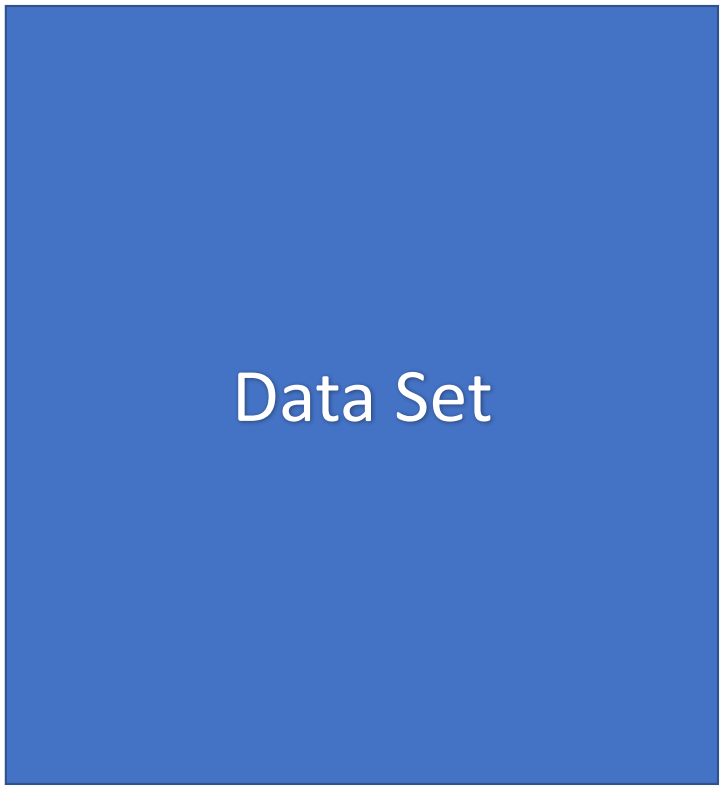
LING 472 Section 5/19

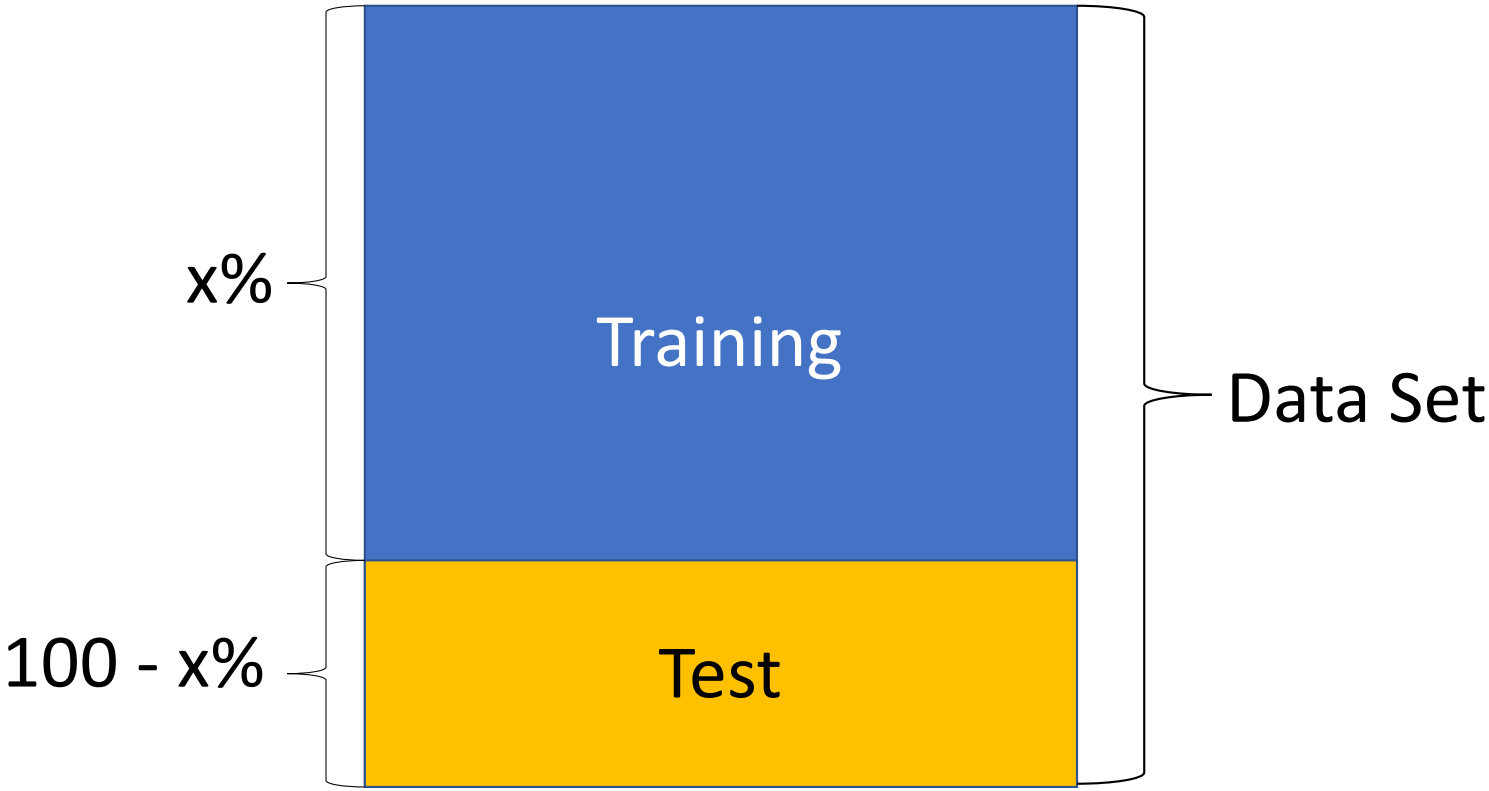
Assignment 4 Parts 2 & 3

- Part 2
  - Restrict the grammar to account for agreement between subject and verb.
  - Restrict the grammar so that transitive and intransitive verbs must go through the appropriate VP rules.
  - Your rules.tdl should have a lot of redundancy in it.
- Part 3
  - The goal is to remove the redundancy by putting common rules into types.tdl
  - There is no restriction on what features you can access in types.tdl – all can be accessed, you can use reentrancy
  - You can think of types.tdl as a list of interfaces, and rules.tdl and lexicon.tdl as a list of classes and instantiations that inherit from those interfaces.
  - When you have a < list > in tdl, it is always of a certain length.

