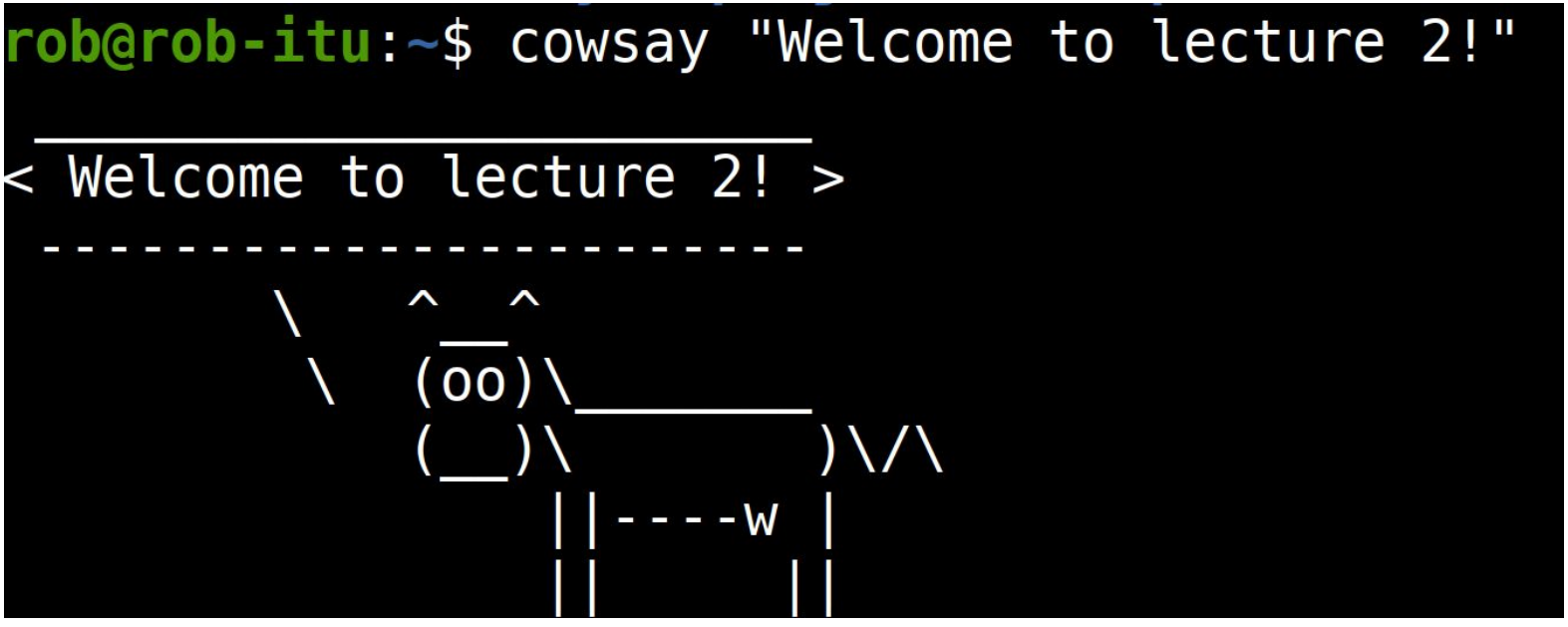

rob@rob-itu:~$ cowsay "Welcome to lecture 2!"
< Welcome to lecture 2! >
^__^
(oo)_____
(__)\ )\/\
||----w |
|| ||

Until now:

- Introduction to a variety of NLP tasks
- Hands on experience with tokenization

Command line and experimental setup

Today:

- Linux command line
- Data splits
- Baselines
- Metrics
- Reporting
- Reproducibility
- Statistical significance
- Analysis

After today, you should be able to:

- Perform basic text search and replacement operations on the linux command line
- Reflect on experimental setups in NLP
- Identify problems in experimental setups
- Design and implement a solid NLP experiment

After today, you should be able to:

- Perform basic text search and replacement operations on the linux command line
- Reflect on experimental setups in NLP
- Identify problems in experimental setups
- Design and implement a solid NLP experiment

Larger variety of topics: no menti, but ask questions anytime!

Recap square brackets

How to read the square brackets in a regex:

- [Tt]he: One instance of a T or t
- [Tt+]he: One instance of a T, t or +!
- [Tt]+he: One or more instances of T or t: TTTThe

This course

- We will use common GNU applications
- HPC runs on Ubuntu
- Mac OS has all tools we use
- Windows subsystem for linux?

Why command line?:

GUI

- Graphical representations
- Easy to do easy things
- Most options/possibilties directly visible
- Complex applications

Commandline

- Text-based
- Can be hard to do easy things (i.e. selecting multiple files)
- Powerful for file-processing (especially text-data)
- Most applications have one goal (Unix philosophy)
- Applications can be combined easily
- Easier to document what was done exactly (reproducibility)
- Used on HPC's

Command line standard syntax

```
command -c value input
```

Command line standard syntax

Example:

```
cowsay -e "^^" "hi there"
 ----------
< hi there >
 ----------
        \   ^__^
         \  (^^)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Basic commands:

- cat
- head/tail
- wc
- less
- pipe/redirect

cat: print contents of file to stdout:

