CS918: LECTURE 7

Text Classification, Evaluation and Error Analysis

Arkaitz Zubiaga, 24th October, 2018

LECTURE 7: CONTENTS

- What is text classification?
  - Examples of text classification.
- Supervised Text Classification.
  - Sentiment analysis.
- Evaluation.
- Error Analysis.

WHAT IS TEXT CLASSIFICATION?

WARWICK

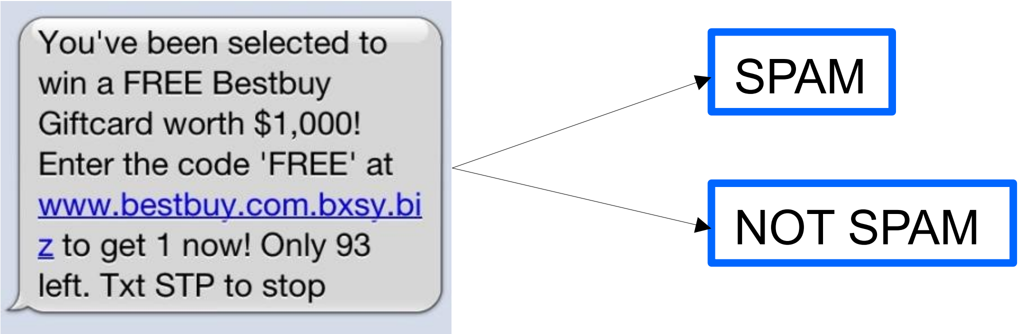
- Having as input:
  - A **text document d**
  - A **set of categories C={c_1, …, c_m}**

- The **text classification** task **outputs**:
  - Predicted class **c*** that **document d belongs to**.

$$c^* \in C$$

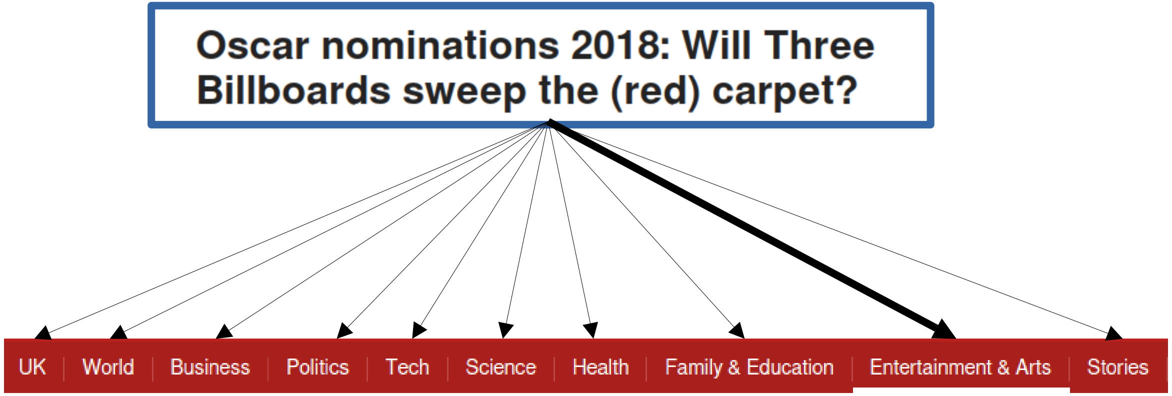
EXAMPLES OF TEXT CLASSIFICATION

- **Spam detection**: classifying emails/web pages as spam (or not).



EXAMPLES OF TEXT CLASSIFICATION

- **Classification by topic**: what is the text about?



EXAMPLES OF TEXT CLASSIFICATION

- **Sentiment analysis**: is a text positive, negative or neutral?

☺ • I really **liked** the food at the restaurant.

😐 • We were 8 friends who went there for the first time.

☹ • The service was **terrible**.

EXAMPLES OF TEXT CLASSIFICATION

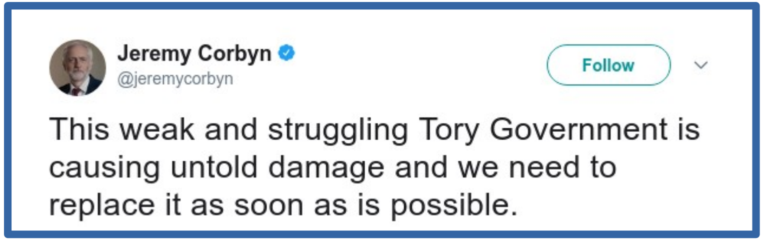
- **Language identification**: what language a text is written in?