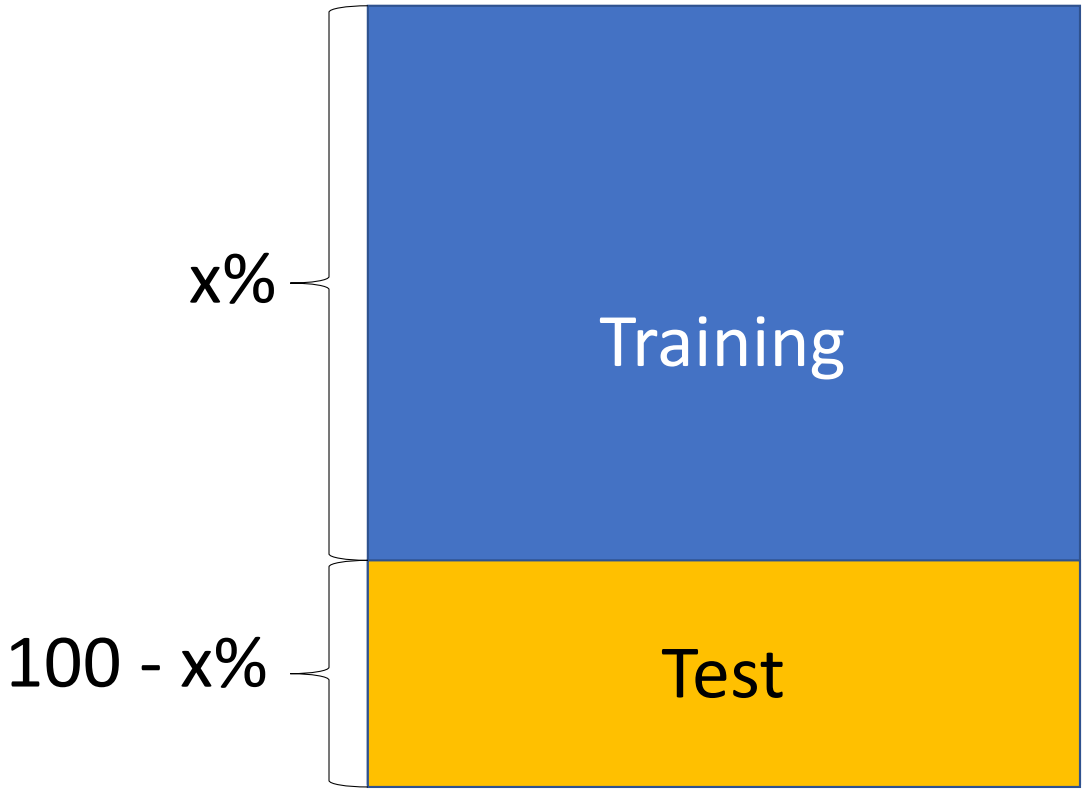
Training, Development, & Test Sets



Training, Development, & Test Sets



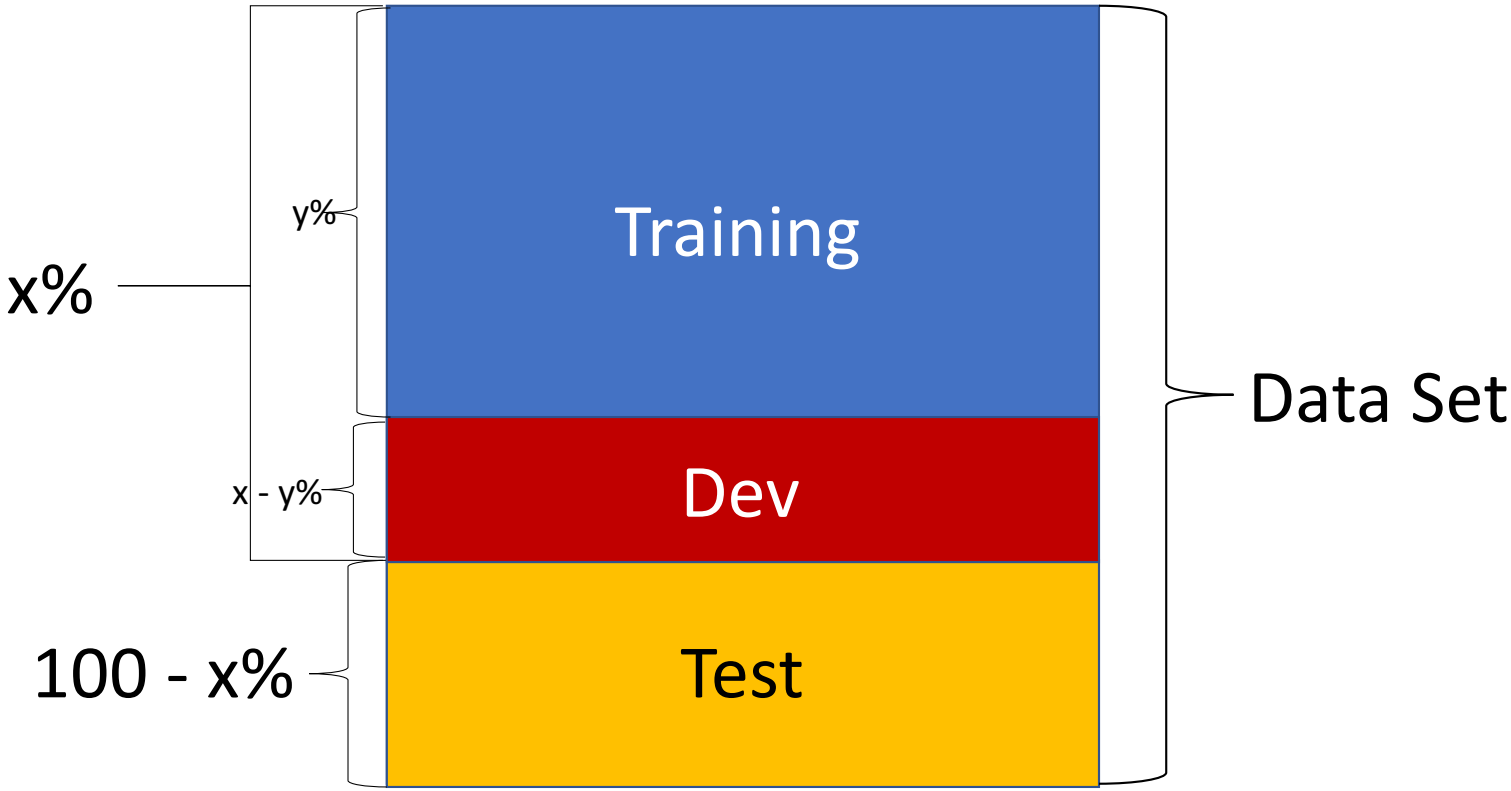
Training, Development, & Test Sets