```
$ cat twitter-robvanderg.txt
@user @user The only random factor is the seed used to trai
@user @user Is a lower performance really an indication tha
Man is to doctor as woman is to:... doctor!, see our paper
@user @user It does not always fail, and it definitely does
@user @user Yes, the bias is definitely present. But it is
@user @user We cited two of her earlier papers, which seeme
For any master students expecting to graduate this year, pl
The overview paper is here:  https://www.website.com , and
Ok, fair enough, a large transformer model still ranked hig
@user @user @user @user @user @user @user @user Thanks for
@user @user @user @user @user @user @user @user Interesting
MaChAmp is implemented in @user (and @user Many thanks to t
@user @user Thanks to you as well!
@user Out of curiousity, how would you go about ‘tokenizing
@user @user @user @user No worries, I completely understand
@user @user @user @user Congrats!, interesting comparison,
@user @user @user on the website FAQ it still says 4-8 thou
@user @user @user @user Thanks for the link. We missed that
```

head/tail: print beginning or end of file

```
$ head -n 2 twitter-robvanderg.txt
@user @user The only random factor is the seed used to trai
@user @user Is a lower performance really an indication tha

$ tail -n 2 twitter-robvanderg.txt
@user I agree, if you use a (non-neural) model without mode
Also bored by shared tasks where the largest finetuned tran
```

‘wc‘: wordcount: but does also character/line count

‘wc‘: wordcount: but does also character/line count

```
$ wc twitter-robvanderg.txt
  29  591 3759 twitter-robvanderg.txt
```

Can also get one at a time:

```
wc -c twitter-robvanderg.txt
wc -w twitter-robvanderg.txt
wc -l twitter-robvanderg.txt
```

less: read contents of file interactively

- space/enter/arrowkeys/PgDn/PgUp for scrolling
- / for searching
- q for quit!

Input and output:

- ‘stdin‘: input to a command (i.e. through pipe)
- ‘stdout‘: output of a command (can be passed through pipe)
- ‘stderr‘: alternative output containing error messages

Combine commands

- Unix philosophy: many simple applications that do one thing, and do it well
- They can be chained together with a pipeline: |
- Output can be written to file with >
- Stderr can be written with 2 >

Combine commands

- Unix philosophy: many simple applications that do one thing, and do it well
- They can be chained together with a pipeline: |
- Output can be written to file with >
- Stderr can be written with 2 >

```
$ cat robv.txt chrha.txt | wc -l > twitter_count.txt
```

Combine commands

- Unix philosophy: many simple applications that do one thing, and do it well
- They can be chained together with a pipeline: |
- Output can be written to file with >
- Stderr can be written with 2 >

```
$ cat robv.txt chrha.txt | wc -l > twitter_count.txt
```

```
$ cat teacher_twitter_counts.txt
29
```

Text processing

- grep
- sort
- uniq
- cut/paste
- tr
- sed

‘grep‘ options:

- ‘-i‘: ignore capitalization
- ‘-o‘: match only regexp
- ‘-n‘: show line numbers
- ‘-c‘: count lines that match
- ‘-v‘: invert

Note that there are multiple ways to do the same thing:

```
grep "[a-z]*ing" twitter-robvanderg.txt -c ==
grep [a-z]*ing" twitter-robvanderg.txt  | wc
```

‘grep‘ options:

- ‘-i‘: ignore capitalization
- ‘-o‘: match only regexp
- ‘-n‘: show line numbers
- ‘-c‘: count lines that match
- ‘-v‘: invert

Note that there are multiple ways to do the same thing:

```
grep "[a-z]*ing" twitter-robvanderg.txt -c ==
grep [a-z]*ing" twitter-robvanderg.txt  | wc
```

Note that there are different interpretations of regular expressions!, for extended suport use `-E` This adds support for example for the +.

‘sort‘

- Sorts the content of a file
- ‘-n‘ numeric
- ‘-r‘ reverse

‘sort‘

- Sorts the content of a file
- ‘-n‘ numeric
- ‘-r‘ reverse

get sorted list of “words” in document:

```
grep -Eo " [a-z]+ " twitter-robvanderg.txt | sort | head
```

‘uniq‘

- removes duplicate lines (only adjacent!)
- ‘-c‘ count how many repetitions

‘uniq‘

- removes duplicate lines (only adjacent!)
- ‘-c‘ count how many repetitions

get sorted list of “words” in document:

```
!grep -Eo " [a-z]+ " twitter-robvanderg.txt | sort | uniq
```

‘cut‘

- split columns from file
- ‘-d‘: delimeter (default=tab)
- ‘-f‘: select fields

‘paste‘

- paste columns together from multiple files

cut/paste

Why?

```
# sent_id = dev-0
# text = Hvor kommer julemanden fra?
1       Hvor    hvor    ADV     _       _       2       advmod  _       _
2       kommer  komme   VERB    _       Mood=Ind|Tense=Pres|VerbForm=Fin|Voice=Act      0       root    _       _
3       julemanden      julemand        NOUN    _       Definite=Def|Gender=Com|Number=Sing     2       nsubj   _
4       fra     fra     ADP     _       AdpType=Prep    1       case    _       SpaceAfter=No
5       ?       ?       PUNCT   _       _       2       punct   _       _
```

‘tr‘

- Replace characters
- Does not take path as argument, only reads from stdin!

```
cat text.txt | tr " " "\n"
tr " " "\n" < text.txt
```

‘tr‘

- Replace characters
- Does not take path as argument, only reads from stdin!

