

THE UNIVERSITY *of* EDINBURGH

Text Technologies for Data Science

INFR11145

Text Classification (2)

Instructor:

Björn Ross

18-Nov-2026

Lecture Objectives

- Implement your first text classifier in easy steps
- Show (some) steps that often happen “under the hood” in popular libraries
- This is a practical lecture
  No equations this time ☺

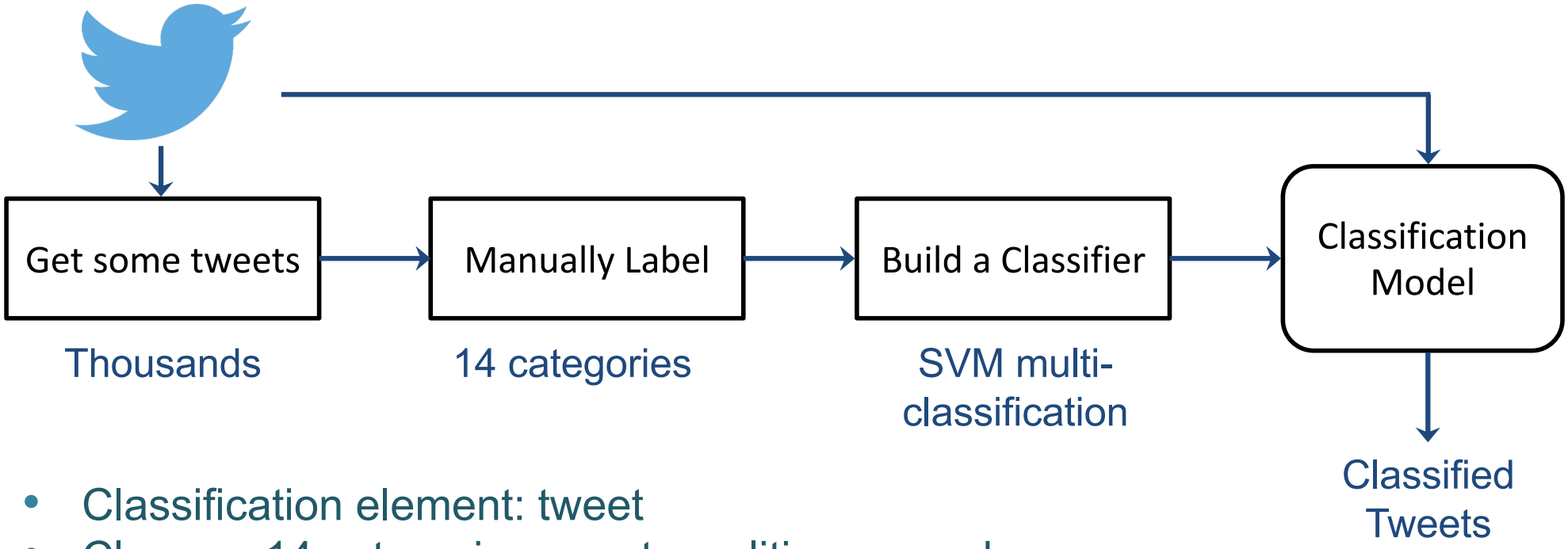
Building a text classifier: Ingredients



- Text elements to be classified
  - Document, paragraph, sentence
- Set of predefined classes (classification task)
  - At least two (binary)
  - Topical, spam, relevance, sentiment, …
- Training set
  - Enough samples of text elements for each class
- Development set + test set
  - Some samples of each class that not used in training
- Feature set
  - A set of features extracted from the text to train the classifier
- Classifier
  - The ML module that learns a classification model

Building a text classifier: Application

- Classifying tweets into general-purpose categories



- Classification element: tweet
- Classes: 14 categories: sports, politics, comedy, …
- Data: 2,503 training + 625 test tweets; 20% of training held out as development set
- Features: BOW
- Classifier: SVM multiclass classifier

Building a text classifier: Steps

1. Prepare training data
   required: piece of text (tweet) + label to class
2. Extract features
   1. Pre-process text: lowercase, tokenise, remove useless strings
   2. Create a list of all unique terms in the training data. Give each term a unique ID
   3. Convert the text into features, by replacing each term with its corresponding feature ID. Add value to the feature (simplest: value "1" if exists, or count of occurrences)
3. Prepare development data (and, at the end, test data)
   Convert the text into features using the same mapping from the training data. For terms that are not in the features list, it could be neglected, or assigned to an ID representing OOV.
4. Run the learning process on the training data features to create a model
5. Run the classification on the development features and get predictions (test features: once, at the end)
6. Evaluate performance