Wieviel Uhr ist es? $\longrightarrow$ {German, English, Spanish, French}

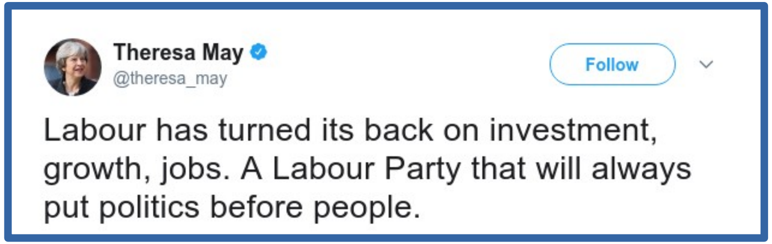
EXAMPLES OF TEXT CLASSIFICATION

WARWICK

- **Classification of political orientation**:
  does a text support labour or conservative?



Labour



Conservative

WHAT IS TEXT CLASSIFICATION?

- A range of different problems, with a **common goal**:
  - Assigning a **category/class to each document**.
  - We **know the set of categories** beforehand.

TEXT CLASSIFICATION: APPROACHES

- **Rule-based classifiers**, e.g. if email contains 'viagra' → spam
  - Significant **manual effort** involved.

- **Supervised classification**:
  - **Given**: a hand-labeled set of **document-class pairs**
    (d_1,c_1), (d_2, c_j), …, (d_m,c_2) → classified into $C=\{c_1, …, c_j\}$
  - The classifier **learns a model** that can **classify new documents into C**.

SUPERVISED TEXT CLASSIFICATION

SUPERVISED CLASSIFICATION

- **Assumption:** We have a manually labelled dataset, e.g.:
  - d_1: 'That's really good, I love it' → **positive**
  - d_2: 'It was boring, don't recommend it' → **negative**
  - …
  - d_n: 'I wouldn't go again, awful' → **negative**
- **If not**, we need to **find one or label one ourselves**.

SUPERVISED CLASSIF.: DECISIONS TO MAKE

WARWICK

- Split the dataset into train/dev/test sets.
  - or often just train/test.

- What **features** are we going to use to represent the documents?

- What **classifier** are we going to use?
  - Choose **settings, parameters**, etc. for the classifier.

SPLITTING THE DATASET

- We can **split the dataset into 3 parts**:
  - Training set → the largest set as we want proper training.
  - Development set.
  - Test set.

Training set	Development Set	Test Set

- **Tweak classifier based on the development set**, then test it on the test set.
  - **Tweaking and testing on the test set may lead to overfitting** (doing the right things specifically for that test set, not necessarily generalisable)

WHAT IS OVERFITTING?

- **Overly adjusting** our classifier and features **to a specific test set**.
- The classifier **may not generalise** to new instances beyond the current test set.

Pretty much like designing a mattress based on a single observation of sleeping position!

HOW TO AVOID OVERFITTING

- Having train/dev/test, **tweak our classifier for the dev set**.
    - Then apply to the **test set**, does it **generalise**?

- **Cross-validation**.