```
cat text.txt | tr " " "\n"
tr " " "\n" < text.txt
```

naive token-list

```
cat twitter-robvanderg.txt | tr " " "\n" | sort | uniq
```

‘sed‘

- Filtering and transforming text
- We mostly use for replacing
- ‘-i‘ for in-place (dangerous!)
- Usage:

```
sed "s;<FIND>;<REPLACE>;g"
```

OR

```
sed "s/<FIND>/<REPLACE>/g"
```

‘sed‘

- Filtering and transforming text
- We mostly use for replacing
- ‘-i‘ for in-place (dangerous!)
- Usage:

```
sed "s;<FIND>;<REPLACE>;g"
```

OR

```
sed "s/<FIND>/<REPLACE>/g"
```

```
$ sed -i "s;significant;substantial;g" *tex
```

Differences between tr and sed:

- sed supports more functionality
- But tr can be used across lines!

Timesavers:

- Tab-completion:

`ls /home/rob/Doc<TAB>`

becomes:

`ls /home/rob/Documents`

- arrow-up/down to scroll through history of commands
- '<ctrl>-r' can be used to search through history of commands

What if you forget how a command works?

- man pages
- Ask your peers
- Google, Stackoverflow, etc.

Why useful in NLP?

We work with text!

- Quick conversion between formats
- Inspection of data
- Analysis of errors

What can't we do with the command line?:
`telnettowel.blinkenlights.nl`

Data splits

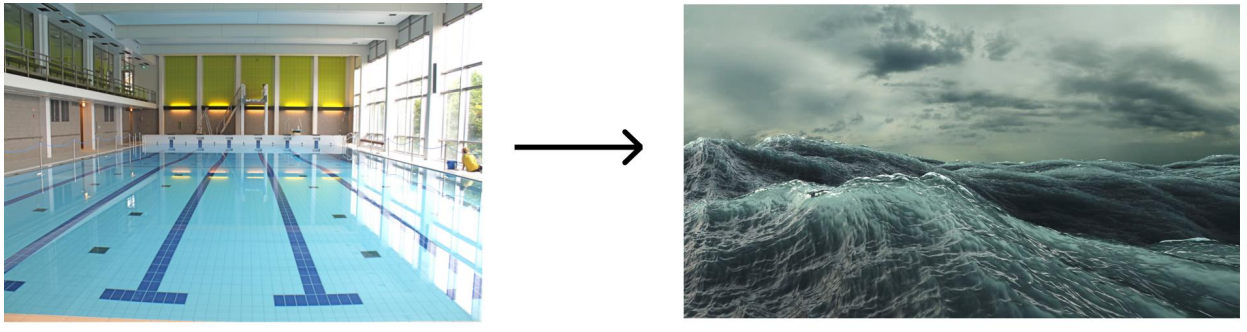
Basic setup during lecture phase: train->dev

- To test whether new feature is beneficial, train with it and see if scores on dev data increases
- However, unsure how this improvement will transfer (did we overfit?)

Data splits

Basic setup during lecture phase: train->dev

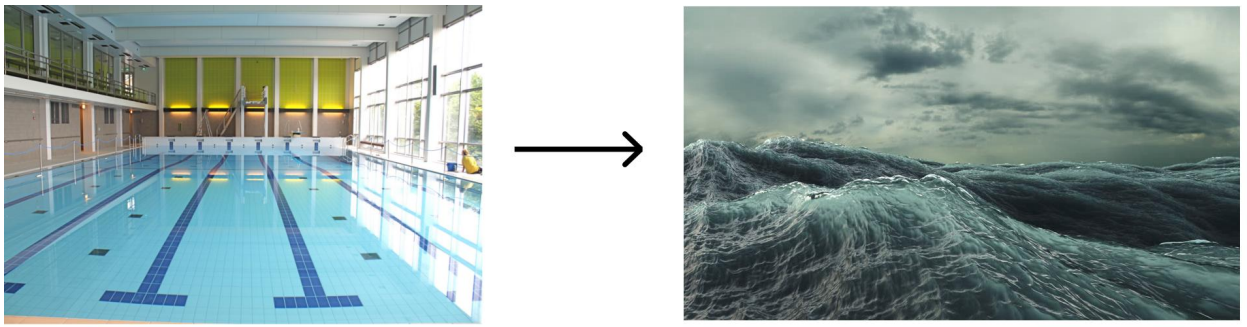
- To test whether new feature is beneficial, train with it and see if scores on dev data increases
- However, unsure how this improvement will transfer (did we overfit?)



Data splits

Basic setup during lecture phase: train->dev

- To test whether new feature is beneficial, train with it and see if scores on dev data increases
- However, unsure how this improvement will transfer (did we overfit?)

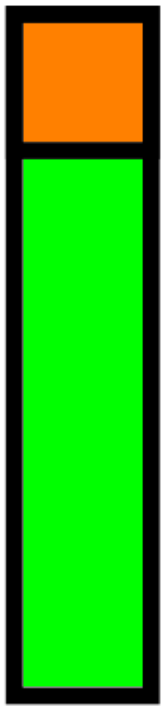


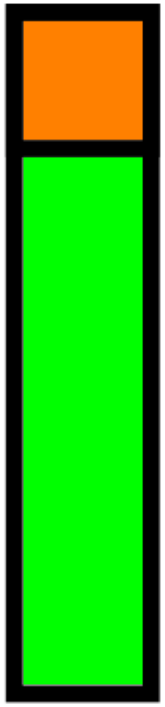
More commonly known as the problem of *domain adaptation*, however we should take this into account in all cases! Note that for a long time, the Wall Street Journal part of the PennTreebank was one of the main benchmarks in NLP

Instead of using random parts of your dataset for splits, it is good practice to split your data based on some criteria like:

- time
- document
- domain
- writer/speaker
- ...

This is called a stratified split and ensures that your estimated performance is realistic

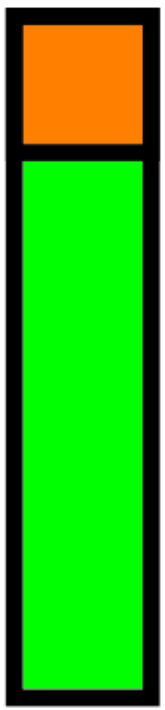




- Two types of **overfitting**:
  - overfitting of the optimization function on the training data
  - overfitting of the design decisions on the evaluation data

- overfitting of the design decisions on the evaluation data

Can be solved by using a separate development set:

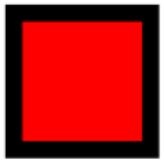


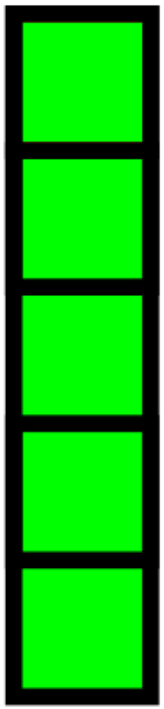
- train: Used for training models, in some setups this split can be omitted (zero-shot or unsupervised learning)
