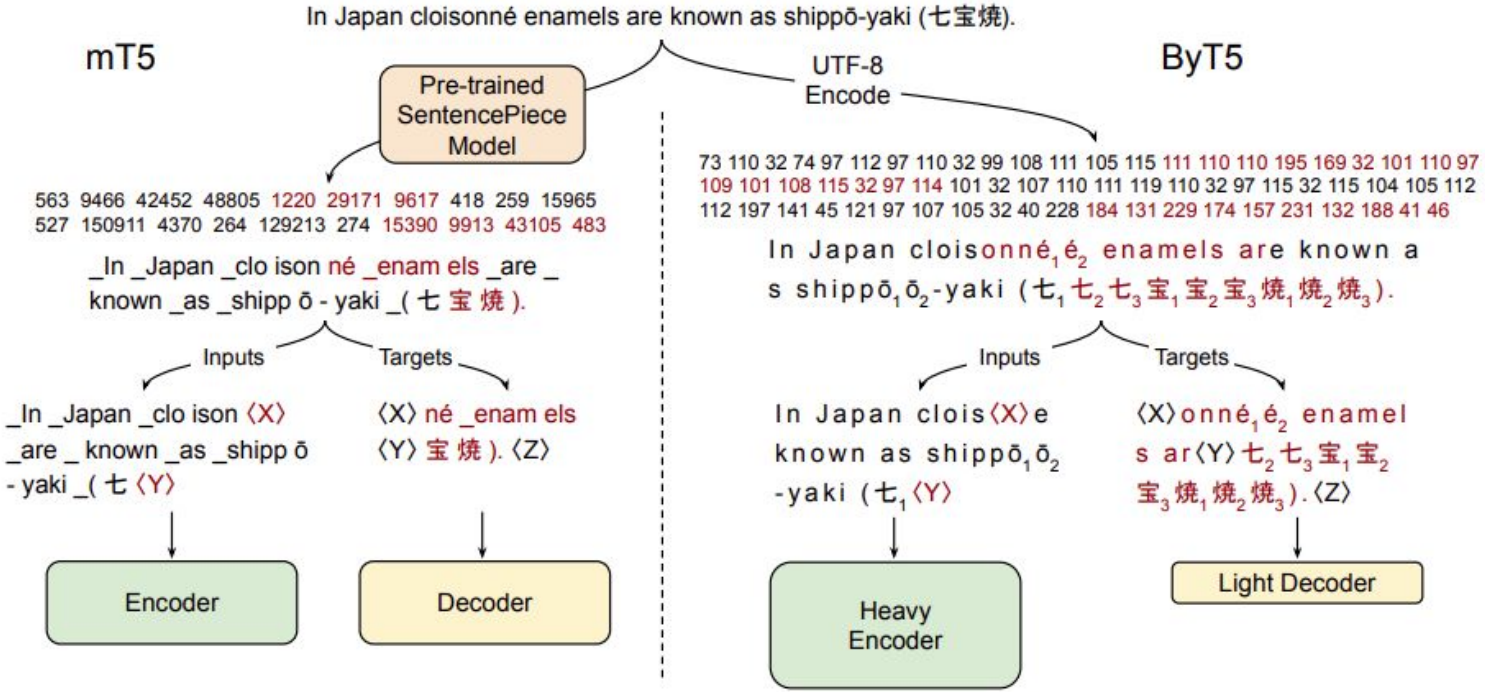
Characters

We use byte-pair encoding for efficiency, not because we think this is the best representation of language there is. We lose information!

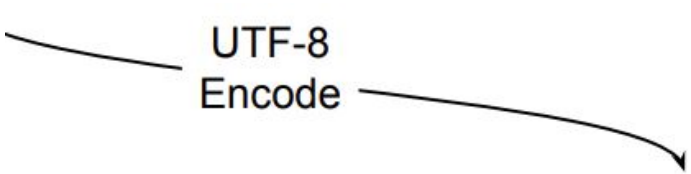
ByT5

- Not even on characters, model directly on the byte-level
- Use UTF-8 encoding
- Very small vocabulary: **very efficient**
- However, very large sequences during training: **very inefficient**