CROSS-VALIDATION

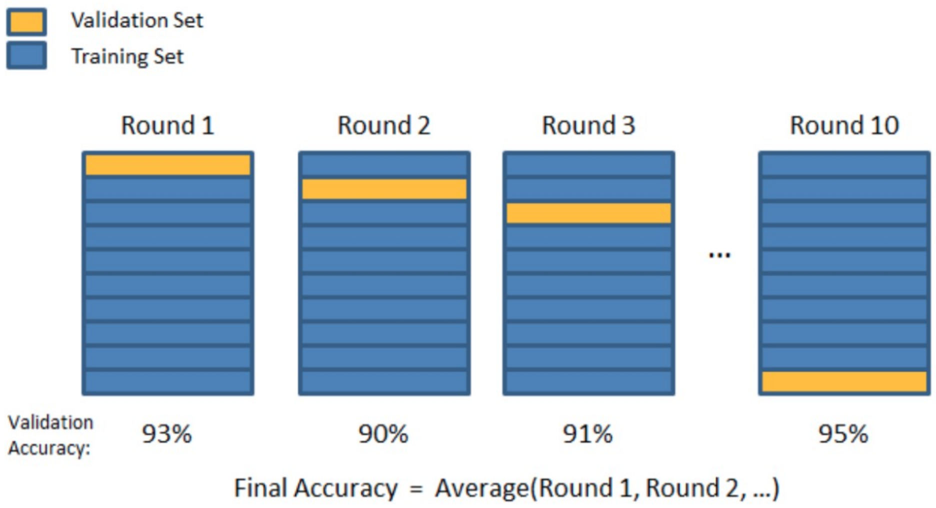
WARWICK

- **Cross-validation:** train and test on different "folds"
  - e.g. 10-fold cross-validation, split the data into 10 parts.
  - each time 1 fold is used for testing, the other 9 for training.
  - after all 10 runs, compute the average performance.

CROSS-VALIDATION

WARWICK

- **Cross-validation:** example.



CHOOSING THE FEATURES

- Usually **start with some basic features**:
  - **Bag of words**.
  - Or preferably **word embeddings**.

- Keep **adding new features**:
  - **Need to be creative.**
    Think of features could **characterise the problem at hand**.

THINKING OF FEATURES

WARWICK

- **Possible features:**
    - **Sentiment analysis** → counts of positive/negative words.
    - **Language identification** → probabilities of characters (how many k's, b's, v's...), features from word suffixes (e.g. many -ing words → English)
    - **Spam detection** → count words in blacklist, domain of URLs in email (looking for malicious URLs)

CHOOSING THE FEATURES

- How to **assess which features are good**?
- **Empirical evaluation**:
  - **Incremental testing**:
    keep adding features, see if adding improves performance
  - **Leave-one-out testing**:
    test all features, and combinations of all features except one.
    when leaving feature i out performs better than all features, remove feature I
  - **Error analysis:** (later in this lecture)
    look at classifier's errors, what features can we use to improve?

CHOOSING A CLASSIFIER

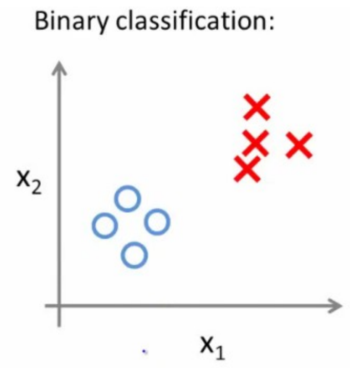
WARWICK

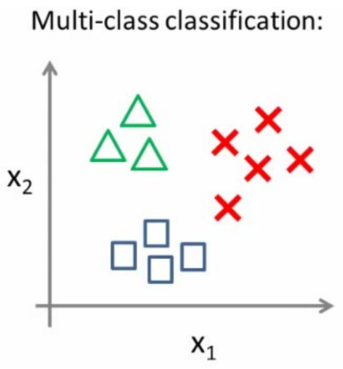
- Many different classifiers exist, **well-known classifiers** include:
  - **Naive Bayes**.
  - **Logistic Regression** (Maximum Entropy classifier)
  - **Support Vector Machines** (SVM).
- Classifiers can be **binary** (k = 2) or **multiclass** (k > 2).

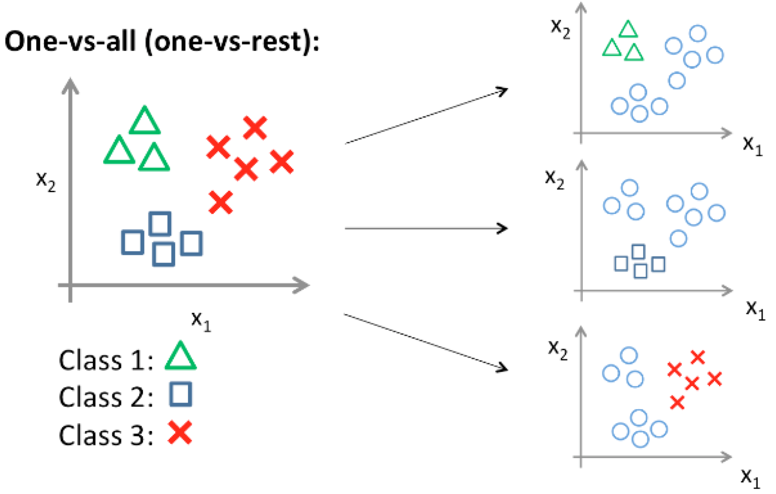
CHOOSING A CLASSIFIER

- **How many categories** (k)?
  - [k=2] **Binary** → binary classifier.
  - [k>2] **Multiclass**:
    - **One-vs-all** classifiers.
      Build *k* classifiers, each able to distinguish class *i* from the rest. Then combine output of all classifiers (e.g. based on their confidence scores)
    - **Multinomial/multiclass** classifiers.