- development (also called validation/evaluation): Used to compare all different versions of the proposed model(s). Can also be used to get preliminary answers to the main research questions
- test: Used to confirm the final answer to the research question

- train: Used for training models, in some setups this split can be omitted (zero-shot or unsupervised learning)
- development (also called validation/evaluation): Used to compare all different versions of the proposed model(s). Can also be used to get preliminary answers to the main research questions
- test: Used to confirm the final answer to the research question
- In the assignments, we will sometimes only use dev-data, as we do not compare too many models

To be even more safe:

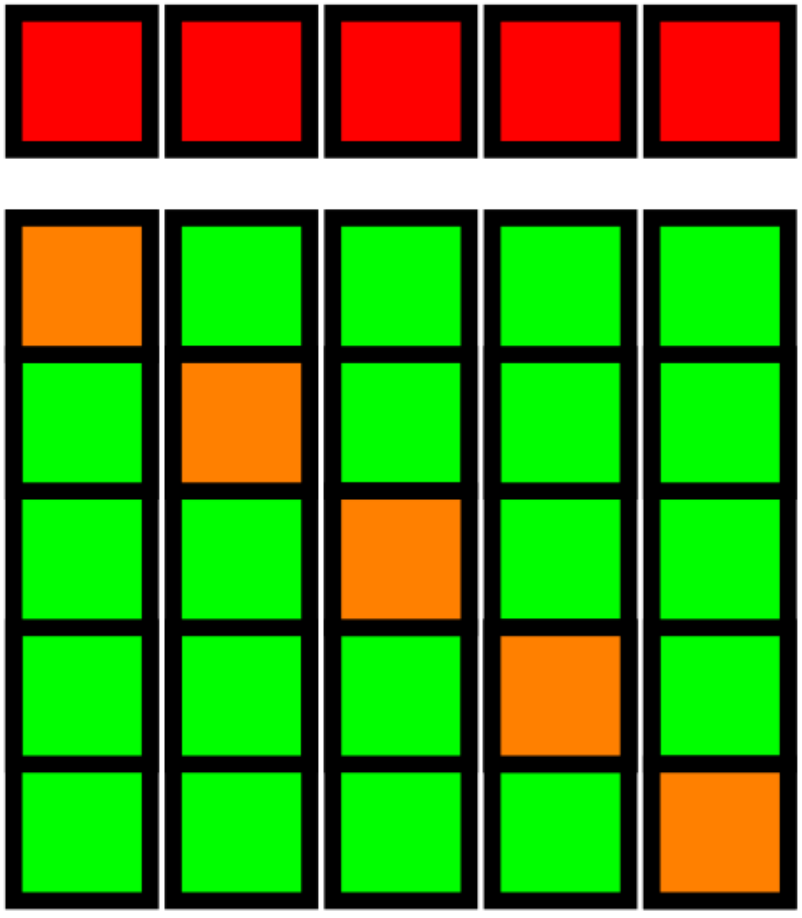
- k-fold:





To be even more safe:

- k-fold:



We need to talk about standard splits (2019)

- https://aclanthology.org/P19-1267/
- Suggests to use random splits instead of standard splits

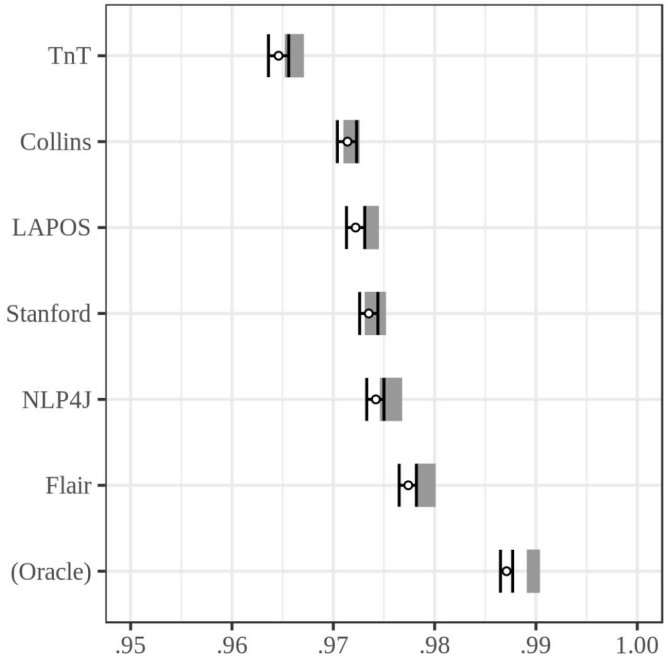


Figure 1: A visualization of Penn Treebank token accuracies in the two experiments. The whiskers shows accuracy and 95% confidence intervals in experiment 1, and shaded region represents the range of accuracies in experiment 2.

- What is the potential problem?

We need to talk about random splits (2021)

- https://aclanthology.org/2021.eacl-main.156/
- Suggest to automatically find stratified splits

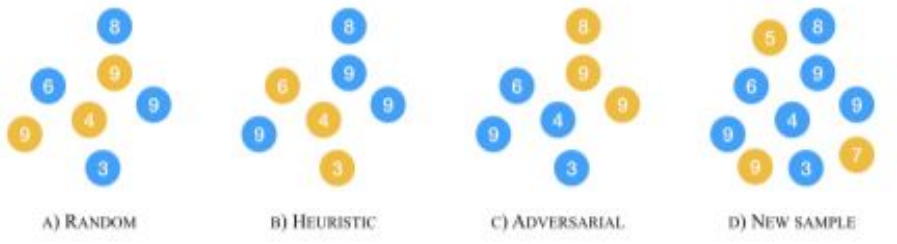


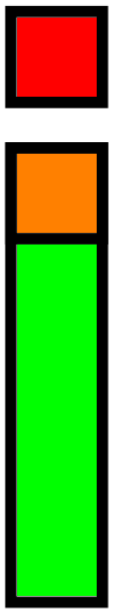
Figure 1: Data splitting strategies. Each ball corresponds to a sentence represented in (two-dimensional) feature space. Blue (dark)/orange (bright) balls represent examples for training/test. Numbers represent sentence length. Heuristic splits can, e.g., be based on sentence length; adversarial splits maximize divergence.

- Results show that its better than standard and random, but still distant from new-samples

We need to talk about train-dev-test splits (2021):

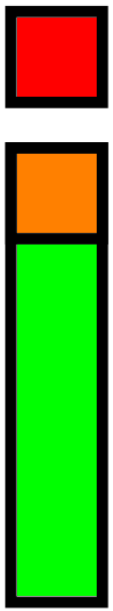
We need to talk about train-dev-test splits (2021):

- `https://aclanthology.org/2021.emnlp-main.368.pdf`
- Neural network models often rely more on development data: they train N versions of the model, and pick the best based on development set performance
- It has become more common to report a lot of scores on test-data recently

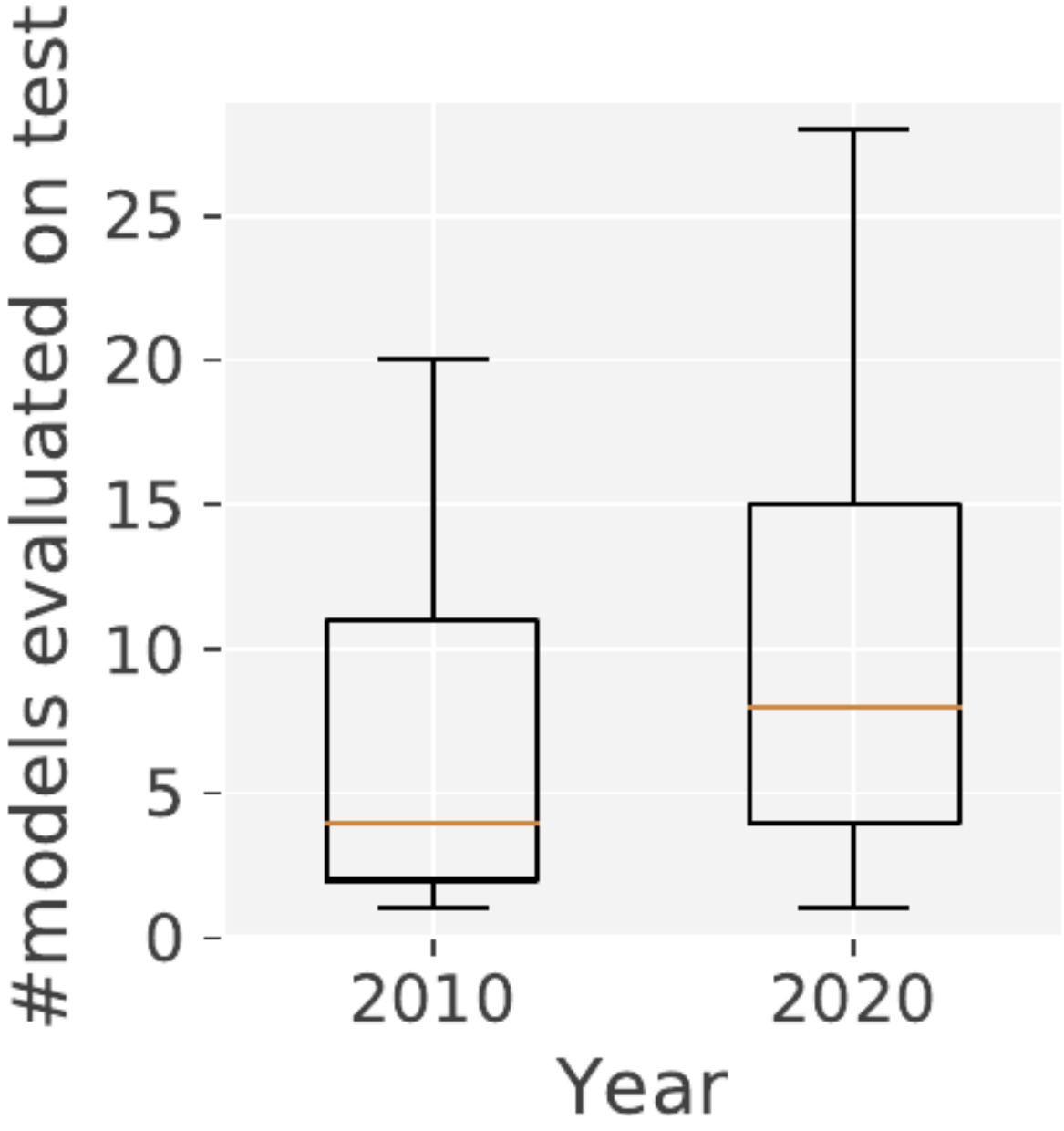


We need to talk about train-dev-test splits (2021):

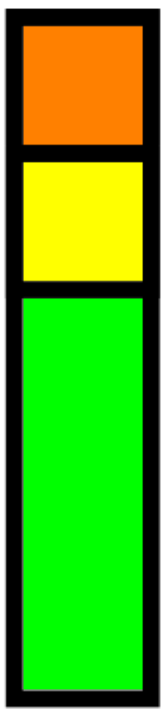
- `https://aclanthology.org/2021.emnlp-main.368.pdf`
- Neural network models often rely more on development data: they train N versions of the model, and pick the best based on development set performance
- It has become more common to report a lot of scores on test-data recently



Is that really the case?


#models evaluated on test
0
5
10
15
20
25
2010
2020
Year

- Tune split: a split to use for model selection: This will avoid overfitting and/or over-reporting on the secret test data.



However:

Home

Papers

CV

Datasets