ByT5



[ByT5: Towards a Token-Free Future with Pre-trained Byte-to-Byte Models](). Xue et al (2021)



ByT5

73 110 32 74 97 112 97 110 32 99 108 111 105 115 111 110 110 195 169 32 101 110 97
109 101 108 115 32 97 114 101 32 107 110 111 119 110 32 97 115 32 115 104 105 112
112 197 141 45 121 97 107 105 32 40 228 184 131 229 174 157 231 132 188 41 46

In Japan cloisonné$_1$ é$_2$ enamels are known a
s shippō$_1$ ō$_2$-yaki (七$_1$ 七$_2$ 七$_3$ 宝$_1$ 宝$_2$ 宝$_3$ 焼$_1$ 焼$_2$ 焼$_3$).

ByT5

- Minimal set of modifications with mT5 to limit experimental confounds
- Match total amount of parameters with mT5, i.e. less in the vocab/softmax matrices and more in the layers
- Trained on the same number of “tokens”, which for bytes is 75% less text

		mT5				ByT5				
Size	Params	Vocab	d_{model}	d_{ff}	# Enc/Dec	Vocab	d_{model}	d_{ff}	# Enc	# Dec
Small	300M	85%	512	1024	8	0.3%	1472	3584	12	4
Base	582M	66%	768	2048	12	0.1%	1536	3968	18	6
Large	1.23B	42%	1024	2816	24	0.06%	1536	3840	36	12
XL	3.74B	27%	2048	5120	24	0.04%	2560	6720	36	12
XXL	12.9B	16%	4096	10240	24	0.02%	4672	12352	36	12

Table 1: Comparison of mT5 and ByT5 architectures. For a given named size (e.g. “Large”), the total number of parameters and layers is fixed. The “Vocab” columns indicate the percentage of vocabulary-related parameters, covering both the input embedding matrix and the decoder softmax layer. ByT5 moves these parameters out of the vocab and into the transformer layers, as well as shifting to a 3:1 ratio of encoder to decoder layers.

ByT5 - Results

- English: small models are better than mT5, but large models are not
- Better performance on most realistic cross-lingual benchmarks
- No robust trends regarding morphological complexity, language family, script, character set size, or data availability
- ByT5 is a lot better if you add synthetic noise

More impressive if you consider it was only trained on 25% of text of mT5

Machine Learning Best Practices

All advice I'll give in the rest of this lecture should apply for many of your other projects as well!

What if I don’t have much labeled data?

Working with limited data

- If possible, annotate some data yourself!
  - Future research can benefit from this
  - Gives you a better feeling for the task and data set
  - Often quite effective in terms of improved accuracy given time spent

Otherwise, look into methods of **semi-supervised learning**

- Use same classifier to label more data: **self-training**
- Use a proxy to determine labels: **distant supervision**