BINARY VS MULTICLASS CLASSIFIERS







WHEN TO USE MULTINOMIAL OR ONE-VS-ALL?

WARWICK

- **Multinomial:**
  - **is generally faster**, a single classifier.
  - classes are **mutually exclusive**, no overlap.
- **One-vs-all:**
  - **Multilabel classification**, document can fall in **1+ categories**: e.g. classify by language:

    I said "bonjour mon ami" to my friend → English & French

    How? Out of k classifiers, those with **confidence > threshold**

SENTIMENT ANALYSIS

HOW TO BUILD A SENTIMENT CLASSIFIER

- Two main approaches:
  - Unsupervised classifier using lexicons.
  - Supervised classifier leveraging training data.

UNSUPERVISED CLASSIFIER USING LEXICONS

- Using **lexicons** of positive and negative words, e.g.:

```
fantastic
fantastically
fascinate
fascinating
fascinatingly
fascination
fashionable
fashionably
```

```
bad
badly
baffle
baffled
bafflement
baffling
```

- Count **number of +/- words** in a review:
  - **Majority wins** (are there more + or - words?).
  - **Produce percentages** (x% positive, y% negative).

CLASSIFIER USING LEXICONS

- + No need for training data (reviews annotated as +/-), just lexicons.
- + Quick and easy to implement.
- - Can't handle negations and other expressions, e.g.:

☺ The restaurant is not **bad** at all, so we will be back!

☺ I had heard that the service was **slow**, the food was **terrible**, and that it was **pricey**… but hey, it's actually **awesome**!

WORKAROUND FOR NEGATIONS

WARWICK

- The restaurant is not bad at all, so we will be back!
- **Workaround:** add “not_” to words after negation, up to next punctuation.
- The restaurant is not **not_bad not_at not_all**, so we will be back!
- **Expand lexicons to incorporate “not_*” in opposite list**, e.g.:
  not_happy, not_good as negative
  not_bad, not_sad as positive

SUPERVISED SENTIMENT CLASSIFIER

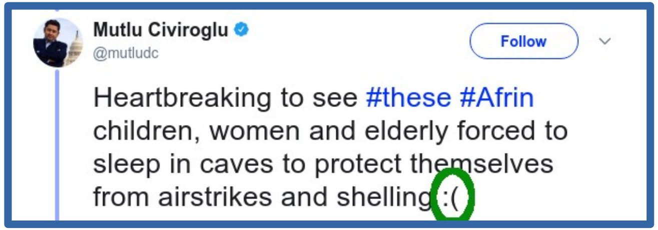
WARWICK

- Pretty much like any text classification, we **need labelled data**.
- Where we don’t have data, a good solution is **distant supervision**.

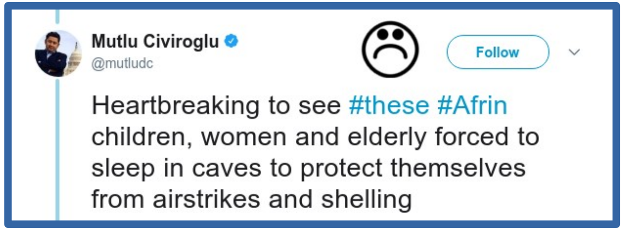
DISTANT SUPERVISION

WARWICK

- Use positive and negative keywords to collect data, e.g.: `#happy, :), #sad, :(,…`

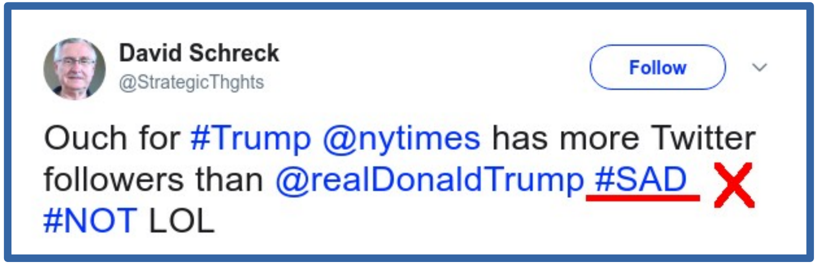


- Build dataset after removing those keywords:



DISTANT SUPERVISION

- + **Easy and quick** to get large collection of data.
- + **Generally free**, e.g. Twitter.
- - We can only get **positive/negative labels**, not neutral.
- - Data tends to be **noisy**, e.g. sarcasm.



