Natural Language Processing
Text Classification and Naïve Bayes

Felipe Bravo-Marquez

June 14, 2023

Is this spam?

Subject: **Important notice!**
From: Stanford University <newsforum@stanford.edu>
Date: October 28, 2011 12:34:16 PM PDT
To: undisclosed-recipients:;

Greats News!

You can now access the latest news by using the link below to login to Stanford University News Forum.

http://www.123contactform.com/contact-form-StanfordNew1-236335.html

Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information
about the new services.

This slides are based on the course material by Daniel Jurafsky :

`https://web.stanford.edu/~jurafsky/slp3/4.pdf`

Who wrote which Federalist papers?

- 1787-8: Anonymous essays attempted to convince New York to ratify the U.S Constitution: Jay, Madison, Hamilton.
- Authorship of 12 of the letters is in dispute.
- 1963: Solved by Mosteller and Wallace using Bayesian methods.

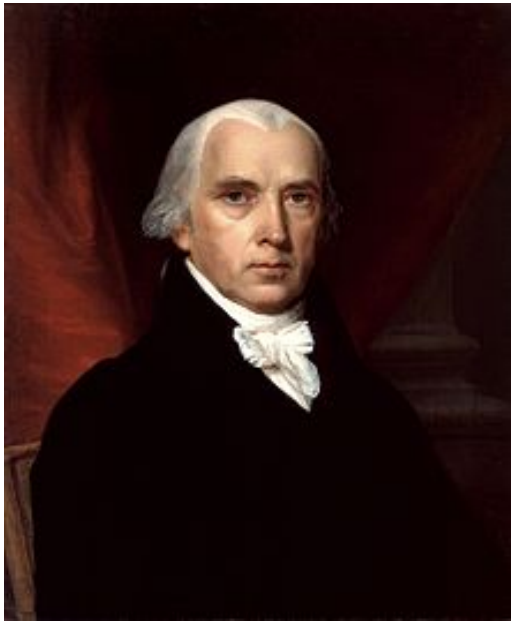


Figure: James Madison

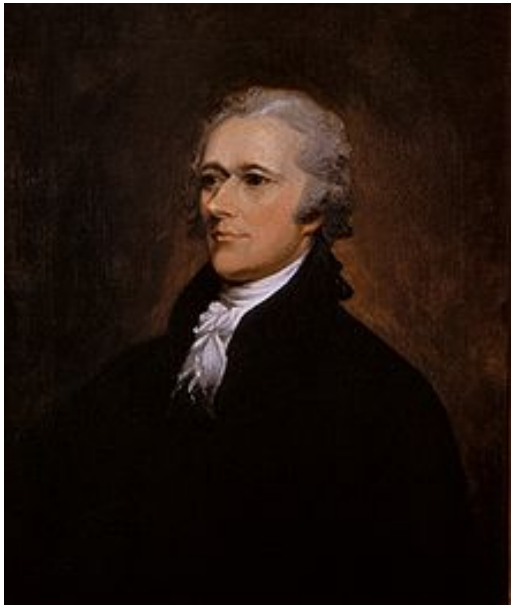


Figure: Alexander Hamilton

What is the subject of this medical article?

MEDLINE Article

?

MeSH Subject Category Hierarchy

- Antogonists and Inhibitors
- Blood Supply
- Chemistry
- Drug Therapy
- Embryology
- Epidemiology
- …

Positive or negative movie review?

- **+** ...zany characters and richly applied satire, and some great plot twists
- **-** It was pathetic. The worst part about it was the boxing scenes...
- **+** ...awesome caramel sauce and sweet toasty almonds. I love this place!
- **-** ...awful pizza and ridiculously overpriced...

Why sentiment analysis?

- Movie: Is this review positive or negative?
- Products: What do people think about the new iPhone?
- Public sentiment: How is consumer confidence?
- Politics: What do people think about this candidate or issue?
- Prediction: Predict election outcomes or market trends from sentiment.