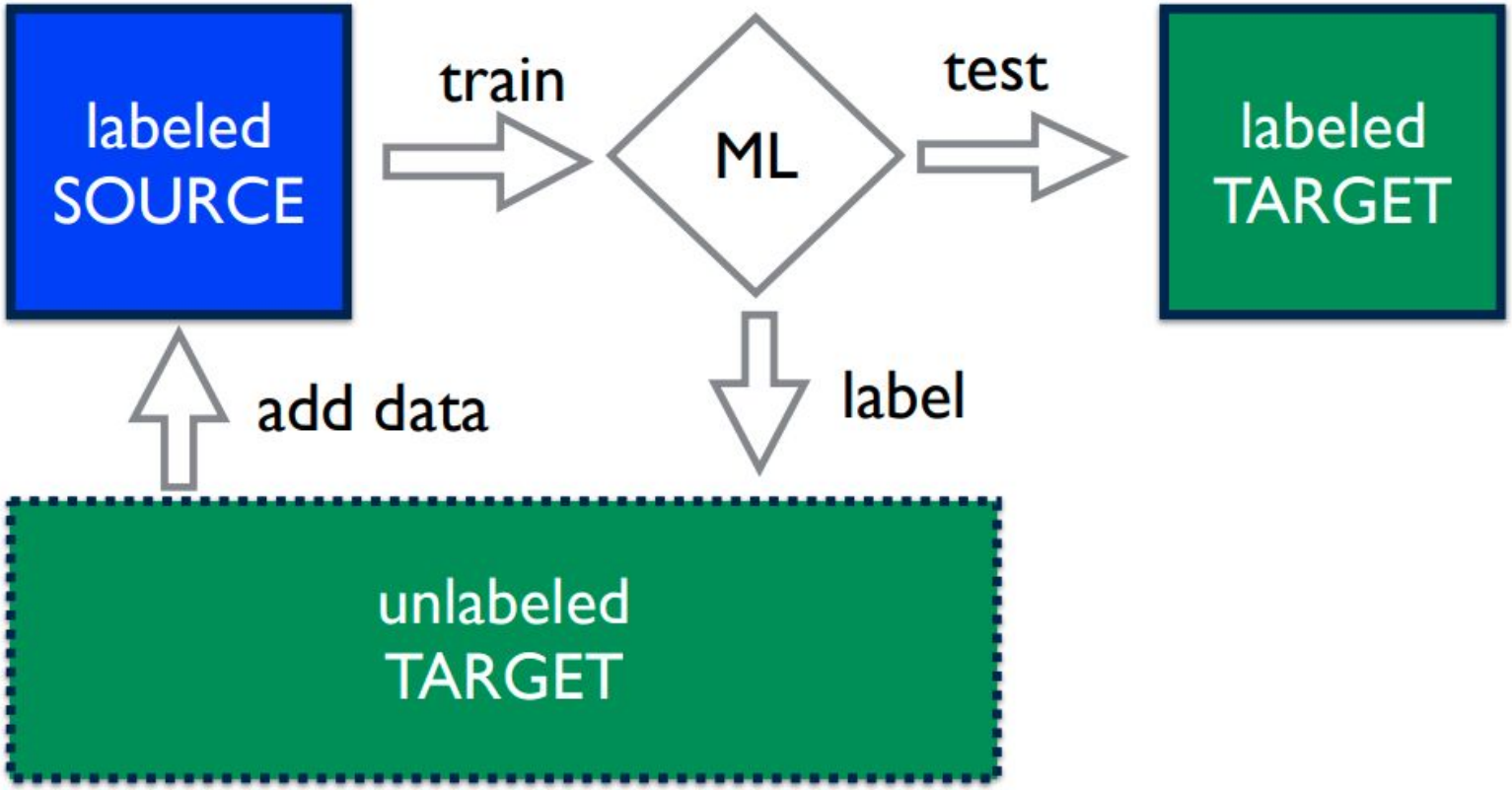
Self-training

Also called **bootstrapping**: the classifier uses its own predictions to train itself

1. Train a model on a data set
2. Use this model to predict on unlabeled sentences
3. Add predictions to original data set
    - Perhaps only if confidence is above some threshold
4. Go to step 1 and repeat until we are satisfied

Important decisions to make:

- How many instances we add at step 3?
- What is the threshold in step 3?
- How many iterations do we perform?


labeled
SOURCE
train
ML
test
labeled
TARGET
add data
label
unlabeled
TARGET

Distant supervision

- This is usually only done if there's no (or barely any) labeled data
- Try to get approximate labeled data (in large amounts) by using a proxy
- This can be very powerful with the right proxy!

Say we want to do **sentiment analysis** on tweets:

- Crawl all tweets that use hashtags #happy and #sad (or other emotions)
- Crawl all tweets with certain emojis
- Crawl all tweets from certain accounts

Note:

- You cannot use this proxy as a feature!
- Highly preferably to **not** use this data as dev or test

How to handle imbalanced data?

Imbalanced data

- If possible: get more data for minority classes
- Accuracy might not tell the full story! Use F-score

Solutions:

- Experiment with different algorithms
  - Some might be more suitable than others!
- Use penalties/weights in training the model
  - In SVM you can weigh classes
  - Loss function that gives more weight to certain classes
- Most common solution: **resampling** of the data

Resampling

Downsampling:

- Remove instances from majority (randomly or some other way)
- Train model on this smaller data set

Upsampling:

- Replicate instances from minority classes (copying or a smarter method)
- Train model on this larger data set

What if the scores vary a lot between different random seeds?