EVALUATION OF TEXT CLASSIFICATION

EVALUATION OF TEXT CLASSIFICATION

- Evaluation is **different for binary and multiclass** classification.
  - **Binary:** we generally have a **positive and a negative class** (spam vs non-spam, medical test positive vs negative, exam pass vs fail).
    Classification **errors can only go the other class**.
  - **Multiclass:** multiple categories, may have different level of importance.
    Classification **errors can go to any other class**.

EVALUATION OF BINARY CLASSIFICATION

- 2-by-2 contingency table:

	Actually positive	**Actually negative**
Classified as positive	True Positive (TP)	False Positive (FP)
Classified as negative	False Negative (FN)	True Negative (TN)

EVALUATION OF BINARY CLASSIFICATION

- 2-by-2 contingency table:

	Actually positive	Actually negative
Classified as positive	True Positive (TP)	False Positive (FP)
Classified as negative	False Negative (FN)	True Negative (TN)

- **Precision:** ratio of items classified as positive that are correct.
- **Recall:** ratio of actual positive items that are classified as positive.

EVALUATION OF BINARY CLASSIFICATION

- 2-by-2 contingency table:

	Actually positive	Actually negative
Classified as positive	True Positive (TP)	False Positive (FP)
Classified as negative	False Negative (FN)	True Negative (TN)

- **Precision:** ratio of items classified as positive that are correct.
- **Recall:** ratio of actual positive items that are classified as positive.

EVALUATION OF BINARY CLASSIFICATION

- 2-by-2 contingency table:

	Actually positive	Actually negative
Classified as positive	True Positive (TP)	False Positive (FP)
Classified as negative	False Negative (FN)	True Negative (TN)

- **Precision:** ratio of items classified as positive that are correct.

$$\text{Precision} = \frac{tp}{tp + fp}$$

- **Recall:** ratio of actual positive items that are classified as positive.

$$\text{Recall} = \frac{tp}{tp + fn}$$

EVALUATION OF BINARY CLASSIFICATION

- We want to optimise for both precision and recall:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$ (harmonic mean of precision and recall)

- Equation as follows, however generally ß = 1:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$$

EVALUATION OF MULTICLASS CLASSIFICATION

WARWICK

- Bigger **confusion matrix**:

	Actually UK	Actually World	Actually Tech	Actually Science	Actually Politics	Actually Business
Classified as UK	**95**	1	13	0	1	0
Classified as World	0	**1**	0	0	0	0
Classified as Tech	10	90	**0**	1	0	0
Classified as Science	0	0	0	**34**	3	7
Classified as Politics	0	1	2	13	**26**	5
Classified as Business	0	0	2	14	5	**10**

EVALUATION OF MULTICLASS CLASSIFICATION

- **Overall Accuracy:** ratio of correct classifications.

$$\frac{\sum_i c_{ii}}{\sum_j \sum_i c_{ij}}$$

EVALUATION OF MULTICLASS CLASSIFICATION

- **Overall Accuracy:** ratio of correct classifications.

- **Generally a bad evaluation approach**, e.g.:
  - We classify 1,000 texts → data is skewed with 990 texts actually having positive sentiment.
  - We (naively) classify everything as positive.
  - 990 classified correctly: 990 / 1000 = 0.99 accuracy!

Is it fair?

EVALUATION OF MULTICLASS CLASSIFICATION

WARWICK

- **Per-class precision and recall**:

Precision: $\dfrac{c_{ii}}{\sum_j c_{ji}}$

→ # of items we correctly classified as class i

→ # of items we classified as class i

Recall: $\dfrac{c_{ii}}{\sum_j c_{ij}}$

→ # of items we correctly classified as class i

→ # of actual i items

- With the **harmonic mean**, we can then get **per-class F1 score**.