Scherer Typology of Affective States

- Emotion: Brief organically synchronized ... evaluation of a major event
  - Angry, sad, joyful, fearful, ashamed, proud, elated
- Mood: Diffuse non-caused low-intensity long-duration change in subjective feeling
  - Cheerful, gloomy, irritable, listless, depressed, buoyant
- Interpersonal stances: Affective stance toward another person in a specific interaction
  - Friendly, flirtatious, distant, cold, warm, supportive, contemptuous
- Attitudes: Enduring, affectively colored beliefs, dispositions towards objects or persons
  - Liking, loving, hating, valuing, desiring
- Personality traits: Stable personality dispositions and typical behavior tendencies
  - Nervous, anxious, reckless, morose, hostile, jealous

Basic Sentiment Classification

Sentiment analysis is the detection of attitudes.

- Simple task we focus on in this class
  - Is the attitude of this text positive or negative?

Summary: Text Classification

Text classification can be applied to various tasks, including:

- Sentiment analysis
- Spam detection
- Authorship identification
- Language identification
- Assigning subject categories, topics, or genres
- ...

Text Classification: Definition

Input:

- A document d
- A fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

Output: A predicted class $c \in C$

Classification Methods: Hand-coded rules

Rules based on combinations of words or other features

- Spam: *black-list-address* OR (*“dollars” AND “you have been selected”*)
- Accuracy can be high if rules carefully refined by experts
- But building and maintaining these rules is expensive

Classification Methods: Supervised Machine Learning

Input:

- A document d
- A fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
- A training set of m hand-labeled documents: $(d_1, c_1), (d_2, c_2), \ldots, (d_m, c_m)$

Output:

- A learned classifier $\gamma : d \rightarrow c$

Classification Methods: Supervised Machine Learning

Any kind of classifier can be used:

- Naïve Bayes
- Logistic regression
- Neural networks
- k-Nearest Neighbors

Naive Bayes Intuition

Naive Bayes is a simple (”naive”) classification method based on Bayes’ rule.

- Relies on a very simple representation of a document: *Bag of words*

The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

Bayes' Rule Applied to Documents and Classes

For a document d and a class c:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

Naive Bayes Classifier (I)

- MAP stands for ”maximum a posteriori,” which represents the most likely class:

$$c_{\text{MAP}} = \arg\max_{c \in C} P(c|d)$$

- To calculate the most likely class, we apply Bayes' rule:

$$= \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)}$$

- Finally, we can drop the denominator since it remains constant for all classes:

$$= \arg\max_{c \in C} P(d|c)P(c)$$

Naive Bayes Classifier (II)

- To classify document d, we use the MAP estimate:

$$c_{\text{MAP}} = \arg\max_{c \in C} P(d|c)P(c)$$

- The document d is represented as a set of features $x_1, x_2, \ldots, x_n$.
- The classifier calculates the conditional probability of the features given a class and the prior probability of the class:

$$= \arg\max_{c \in C} P(x_1, x_2, \ldots, x_n|c)P(c)$$

- The term $P(x_1, x_2, \ldots, x_n|c)$ represents the ”likelihood” of the features given the class.
- The term $P(c)$ represents the ”prior” probability of the class.

Naïve Bayes Classifier (IV)

- The Naïve Bayes classifier calculates the MAP estimate by considering the likelihood and prior probabilities:

$$c_{\text{MAP}} = \arg\max_{c \in C} P(x_1, x_2, \ldots, x_n|c)P(c)$$

- The probability of the features given the class, $P(x_1, x_2, \ldots, x_n|c)$, can be estimated by counting the relative frequencies in a corpus.
- The prior probability of the class, $P(c)$, represents how often this class occurs.
- The Naïve Bayes classifier requires estimating $O(|X|^n \cdot |C|)$ parameters, where $|X|$ is the number of distinct features and n is the length of the document.
- Accurate estimation of these parameters would require a very large number of training examples.