Variation between runs

If there's a large variation between runs, you should first see if you can change the model such that it gives more **stable performance:**

- Change learning rate (might be too low or too high)
- Add regularization (dropout)
- Increase model capacity (features, layers, nodes)
- Run for different number of epochs
- Monitor training loss: some runs simply fail, you can often just discard them
- … and other parts of the model you can think of

Multiple runs per experiment

There will **always** be some variability between runs with different random seeds

Best practice: for each experiment, average over N random seeds (often 3 or 5)

- Keep all settings the same, except the random seed
- Use the same 3 or 5 random seeds for all experiments in the project
- **Do not tune** these random seeds
- Report average and standard deviation, e.g. 81.4 ± 0.6
- Keep this in mind when evaluating results:
  - Is there really a difference between 81.2 ± 0.5 and 81.4 ± 0.6?

This is **not required** to do for the project, but encouraged.

Some advice about the process

Data

- Look at the data!
  - Does the data make sense?
  - Could you annotate it?
- Look at document/sentence lengths and how clean the data is
  - Do you need any extra steps to clean or normalize it?
  - Remember that all preprocessing should be reported!
- Look at the division of labels
  - Is the data imbalanced? If so, would it be a problem?
- Split data in train/dev/test
  - How much data for dev and test?
  - Random split or based on other features?
  - **Advice:** use actual different files, do not split in your train script

Baselines

- First implement a simple bag-of-words model
  - Probably good to try a few: NB, KNN, SVM, etc
- Look at the scores: are they as you expected?
  - If really **low:** perhaps a problem with the data set
  - If really **high:** also likely a problem with the data set
- Potential issues:
  - **Data leakage:** dev and test contain many similar instances as in the train set
  - **Label leakage:** some features correspond to labels too well
- Look at the best features per class
  - See if they make sense. If not, there is likely a problem with the data
  - Look at instances in which these features occur

Tuning your baseline

Often in machine learning, you come up with a new model that outperformed baseline models and other existing models. Given the current state of NLP, it's likely that you will be using a neural network.

For a fair comparison, you have to **tune** your baseline(s)! Or in other words, optimize all hyperparameters that you also optimize for your own model. If anything, you should put more effort in tuning your baseline than your own model.

This is a big problem in machine learning currently. Please be an exception!

Ablation experiments

Say we built a complex LSTM model, with multiple layers, dropout, a self-training pipeline and an extra linear layer after final LSTM state. This model does very well, but we want to know what components are the most important

We do an ablation experiment to assess the **contribution** of a single component. This is also known as leave-one-out: we leave out a single component at a time

Full model	81.5
Not multi-layer	80.7
No dropout	80.4
No extra linear layer	81.3
No self-training	78.5

Ablation experiments

Say we built a complex LSTM model, with multiple layers, dropout, a self-training pipeline and an extra linear layer after final LSTM state. This model does very well, but we want to know what components are the most important

We do an ablation experiment to assess the **contribution** of a single component. This is also known as leave-one-out: we leave out a single component at a time

Full model	81.5
Not multi-layer	80.7
No dropout	80.4
No extra linear layer	81.3
No self-training	78.5

Advice regarding experiments

- Split training, predicting and evaluating over multiple Python files
  - Train a model and save this trained model: train.py
  - Use this model to predict on dev/test set: predict.py
  - Write output to a file and evaluate performance: evaluate.py
  - Of course, you can run everything with a shell script like pipeline.sh
- **Advantages:**
  - You always know on which files you obtained the results
    - This can often get confusing with multiple experiments
  - You can easily re-run the evaluation only
  - You can easily predict on new data with the saved model (e.g. run on test later)
  - You can easily do error analysis in a later stage
  - You can easily release output files so other people can analyze them
  - You can easily release the trained model so other people can use it

Advice regarding experiments

If you are running experiments that will go in the report, **document everything!**

You probably have already noticed how easily you forget how you obtained certain results, e.g. with what hyper-parameters and with what architecture

Advice:

- Create a folder for each experiment
- Log all information about the model and its settings automatically
- Log all information about the training process (train/dev loss, epochs, etc)
- Save trained model and output files to a folder
- Save result of running evaluation
- Potentially copy train.py, predict.py and evaluate.py to this folder as well
    - Seriously, Python files barely take any space, this way you can always go back and check

```
#!/bin/bash
# Experiment folder is a command line argument
exp_fol=$1
# Data sets are fixed
train="data/train.txt"
dev="data/dev.txt"
# Set variables
log=${exp_fol}/log/
out=${exp_fol}/out/
mod=${exp_fol}/mod/
bkp=${exp_fol}/bkp/
# Create folders in exp folder
mkdir -p $exp_fol $log $out $mod $bkp
# Train, predict and evaluate
python src/train.py -i $train -d $dev -m ${mod}model > ${log}train
python src/predict.py -i $dev -m ${mod}model -o ${out}out > ${log}pred
python src/evaluate.py -i $dev -o ${out}out > ${log}eval
# Backup scripts
cp src/train.py src/predict.py src/evaluate.py $bkp
```

In machine learning, **details matter!**
You have to get them all right for your results to mean anything.

How to avoid machine learning pitfalls: a guide for academic researchers

https://arxiv.org/pdf/2108.02497.pdf

Michael A. Lones

Let’s now focus on the project!

Project

We give you a data set on **Offensive Language Identification** on social media. Given a tweet, you have to train model that can predict whether it is offensive.

You have 2 goals:

- Create a model that works as well as possible on this task
- Coming up with an additional original and creative research question

Project

- Data can be downloaded from Brightspace
  - No need to download the original data set
- Write up results as research paper
- Train a number of machine learning models and analyse results
- This project is 50% of your final grade, and has to be >= 5.5

What you have to hand in:

- Report based on the template structured like an academic paper

Deadline: Monday November 3rd 23:59

Data

- 14,100 English tweets annotated in Zampieri et. al (2019a)
- This data was already used in a **shared task**
  - Competition between researchers to find the best model, the test set is hidden
- Shared task is described in Zampieri et. al (2019b)
  - Best models used BERT, but not newer LMs
  - Lot of previous work to check out
- SVMs do quite well on the task, still

There is also a follow-up shared task, described in Zampieri et. al (2020)

Data

Tweet	Label
@USER Buy more icecream!!!	

Data

Tweet	Label
@USER Buy more icecream!!!	NOT

Data

Tweet	Label
@USER Buy more icecream!!!	NOT
Canada doesn’t need another CUCK! We already have enough #LooneyLeft #Liberals f**king up our great country! #Qproofs #TrudeauMustGo	

Data

Tweet	Label
@USER Buy more icecream!!!	**NOT**
Canada doesn’t need another CUCK! We already have enough #LooneyLeft #Liberals f**king up our great country! #Qproofs #TrudeauMustGo	**OFF**

Data

Tweet	Label
@USER Buy more icecream!!!	NOT
Canada doesn’t need another CUCK! We already have enough #LooneyLeft #Liberals f**king up our great country! #Qproofs #TrudeauMustGo	OFF
Democrats soft on crime soft on border security. Since we can’t use Illegal Aliens. I will now call them CRIMINAL Aliens	

Data

Tweet	Label
@USER Buy more icecream!!!	**NOT**
Canada doesn’t need another CUCK! We already have enough #LooneyLeft #Liberals f**king up our great country! #Qproofs #TrudeauMustGo	**OFF**
Democrats soft on crime soft on border security. Since we can’t use Illegal Aliens. I will now call them CRIMINAL Aliens	**NOT**

Data

Tweet	Label
@USER Buy more icecream!!!	NOT
Canada doesn’t need another CUCK! We already have enough #LooneyLeft #Liberals f**king up our great country! #Qproofs #TrudeauMustGo	OFF
Democrats soft on crime soft on border security. Since we can’t use Illegal Aliens. I will now call them CRIMINAL Aliens	NOT
@USER @USER Oh noes! Tough shit.	