WARWICK

- We have **per-class precision, recall and F1 scores**.
  - How do we combine them all to **get a single score**?

EVALUATION OF MULTICLASS CLASSIFICATION

- Obtaining overall performances:
  - **Macroaveraging:**
    Compute **performance for each class, then average them**.
    All **classes contribute the same** to the final score (e.g. class with 990 and class with 10 instances).
  - **Microaveraging:**
    Compute **overall performance** without computing per-class performances.
    Large classes contribute more to the final score.

EVALUATION OF MULTICLASS CLASSIFICATION

WARWICK

- Macroaveraging:

$$\text{macroaveraged precision} = \frac{\sum_i \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}}{k}$$

$$\text{macroaveraged recall} = \frac{\sum_i \frac{\text{TP}}{\text{TP}_i + \text{FN}_i}}{k}$$

- The macroaveraged F1 score is then the harmonic mean of those.

EVALUATION OF MULTICLASS CLASSIFICATION

- Microaveraging:

$$\text{microaveraged precision} = \frac{\sum_i \mathrm{TP}_i}{\sum_i \mathrm{TP}_i + \mathrm{FP}_i}$$

$$\text{microaveraged recall} = \frac{\sum_i \mathrm{TP}}{\sum_i \mathrm{TP}_i + \mathrm{FN}_i}$$

- The microaveraged F1 score is then the harmonic mean of those.

MICRO- VS MACRO-AVERAGING EXAMPLE

WARWICK

	S	D	Q	C
S	**366 (40.4%)**	32 (3.5%)	22 (2.4%)	487 (53.7%)
D	38 (11.1%)	**22 (6.4%)**	23 (6.7%)	260 (75.8%)
Q	11 (3.1%)	10 (2.8%)	**149 (41.6%)**	188 (52.5%)
C	261 (9.0%)	91 (3.1%)	133 (4.6%)	**2,421 (83.3%)**

- Microaveraged F1 score: **0.665**
- Macroaveraged F1 score: **0.440**

where I have so many instances with label C, which metric is fairer?

depends on our objective: do we need a classifier that performs well for all labels? Do we just want to perform well for C?

FURTHER WEIGHTING/SELECTION

WARWICK

- We can choose to **prioritise certain categories**:
  - Give **higher weight to important categories**:
    $0.3*Prec(c_1) + 0.3*Prec(c_2) + 0.4*Prec(c_3)$
    → unless we have clear criteria to choose these weights, it's rather arbitrary though!

  - Select some categories for inclusion in macro/microaverage:
    - e.g. Semeval task (**Exercise 2 of the module**), we only **macroaverage over positive and negative sentiment classes**. Performance over the neutral class is not included.

ERROR ANALYSIS FOR TEXT CLASSIFICATION

ERROR ANALYSIS

WARWICK

- **Error analysis:** can help us find out where our classifier can do better.
- No magic formula for performing error analysis.
  - Look **where we are doing wrong**, what labels particularly.
  - Do our errors **have some common characteristics**? Can we infer a new feature from that?
  - Could our **classifier be favouring one of the classes** (e.g. the majority class)?

ERROR ANALYSIS

WARWICK

- **Error analysis:** where are we doing wrong? What labels?

  Look at frequent deviations in the confusion matrix.

	Actually UK	Actually World	Actually Tech	Actually Science	Actually Politics	Actually Business
Classified as UK	**95**	1	13	0	1	0
Classified as World	0	**1**	0	0	0	0
Classified as Tech	10	90	**0**	1	0	0
Classified as Science	0	0	0	**34**	3	7
Classified as Politics	0	1	2	13	**26**	5
Classified as Business	0	0	2	14	5	**10**

ERROR ANALYSIS

WARWICK

- **Error analysis:** do our errors have some common characteristics?

Print some of our errors, e.g. classifying person names by gender.

```
>>> for (tag, guess, name) in sorted(errors):
...     print('correct={:<8} guess={:<8s} name={:<30}'.format(tag, guess, name))
correct=female   guess=male     name=Abigail
  ...
correct=female   guess=male     name=Cindelyn
  ...
correct=female   guess=male     name=Katheryn
correct=female   guess=male     name=Kathryn
  ...
correct=male     guess=female   name=Aldrich
  ...
correct=male     guess=female   name=Mitch
  ...
correct=male     guess=female   name=Rich
  ...
```

ERROR ANALYSIS

WARWICK

- **Error analysis:** do our errors have some common characteristics?