Multinomial Naive Bayes Independence Assumptions

- Bag of Words assumption: We assume that the position of words in the document does not matter.
- Conditional Independence assumption: We assume that the feature probabilities $P(x_i|c_j)$ are independent given the class c_j.
- In the Multinomial Naive Bayes classifier, the probability of a document with features $x_1, x_2, \ldots, x_n$ given class c can be calculated as:

$$P(x_1, x_2, \ldots, x_n|c) = P(x_1|c) \cdot P(x_2|c) \cdot P(x_3|c) \cdot \ldots \cdot P(x_n|c)$$

Multinomial Naive Bayes Classifier

- The Maximum A Posteriori (MAP) estimate for class c in the Multinomial Naive Bayes classifier is given by:

$$c_{\text{MAP}} = \arg\max_{c \in C} P(x_1, x_2, \ldots, x_n | c) P(c)$$

- Alternatively, we can write it as:

$$c_{\text{NB}} = \arg\max_{c \in C} P(c_j) \prod_{x \in X} P(x|c)$$

- $P(c_j)$ represents the prior probability of class c_j.
- $\prod_{x \in X} P(x|c)$ represents the likelihood of the features $x_1, x_2, \ldots, x_n$ given class c.

Applying Multinomial Naive Bayes Classifiers to Text Classification

- The Multinomial Naive Bayes classifier for text classification can be applied as follows:

$$c_{\text{NB}} = \arg\max_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i|c_j)$$

- c_{NB} represents the predicted class for the test document.
- C is the set of all possible classes.
- $P(c_j)$ is the prior probability of class c_j.
- $\prod_{i \in \text{positions}} P(x_i|c_j)$ calculates the likelihood of each feature x_i at position i given class c_j.
- The product is taken over all word positions in the test document.

Problems with Multiplying Lots of Probabilities

- Multiplying lots of probabilities can result in floating-point underflow, especially when dealing with small probabilities.
- Example: $0.0006 \times 0.0007 \times 0.0009 \times 0.01 \times 0.5 \times 0.000008 \ldots$
- Idea: Use logarithms, as $\log(ab) = \log(a) + \log(b)$.
- Instead of multiplying probabilities, we can sum the logarithms of probabilities.
- The Multinomial Naive Bayes classifier can be expressed using logarithms as follows:

$$c_{\text{NB}} = \arg\max_{c_j \in C} \left(\log(P(c_j)) + \sum_{i \in \text{position}} \log(P(x_i|c_j)) \right)$$

- By taking logarithms, we avoid the issue of floating-point underflow and perform calculations in the log space.
- The classifier becomes a linear model, where the prediction is the argmax of a sum of weights (log probabilities) and the inputs (log conditional probabilities):
- Thus, Naïve Bayes is a linear classifier, operating in the log space.

Learning the Multinomial Naive Bayes Model

First attempt: Maximum Likelihood Estimates

- The probabilities are estimated using the observed counts in the training data.
- The prior probability of a class c_j is estimated as:

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{\text{total}}}$$

  where N_{c_j} is the number of documents in class c_j and N_{total} is the total number of documents.
- The estimate of the probability of word w_i given class c_j is calculated as:

$$\hat{P}(w_i|c_j) = \frac{\text{count}(w_i, c_j)}{\sum_{w \in V} \text{count}(w, c_j)}$$

  where $w \in V$ represents a word in the vocabulary V.
- The denominator is the sum of counts of all words in the vocabulary within class c_j.

Parameter Estimation

To estimate the parameters of the Multinomial Naive Bayes model, we follow these steps:

- Create a mega-document for each topic c_j by concatenating all the documents in that topic.
- We calculate the frequency of word w_i in the mega-document, which represents the fraction of times word w_i appears among all words in the documents of topic c_j.
- The estimated probability $\hat{P}(w_i|c_j)$ of word w_i given class c_j is obtained by dividing the count of occurrences of w_i in the mega-document of topic c_j by the total count of words in the mega-document:

$$\hat{P}(w_i|c_j) = \frac{\text{count}(w_i, c_j)}{\sum_{w \in V} \text{count}(w, c_j)}$$

Here, $\text{count}(w_i, c_j)$ represents the number of times word w_i appears in the mega-document of topic c_j, and $\text{count}(w, c_j)$ is the total count of words in the mega-document.