Demos

Blog

Thesis Projects

Reflections on the tune split

It is standard practice in natural language processing to split our datasets into three parts: train, dev, and test. The train split can be used to train the model, and the development set is used to evaluate the model during the development phase. Finally, our most promising/interesting models can be evaluated against each other on the test set. In practice, for a new addition/improvement to a model, this would mean that one would train multiple versions of this model (for example for hyperparameter tuning), and evaluate these on the dev data, and the best (or most interesting) model is then compared to the baseline and previous work (i.e. state-of-the-art).

Since the introduction of neural networks, the usage of these standard splits has shifted a bit. Since most approaches make use of model selection and early stopping, the dev set is already used in the training procedure. People have noticed that the internal models can now not fairly be compared on the dev split anymore, and have started to use the test data for this. To quantify this issue, I have counted for 100 random papers from ACL 2010 and 2020 how many runs on each test set where made in each paper. The results confirm a clear trend; in 2020, many more test-runs were made per paper.

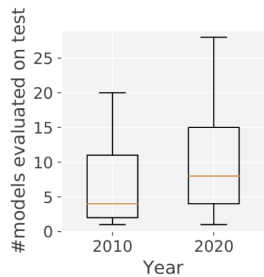


Figure 2: A boxplot visualizing the median and quantiles of the number of models evaluated on test data for a selection of 100 random papers from the ACL 2010 and ACL 2020 proceedings.

The solution I proposed in my 2021 EMNLP paper, was to use a tune split, which we can use to tune the model (i.e. early stopping and/or model selection). However, after having used the tune set for multiple experiments, I became frustrated by the additional complications in the setup that were caused by needing an additional split. First, one needs to have plenty of data, otherwise

Baselines

Note that the word "baseline" can have different meanings

- The simplest possible approach (majority baseline, i.e. everything is positive or noun)
- A simple machine learning classifier (logistic regression with words as features)
- The state-of-the-art approach on which you want to improve

Baselines

Note that the word "baseline" can have different meanings

- The simplest possible approach (majority baseline, i.e. everything is positive or noun)
- A simple machine learning classifier (logistic regression with words as features)
- The state-of-the-art approach on which you want to improve
- Which one to use?

- Which one to use?

depends on your research question! In most cases, we actually use a competetive baseline, as many research questions are like:

- can we use X to improve on Y

Is it interesting to ask this question for the majority baseline?

- Which one to use?

depends on your research question! In most cases, we actually use a competetive baseline, as many research questions are like:

- can we use X to improve on Y

Is it interesting to ask this question for the majority baseline?