Print some of our errors, e.g. classifying person names by gender.

```
>>> for (tag, guess, name) in sorted(errors):
...     print('correct={:<8} guess={:<8s} name={:<30}'.format(tag, guess, name))
correct=female   guess=male     name=Abigail
  ...
correct=female   guess=male     name=Cindelyn
  ...
correct=female   guess=male     name=Katheryn
correct=female   guess=male     name=Kathryn
  ...
correct=male     guess=female   name=Aldrich
  ...
correct=male     guess=female   name=Mitch
  ...
correct=male     guess=female   name=Rich
  ...
```

WARWICK

- **Error analysis:** do our errors have some common characteristics?

  Print some of our errors.

```
>>> for (tag, guess, name) in sorted(errors):
...     print('correct={:<8} guess={:<8s} name={:<30}'.format(tag, guess, name))
correct=female   guess=male     name=Abigail
  ...
correct=female   guess=male     name=Cindelyn
  ...
correct=female   guess=male     name=Katheryn
correct=female   guess=male     name=Kathryn
  ...
correct=male     guess=female   name=Aldrich
  ...
correct=male     guess=female   name=Mitch
  ...
correct=male     guess=female   name=Rich
  ...
```

New feature: suffix, last 2-3 characters

ERROR ANALYSIS

WARWICK

- **Error analysis:** could our classifier be favouring one of the classes?
  - Owing to class imbalance, classifiers tend to predict popular classes more often, e.g.:

    class A (700), class B (100), class C (100), class D (100)
    classifiers will tend to predict A, over-represented
    as in our previous example:

	S	D	Q	C
S	**366 (40.4%)**	32 (3.5%)	22 (2.4%)	487 (53.7%)
D	38 (11.1%)	**22 (6.4%)**	23 (6.7%)	260 (75.8%)
Q	11 (3.1%)	10 (2.8%)	**149 (41.6%)**	188 (52.5%)
C	261 (9.0%)	91 (3.1%)	133 (4.6%)	**2,421 (83.3%)**

ERROR ANALYSIS

WARWICK

- **Error analysis:** could our classifier be favouring one of the classes?
  - How to deal with imbalance (i.e. A-700, B-100, C-100, D-100)?
    - 1) Undersample popular class → A-100, B-100, C-100, D-100

      randomly remove 600 instances of A

WARWICK

- **Error analysis:** could our classifier be favouring one of the classes?
  - How to deal with imbalance (i.e. A-700, B-100, C-100, D-100)?
    - 2) Oversample other classes → A-700, B-700, C-700, D-700

repeat instances of B, C, D to match the number of A's

ERROR ANALYSIS

WARWICK

- **Error analysis:** could our classifier be favouring one of the classes?
  - How to deal with imbalance (i.e. A-700, B-100, C-100, D-100)?
    - 3) Create synthetic data → A-700, B-700, C-700, D-700

      generate new B, C, D items → needs some understanding of the contents of the classes to be able to produce sensible data items

- **Error analysis:** could our classifier be favouring one of the classes?
  - How to deal with imbalance (i.e. A-700, B-100, C-100, D-100)?
    - 4) Cost sensitive learning

      e.g. higher probability to predict uncommon classes
      P(A)=1/700, P(B)=1/100, P(C)=1/100, P(D)=1/100

      `scikit → class_weight=”auto”`

ERROR ANALYSIS

- **Important** for the error analysis:
  - **Subset we analyse for errors** (dev set) has to be **different to the one where we ultimately apply the classifier** (test set).
  - If we **tweak the classifier looking at the test set**, we'll end up **overfitting**, developing a classifier that works very well for that particular test set.

- **NB:** for **exercise 2**, you're **given 3 test sets**, we'll test it in **2 more held-out test sets**. Not looking at test sets while developing will improve **generalisability**.

ASSOCIATED READING

- Jurafsky, Daniel, and James H. Martin. 2009. Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. 3rd edition. **Chapters 4 and 5.**

- Bird Steven, Ewan Klein, and Edward Loper. Natural Language Processing with Python. O'Reilly Media, Inc., 2009. **Chapter 6.**