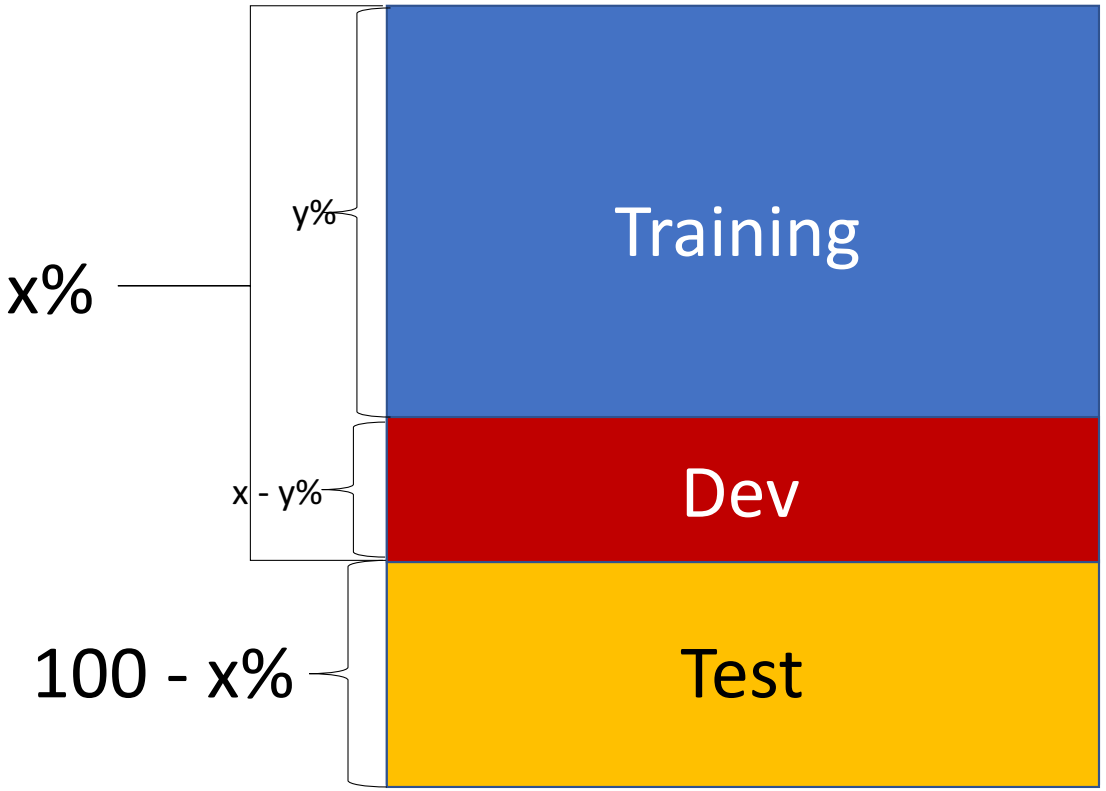
Training: Used to train the model (create probabilities), decide on methods used

Test: Used to test the model, based on what you/the model learned from training. Ideally you test **once**.