- The better the baseline, the more relevant the answer!
- For your final project, we do not require a state-of-the-art baseline, but expect a somewhat competetive baseline if your plan is to improve it.

Is a majority baseline useless?

- No!, if my proposed system scores 80% accuracy on sentiment analysis, is this any good?

Metrics

Which metric to use depends on task **and** dataset

- Standard approach: use same as previous work for easy/fair comparison

Metrics

Which metric to use depends on task **and** dataset

- Standard approach: use same as previous work for easy/fair comparison
- Dutch saying: If I jump in a ditch, would you also do it?

- So: if their metrics makes no sense, or is easy to cheat on, report also an alternative!
- Always ask: does your metric reflect the goal of the system? (is higher always better?)

Common metrics in NLP:

- Accuracy: number of correct instances/all instances
- Recall: number of instances you have found/total number of instances (for a class)
- Precision: the percentage of instances you have found that are correct (for a class)
- F1 score: the harmonic mean between recall and precision
- For some tasks (e.g. machine translation), evaluation is an active research area!
- Span-F1: for span-based tasks (like NER, more next week)

When to use accuracy:

- when number of elements to find is known
- when labels are approximately balanced?

Imagine the following confusion matrix for language classification:

		gold	
		EN	DA
pred	EN	1970	40
	DA	10	30

Imagine the following confusion matrix for language classification:

		gold	
		EN	DA
pred	EN	1970	40
	DA	10	30

We're going to map these scores to:

	positive	negative
true		
false		

		gold	
		EN	DA
pred	EN	1970	40
	DA	10	30

For class DA:

	positive	negative
true	30	1970
false	10	40

Precision: How many of the instances found are correct?

$$Precision = \frac{tp}{tp + fp} \tag{1}$$

Recall: How many of the gold instances in class X did I find?

$$recall = \frac{tp}{tp + fn} \tag{2}$$

Note that *tn* is not in the formula's, but is intrinsically still taken into account

F1 score: Harmonic mean between precision and recall:

$$2 * \frac{precision * recall}{precision + recall} \tag{3}$$

- you are rewarded for higher similarity between precision and recall!:
- e.g. .5 recall .5 precision is much higher then .9 recall and .2 precision

Note that recall, precision and f1 score are calculated per class!

- Commonly macro-averaged F1-score is reported (for n classes):

$$\frac{\sum_1^n F1(preds_n)}{n} \tag{4}$$

Back to our example:

		gold	
		EN	DA
pred	EN	1970	40
	DA	10	30

- Accuracy $= 2000/2050 = 0.98$
- recall_da $= 30/70 = 0.43$
- precision_da $= 30/40 = 0.75$
- f1_da $= 0.55$
- recall_en $= 1970/1980 = 0.99$
- precision_en $= 1970/2010 = 0.98$
- f1_en $= 0.99$
- macro_f1 $= (0.99+0.55)/2 = 0.77$

Back to our example:

		gold	
		EN	DA
pred	EN	1970	40
	DA	10	30

- Accuracy $= 2000/2050 = 0.98$
- recall_da $= 30/70 = 0.43$
- precision_da $= 30/40 = 0.75$
- f1_da $= 0.55$