Examples

- Tweet + Label

Kobe passes Wilt for 4th on all-time scoring list	Sports

- Learned features (BOW) from training data
- After converting text to feature vectors

	0		2943	2944	2945	2946		8330	8331		10000
0	0	…	0	1	0	0	…	1	0	…	0
1	0	…	1	0	0	1	…	0	0	…	0
…							…			…	

Feature ID	Corresponding word
2944	kobe
2945	rapping
..	
4525	4th
4526	trevi
..	
8330	passes
8331	ducks
..	
9929	17
9930	wilt
…	

- SVM prediction output

7

Predicted Class ID

Practical

Same tweets, six methods

Method	Labelled tweets used	Macro-F1, dev (501)	Macro-F1, sample (50)	Time for 50 tweets
Majority class	2,002	0.01	0.01	–
BOW + linear SVM	2,002	0.54	0.54	< 0.1 s
BOW + random forest	2,002	0.59	0.53	< 0.1 s
MiniLM embeddings + LR	2,002	0.76	0.71	≈ 1 s
Zero-shot prompt (hosted LLM)	0	–	0.74	≈ 25 s + pauses
Few-shot prompt (hosted LLM)	14	–	0.78	≈ 30 s + pauses

- Same tweets for every method; all choices made on the development split
- 50 tweets: one tweet = 2 points of accuracy. Test set: once, for the chosen model

Your turn: spot the leak

	What the code does	Leak or fine?
A	vocab = set(words in train_docs + test_docs)	
B	pick top-1,000 features by χ^2 on train + dev, then score on dev	
C	try C = 0.1, 1, 10, 100; keep the best test macro-F1	
D	few-shot examples drawn from the dev set, then score on dev	
E	scaler fitted on train, applied to dev and test	
F	random split; many retweets of the same tweet in the data	

We evaluate on the development set (or, at the end, the test set). Which steps let held-out data influence the model?

Spot the leak: answers

	What the code does		Why
A	vocab = set(words in train_docs + test_docs)	**Leak**	test words get feature ids; build the vocabulary from the training split
B	pick top-1,000 features by χ^2 on train + dev, then score on dev	**Leak**	feature selection saw the dev labels; compute χ^2 on the training split
C	try C = 0.1, 1, 10, 100; keep the best test macro-F1	**Leak**	a choice made on the test set; choose on dev, test once
D	few-shot examples drawn from the dev set, then score on dev	**Leak**	the prompt contains dev tweets with their labels; take examples from training
E	scaler fitted on train, applied to dev and test	**Fine**	fitted on training data only
F	random split; many retweets of the same tweet in the data	**Leak**	copies end up in train and dev; keep copies in one split

Possible Improvements

- Feature Extraction
  - Apply stemming & stopping
  - Duplicate hashtags words (#car → #car car)
  - Expand tweet text that has link with the page title of that link
  - Add new set of features to the terms appearing in the profile description of the author of the tweet
    - E.g. tweets terms features: ID range: 1 → 12000
      profile terms features: ID range: 12001 → 20000
    - If a term appeared in the tweet and in the profile description, these are two different features with two different IDs
  - Try non-textual features
    - Tweet length, presence of hashtags, links, emojis …

Possible Improvements

- Feature weighting
  - Using tfidf, BM25 as the feature value instead of binary
- Learning method
  - Test other ML learning methods other than SVM
    - Random forest
    - Decision trees
    - Naive Bayesian
  - Try pretrained models (embeddings, fine-tuning, prompts)
    - Google Colab: you can experiment with using GPUs for free! https://colab.research.google.com/
- Add more training data
  - Think about a way to create more training data

Resources

- Magdy W., H. Sajjad, T. El-Ganainy and F. Sebastiani. (2015) Bridging Social Media via Distant Supervision.
  Springer SNAM 2015 link, arXiv
- Additional reading:
  Nguyen, D. P., Gravel, R., Trieschnigg, R. B., & Meder, T. How old do you think I am? A study of language and age in Twitter.
  ICWSM 2013
- *Huggingface (2024)*
  Text Classification
  Link: https://huggingface.co/tasks/text-classification
- *Sentence Transformers (2024)*
  Link: https://www.sbert.net/