Training, Development, & Test Sets



Training, Development, & Test Sets

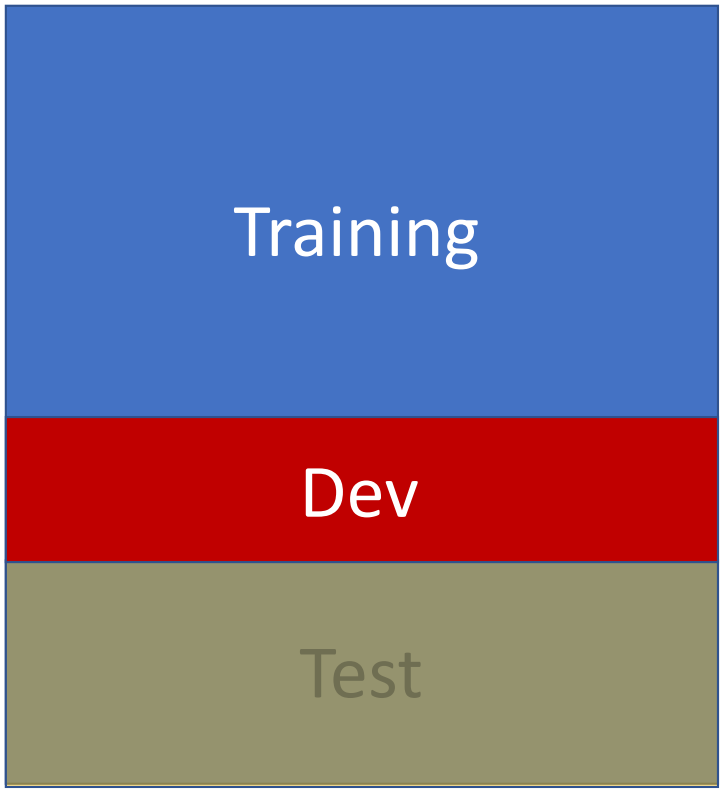


Training: Used to train the model (create probabilities, etc)

Dev: Used to test the model created in training, can refine and run multiple times

Test: Used to test the model, after development is done. Ideally run once.

Training, Development, & Test Sets



During development:

1. Train on the training set.
2. Test on the dev set.
3. Refine your training algorithm.
4. Repeat from 1.

(Test is ignored)

Training, Development, & Test Sets

After development:

1. Train on the training + dev set.
2. Test on the test set.
3. Fix any bugs in running the code, if necessary. (DON'T change the algorithm!)
4. Report on performance.

Training, Development, & Test Sets

- General pointers:
  - Your data sets should all look the same (same formatting, same preprocessing)
  - The ratio for training : dev should be about the same as the ratio of training+dev : test
  - When you're looking at your corpus/corpora to get ideas about how to write your algorithm, restrict yourself to training, or training + dev, don't look at test.

Reporting and Presenting

- You need to present about your:
  - Research question
  - Data set
  - Algorithm (for Part 1)
  - Results (for Part 1)
  - What you plan to do for Part 2

Reporting and Presenting

- Research question
  - What is the question that your project is attempting to answer? Why is it interesting?
- Data set
  - Where did you get your data from?
  - How did you divide into to training, (development,) and test sets?
  - What preprocessing did you do on it, if any?
- Algorithm (Part 1)
  - How are you answering the research question?
  - Be as detailed as necessary without talking about code.

Reporting and Presenting

- Results (Part 1)
  - What was your baseline?
  - What metrics are you reporting?
  - How did your baseline perform on the test set?
  - How did your algorithm perform on the test set?
  - Why do you think your algorithm performed the way it did? What kind of errors did it make?
- Plans for Part 2
  - How are you going to modify your algorithm for Part 2? How will this address shortcomings you discovered in your results from Part 1?