- recall_en $= 1970/1980 = 0.99$
- precision_en $= 1970/2010 = 0.98$
- f1_en $= 0.99$
- macro_f1 $= (0.99+0.55)/2 = 0.77$
- Which one to use?

Note that:

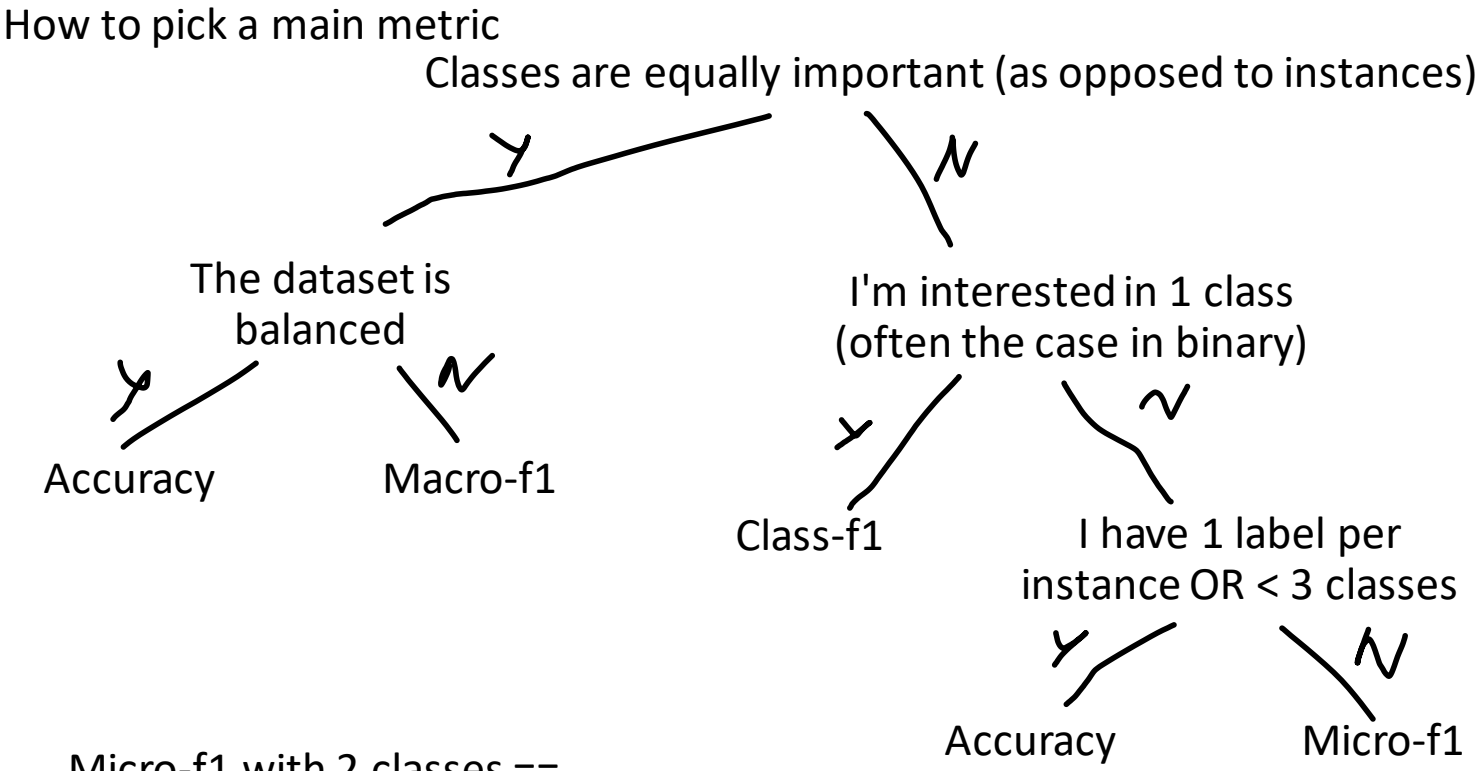
- There is also a micro f1 score, this counts all TP/FP/TN/FN, and calculates precision and recall over all classes: this is hard to interpret, and not commonly used
- Sometimes precision or recall might be more important, then they can be tuned using the more general f-score:

$$F_\beta = (1 + \beta^2) * \frac{\textit{precision} * \textit{recall}}{(\beta^2 * \textit{precision}) + \textit{recall}} \tag{5}$$

$\beta > 1 ==$ recall $>$ precision

Robs chart to metrics in NLP

Robs chart to metrics in NLP



Micro-f1 with 2 classes == accuracy; but I prefer to use the simpler one for naming it

Weighted f1 is not in here as Rob thinks it is not a good fit for main metric

If Class-f1 is used, be clear about this (which class)!

Reporting

- Q: How much detail do I have to report?
- A: As much as is relevant for your research question
- pause
- NOT: We save sentences in a list of python lists etc. etc.
- But arent hyperparameters important for reproducability? (appendix)
- See also for example: `https://2021.emnlp.org/call-for-papers#reproducibility-criteria`

More on this: 15-03

Recent updates in the field:

- dataset statement
- model cards
- ethics statements
- responsible researcher form

More on this: 15-03

Reproducibility and Replicability

Definitions we follow in this course:

- reproducibility: reproduce conclusion in a *similar* setup
- replicability: reproduce exact results with access to same code/data+annotations (narrower than reproducability)

This definition follows (Fokkens et al., 2013), we however strictly assume access to the code for **replicability**
NOTE, definitions are not agreed upon!: Liberman (2015), Cohen et al. (2018).

When having acces to code and data, isn't it easy to get the same results?

When having acces to code and data, isn't it easy to get the same results? **No!**

When having acces to code and data, isn't it easy to get the same results? **No!**

- Difference in package versions (scikit-learn, pytorch etc.)
- Differences in language settings in linux
- Random seeds
- Different CPU/GPU give different results
- How to run/use someone else his/her code

Simple approach to solve the last point:

- Separate source code and scripts to run experiments
- Number/name each experiment, for naming the scripts
- Create a runAll.sh and genAll.sh script in the scripts folder, which contain each exact step
- Save at least predictions (preferably also models)

Simple approach to solve the last point:

- Separate source code and scripts to run experiments
- Number/name each experiment, for naming the scripts
- Create a runAll.sh and genAll.sh script in the scripts folder, which contain each exact step
- Save at least predictions (preferably also models)

More info: `https://robvanderg.github.io/blog/repro.htm`

This is just one approach, not perfect, and not universally used. But it works well in practice!