Data

Tweet	Label
@USER Buy more icecream!!!	NOT
Canada doesn’t need another CUCK! We already have enough #LooneyLeft #Liberals f**king up our great country! #Qproofs #TrudeauMustGo	OFF
Democrats soft on crime soft on border security. Since we can’t use Illegal Aliens. I will now call them CRIMINAL Aliens	NOT
@USER @USER Oh noes! Tough shit.	OFF

What you have to do

Models

You have to implement **at least** these three models for your classification task:

- Baseline classic model using n-grams (e.g. Naive Bayes, SVM)
- An optimized LSTM model with static embeddings (GloVe, FastText)
  - Embeddings from Brightspace do not work here, download them yourself!
  - Tune settings and architecture (layers, nodes, dropout, optimizer, learning rate, etc)
- A few fine-tuned pretrained language models (BERT, RoBERTa, DeBERTa)

Luckily for you, you already did this in the assignments!

And of course, you are free to use any other model you want!

Research Question

Research question

Some ideas for a creative and original research question. You are still free to come up with your own one, of course.

- Looking into different preprocessing methods of the text, especially in relation the text being tweets (hashtags, emojis)
- Looking at the performance of mono-lingual language models of non-English data. Does this work at all? Does this tell us something about the languages?
- Extensive evaluation of additional features using an SVM. What type of features do you expect to help in this task? Did it work as expected?

Research question

- Using lists of offensive words as features. How well does a baseline model of only these features do? Are there obvious offensive words missing?
- Filtering offensive words from tweets to make the task harder. But does the task then still make sense?
- Similar as above, but filtering the best X features according to an SVM model
- Similar as above, but changing offensive words to a single OFFENSIVE token. Does the content of the offensive words matter at all?

Research question

- Using extra training data from related NLP tasks (e.g. hate speech detection). Does this work at all?
- Using automatically labelled data from OffenseEval 2020. Does this help? How much is needed? Can we use data from different languages?
- Checking how LMs deal with (artificial) noise inserted in the text. Can they still do the task?
- … be creative!

Report

- No length requirements per se, but probably between 6 and 12 pages
- Structure it like an academic paper: read other papers to see what they do
- There are many papers on even just this data set, so I expect a better motivation and description of previous work than before

Introduction:

- Describe problem
- **Motivate** why we want to look at or solve this problem
- Describe what other people did (could be separate section)
- Clearly lay out research question(s)
- Briefly summarize your methods and result

Report

Related work

- Two types of related work:
  - Related to the task
  - Related to your own research question
- How does your work relate to their work?
- How does your work follow their work?
- How is your work different?

The related work section should only fit in **your** paper. If you can copy/paste it in any other paper of this shared task, it is not a good section.

Report

Method

- What data set did you use? What are its characteristics? What language is it in?
- What is your train, dev and test split?
- How will you evaluate the results?
- What models did you use? With what exact settings? **How do they work?**
- What features did you use? What did you try? How did you get them?
- **Reproducibility:** people should be successful in getting the same results
- Sometimes extra information can go in an Appendix

Remember to cite: models, software packages, data sets, ML techniques

Report

Results:

- What are your main results?
  - Add a table or graph with the numbers (usually table is better)
- Include a description of the results in text, do not just rely on the table
- Only have test set scores for a few final models
- Always include dev scores for each test result
- How do your scores compare to the shared task results?
- Some interpretation: were the results as expected?

Report

Discussion

- What are the implications of the findings? Why is that interesting?
- How do your results relate to previous work? What does it add?
- Or more generally: what did we learn?
- Error analysis: manually annotate and show a few interesting examples
- Extra analysis: maybe performance on a particular subset is interesting
- Feature analysis: always nice to look at the best SVM features
- Given these results, what would be the next steps for future research?

Grading sheet

You will get a grade for these 10 criteria, which are averaged as final grade:

Writing (2): structure, coherence & language
Introduction (2): motivation & previous work, research questions
Method (3): description of models, reproducibility, models
Results (3): overview, performance, discussion & analysis

Make sure to check out the more detailed version in the assignment file.

Labs

October 15th:

- Start working on the project, get familiar with the data, implement baselines, read previous work, pose research question, get feedback

October 22nd:

- Discuss progress and research questions

October 29th:

- Final week, Rik/Leo not there physically, but room available
- Possible to discuss progress per mail with Rik

Deadlines

Peer review 2: Monday October 20th 23:59

Project: Monday November 3rd 23:59