Problem with Maximum Likelihood

Zero probabilities and the issue of unseen words

- Consider the scenario where we have not encountered the word ”fantastic” in any training documents classified as positive (thumbs-up).
- Using maximum likelihood estimation, the probability $\hat{P}(\text{”fantastic”} \mid \text{positive})$ would be calculated as:

$$\hat{P}(\text{”fantastic”} \mid \text{positive}) = \frac{\text{count}(\text{”fantastic”}, \text{positive})}{\sum_{w \in V} \text{count}(w, \text{positive})}$$

- In this case, the count of the word ”fantastic” in positive documents is zero, leading to a zero probability:

$$\hat{P}(\text{”fantastic”} \mid \text{positive}) = \frac{0}{\sum_{w \in V} \text{count}(w, \text{positive})} = 0$$

- However, zero probabilities cannot be conditioned away, regardless of the other evidence present.
- This poses a problem when calculating the maximum a posteriori (MAP) estimate, which is used for classification:

$$c_{\text{MAP}} = \arg\max_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

- With a zero probability for a word, the entire expression becomes zero, regardless of other evidence.

Laplace (Add-1) Smoothing for Naïve Bayes

Handling zero probabilities with Laplace (Add-1) smoothing

- To address the problem of zero probabilities, we can employ Laplace (Add-1) smoothing technique.
- The smoothed estimate $\hat{P}(w_i \mid c)$ is calculated as:

$$\hat{P}(w_i \mid c) = \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V}(\text{count}(w, c) + 1)}$$

- Here, an additional count of 1 is added to both the numerator and the denominator.
- The denominator is adjusted by adding the size of the vocabulary V to ensure proper normalization.
- By doing so, we prevent zero probabilities and allow some probability mass to be distributed to unseen words.
- This smoothing technique helps to mitigate the issue of unseen words and avoids the complete elimination of certain classes during classification.

Multinomial Naïve Bayes: Learning

Learning the Multinomial Naïve Bayes Model

- In order to learn the parameters of the model, we need to calculate the terms $P(c_j)$ and $P(w_k \mid c_j)$.
- For each class c_j in the set of classes C, we perform the following steps:
  - Retrieve all the documents $docs_j$ that belong to class c_j.
  - Calculate the term $P(w_k \mid c_j)$ for each word w_k in the vocabulary V:

$$P(w_k \mid c_j) = \frac{n_k + \alpha}{n + \alpha \cdot |\text{Vocabulary}|}$$

    where n_k represents the number of occurrences of word w_k in the concatenated document $Text_j$.
  - Calculate the prior probability $P(c_j)$:

$$P(c_j) = \frac{|docs_j|}{|\text{total number of documents}|}$$

- To calculate $P(w_k \mid c_j)$, we need to extract the vocabulary V from the training corpus.

Unknown Words

Dealing with unknown words in the test data:

- When we encounter unknown words in the test data that do not appear in the training data or vocabulary, we ignore them.
- We remove these unknown words from the test document as if they were not present at all.
- We do not assign any probability to these unknown words in the classification process.

Why don't we build an unknown word model?

- Building a separate model for unknown words is not generally helpful.
- Knowing which class has more unknown words does not provide useful information for classification.

Stop Words

Stop words are frequently used words like ”the” and ”a” that are often considered to have little or no significance in text classification. Some systems choose to ignore stop words in the classification process. Here is how it is typically done:

- Sort the vocabulary by word frequency in the training set.
- Create a stopword list by selecting the top 10 or 50 most frequent words.
- Remove all stop words from both the training and test sets, treating them as if they were never there.

However, removing stop words doesn’t usually improve the performance of Naive Bayes classifiers. Therefore, in practice, most Naive Bayes algorithms use all words and do not utilize stopword lists.