- It is not required to use this for the final project.
- This specific approach is **not** exam material
- Note that reproducability is not only important for other people, experiments often have to be re-ran!

Other approaches:

- Use containers (Docker)
- Don't release code, or just an empty repository: only 36.2% of papers in our main conference shared code: Wieling et al. (2018) (`http://martijnwieling.nl/files/squib.pdf`)!

Other methods to make results more stable:

- Report average over X random seeds
- Also look at the standard deviation, high deviations mean unstable results

Analysis

- What distinguishes an excellent paper from a good paper!
- Crucial in NLP because metrics never tell the whole story
- Where does my proposed system do well, and where not?

Statistical significance: Paired bootstrap test

- No assumptions about distribution
- Compares differences of 2 situations on same samples: before-after (or 2 NLP models)
- Assumes that sample is representative of full population
- Generates many variations of existing results, to test how surprising it is that A outperforms B

$$\delta(x) = M(A, x) - M(B, x)$$
$$H_0 : \delta(x) \leq 0$$
$$H_1 : \delta(x) > 0$$

function BOOTSTRAP(test set x, num of samples b) **returns** p-*value*(x)

Calculate $\delta(x)$ # how much better does algorithm A do than B on x
$s = 0$
for $i =$ 1 **to** b **do**
    for $j = 1$ **to** n **do** # Draw a bootstrap sample $x^{(i)}$ of size n
        Select a member of x at random and add it to $x^{(i)}$
    Calculate $\delta(x^{(i)})$ # how much better does algorithm A do than B on $x^{(i)}$
    $s \leftarrow s + 1$ **if** $\delta(x^{(i)}) > 2\delta(x)$
p-value(x) $\approx \frac{s}{b}$ # on what % of the b samples did algorithm A beat expectations?
return p-value(x) # if very few did, our observed δ is probably not accidental

Figure 4.9 A version of the paired bootstrap algorithm after Berg-Kirkpatrick et al. (2012).

Why: $\delta(x^{(i)}) > 2\delta(x)$

- The mean of the distribution is not 0, but is $\delta(x)$, and we are modeling the surprise!

$\delta(x^{(i)}) - \delta(x) > \delta(x)$

ASO test

Almost Stochastic Order

- proposed in: `https://aclanthology.org/P19-1266.pdf`
- to compare multiple versions of the same models
- why: neural language models are dependent on random factors!
- implementation from Dennis Ulmer: `https://github.com/Kaleidophon/deep-significance`
- not part of the exam (not in reading material), but might be useful for your project

Analysis

- Qualitative analysis: Looking at the actual data (eyeballing)
- Quantitative analysis: Anything that looks at counts
- Often complement each other

Example

Cross-domain sentiment analysis (3 classes): train on Amazon data, evaluate on TikTok data

- Main metric: Macro-F1
- How to do qualitative analysis?

Example

Cross-domain sentiment analysis (3 classes): train on Amazon data, evaluate on TikTok data

- How to do quantitative analysis?

Usefulness of command line

DanTok examples

Questions?