Worked Sentiment Example

Training data:

Category	Text
Negative	Just plain boring, entirely predictable and lacks energy.
Negative	No surprises and very few laughs.
Positive	Very powerful.
Positive	The most fun film of the summer.

Test:

Category	Text
?	Predictable with no fun.

Worked Sentiment Example

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable ~~with~~ no fun

1. Prior from training:

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{total}}$$

P(-) = 3/5

P(+) = 2/5

2. Drop "with"

3. Likelihoods from training:

$$p(w_i|c) = \frac{count(w_i, c) + 1}{(\sum_{w \in V} count(w, c)) + |V|}$$

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \qquad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“no”}|-) = \frac{1+1}{14+20} \qquad P(\text{“no”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \qquad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

4. Scoring the test set:

$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

Naive Bayes as a Language Model

- When using individual word features and considering all words in the text, naive Bayes has an important similarity to language modeling.
- Specifically, a naive Bayes model can be viewed as a set of class-specific unigram language models, in which the model for each class instantiates a unigram language model.
- The likelihood features from the naive Bayes model assign a probability to each word $P(word|c)$, and the model also assigns a probability to each sentence:

$$P(s|c) = \prod_{i \in positions} P(w_i|c)$$

Consider a naive Bayes model with the classes positive (+) and negative (-) and the following model parameters:

w	$P(w\|+)$	$P(w\|-)$
I	0.1	0.2
love	0.1	0.001
this	0.01	0.01
fun	0.05	0.005
film	0.1	0.1
...	...	...

Naive Bayes as a Language Model

- Each of the two columns above instantiates a language model that can assign a probability to the sentence ”I love this fun film”:

$$P(\text{''I love this fun film''}|+) = 0.1 \times 0.1 \times 0.01 \times 0.05 \times 0.1 = 0.0000005$$

$$P(\text{''I love this fun film''}|-) = 0.2 \times 0.001 \times 0.01 \times 0.005 \times 0.1 = 0.0000000010$$

- As it happens, the positive model assigns a higher probability to the sentence:

$$P(s|pos) > P(s|neg)$$

- Note that this is just the likelihood part of the naive Bayes model; once we multiply in the prior, a full naive Bayes model might well make a different classification decision.

Evaluation

- Let's consider just binary text classification tasks.
- Imagine you're the CEO of Delicious Pie Company.
- You want to know what people are saying about your pies.
- So you build a "Delicious Pie" tweet detector with the following classes:
  - Positive class: tweets about Delicious Pie Co
  - Negative class: all other tweets

The 2-by-2 Confusion Matrix

	System Positive	**System Negative**
Gold Positive	True Positive (TP)	False Negative (FN)
Gold Negative	False Positive (FP)	True Negative (TN)

Recall (also known as **Sensitivity** or **True Positive Rate**):

$$\text{Recall} = \frac{TP}{TP + FN}$$

Precision:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Accuracy:

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

Evaluation: Accuracy

Why don't we use accuracy as our metric?
Imagine we saw 1 million tweets:

- 100 of them talked about Delicious Pie Co.
- 999,900 talked about something else.

We could build a dumb classifier that just labels every tweet "not about pie":

- It would get 99.99% accuracy!!! Wow!!!!
- But it would be useless! It doesn't return the comments we are looking for!

That's why we use precision and recall instead.

Evaluation: Precision and Recall

Precision measures the percentage of items the system detected (i.e., items the system labeled as positive) that are in fact positive (according to the human gold labels).

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}$$

Recall measures the percentage of items that were correctly identified by the system out of all the items that should have been identified.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

Why Precision and Recall?

Consider our dumb pie-classifier that just labels nothing as ”about pie.”

- Accuracy = 99.99% (it correctly labels most tweets as not about pie)
- Recall = 0 (it doesn’t detect any of the 100 pie-related tweets)

Precision and recall, unlike accuracy, emphasize true positives:

- They focus on finding the things that we are supposed to be looking for.

A Combined Measure: F-measure

The F-measure is a single number that combines precision (P) and recall (R), defined as:

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

The F-measure, defined with the parameter β, differentially weights the importance of recall and precision.

- $\beta > 1$ favors recall
- $\beta < 1$ favors precision

When $\beta = 1$, precision and recall are equal, and we have the balanced F_1 measure:

$$F_1 = \frac{2PR}{P + R}$$

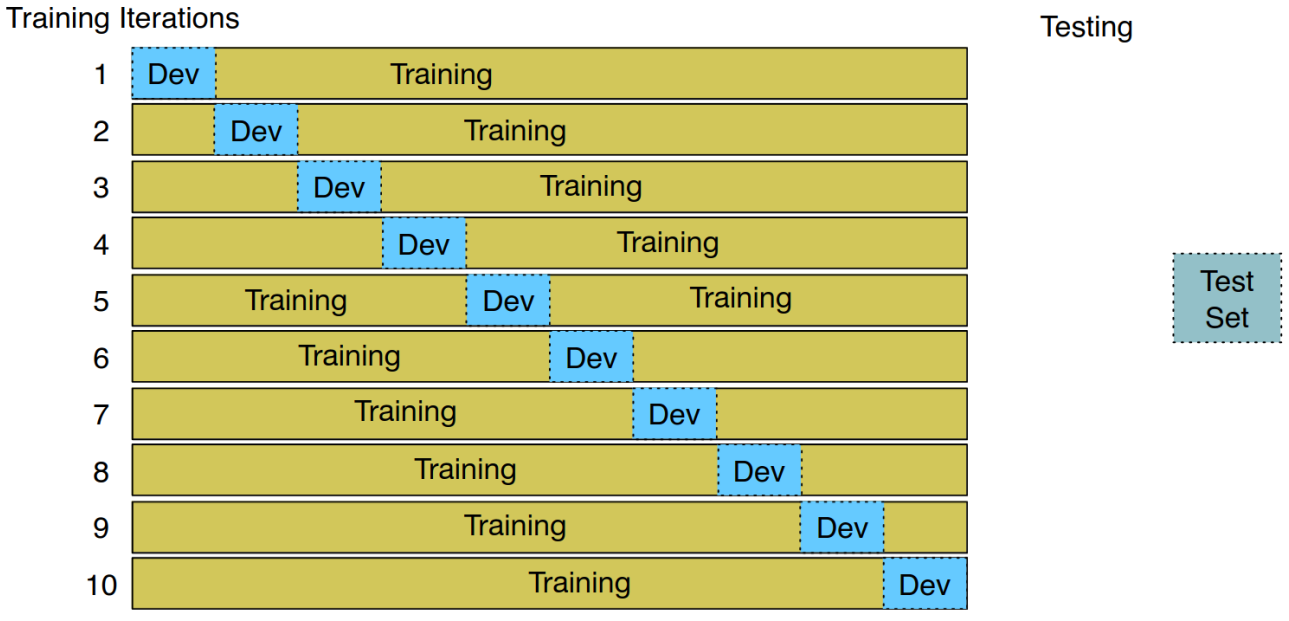
Development Test Sets ("Devsets")

- To avoid overfitting and provide a more conservative estimate of performance, we commonly use a three-set approach: training set, devset, and testset.

  - **Training set**: Used to train the model.
  - **Devset**: Used to tune the model and select the best hyperparameters.
  - **Testset**: Used to report the final performance of the model.
- This approach ensures that the model is not tuned specifically to the test set, avoiding overfitting.
- However, it creates a paradox: we want as much data as possible for training, but also for the devset.
- How do we split the data?

Cross-validation: Multiple Splits

When working with limited data and wanting to gain insights from the corpus, a common approach is to create a fixed training set and test set. Then, perform 10-fold cross-validation within the training set while evaluating the error rate in the test set.



This approach allows us to pool the results over the splits and compute the pooled dev performance of our system. It provides a balance between utilizing the available data for training and gaining insights from the corpus.

Questions?

Thanks for your Attention!

References I