Sentiment Analysis on Social Media

CS918
Week 9

Adam Tsakalidis

Module Organiser: Dr Arkaitz Zubiaga

Definition

- Definition
  - Given a piece of text, classify it with respect to its sentiment
- Different levels
  - Sentence-, phrase-, entity-level
- Three classes
  - Positive, negative, neutral
- Two main approaches
  - Supervised VS Unsupervised
- Similar to other classification/clustering tasks

A.Tsakalidis@warwick.ac.uk

Background

- Closely related tasks:
  - Emotion Classification
  - Sarcasm Detection
  - Stance Detection
- Applications of SA:
  - Monitoring public mood
  - Election prediction
  - Stock market
  - Many others!

A.Tsakalidis@warwick.ac.uk

SA Approaches on OSM

- All-vs-All:
  - Three classes, one model
  - Most popular approach
- One-vs-All
  - Train on true/false examples for every class
  - One model per class
- Two-Step process
  - Subjectivity detection (neutral VS subjective)
  - Polarity detection (positive VS negative)

A.Tsakalidis@warwick.ac.uk

Approaches on SA

- Supervised Methods
  - Learn from labelled instances
  - Most popular/successful on OSM
- Unsupervised Methods
  - No need for manual annotation
  - Lower performance
- Semi-Supervised Methods
  - Manual annotation is expensive
  - Idea: pseudo-label training examples
  - e.g., :) vs :(

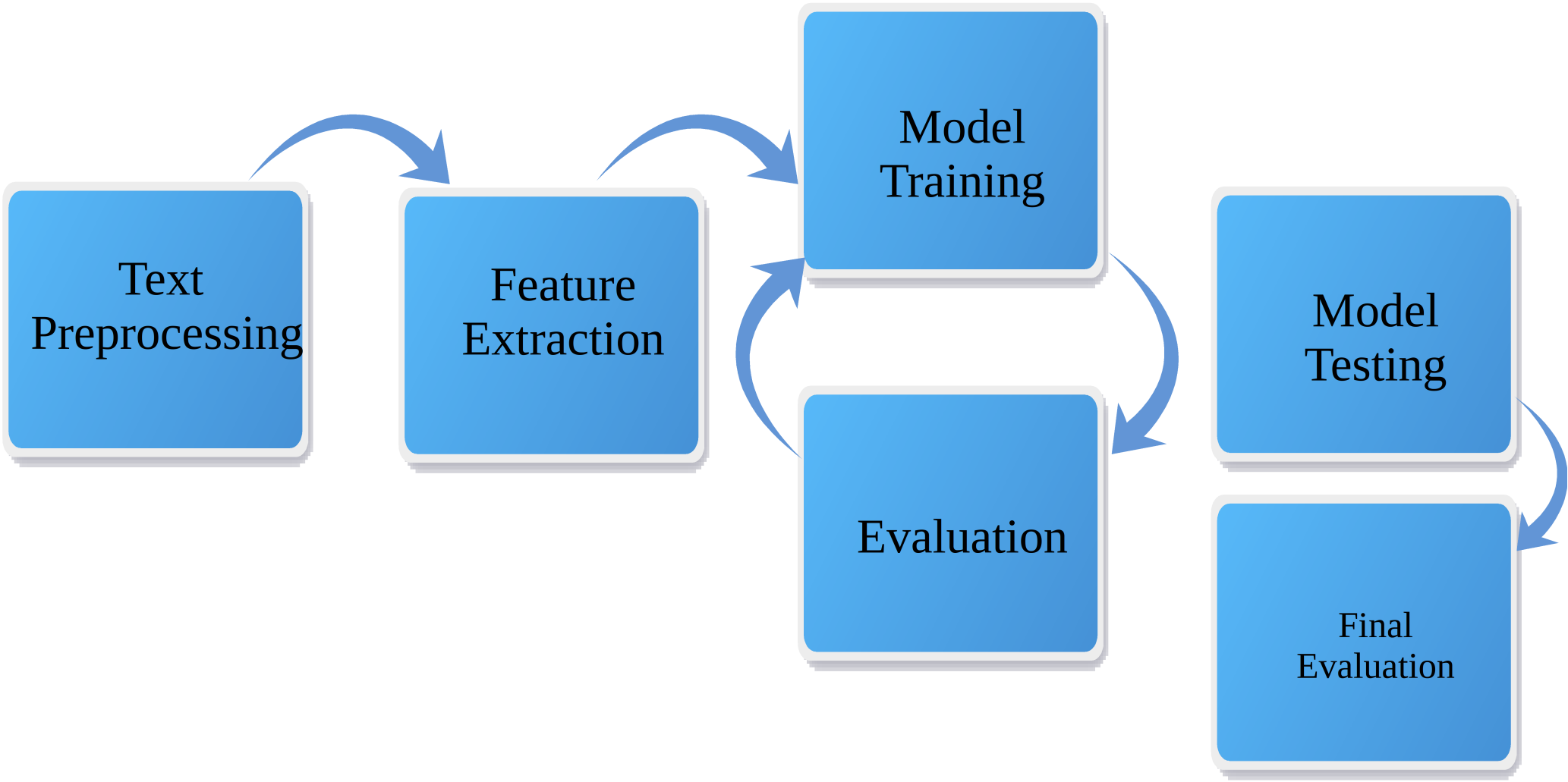
A.Tsakalidis@warwick.ac.uk

Exercise 2

- Tweet Sentiment Classification
  - **Task:** Given a twet, classify whether the tweet is of positive, negative, or neutral sentiment.
  - **Data:** one training set; one development set; 3 test sets
  - **Evaluation:** macro-average f-score (pos/neg)
- **Five steps**:
  - **Preprocess** the tweets (re-use your code!)
  - Extract **features** (see next)
  - Train **classifiers** on the extracted features of the training set
  - **Evaluate** using 10-fold cv (train set) or dev set
  - **Apply** best model to the (3) test sets & **evaluate** its performance

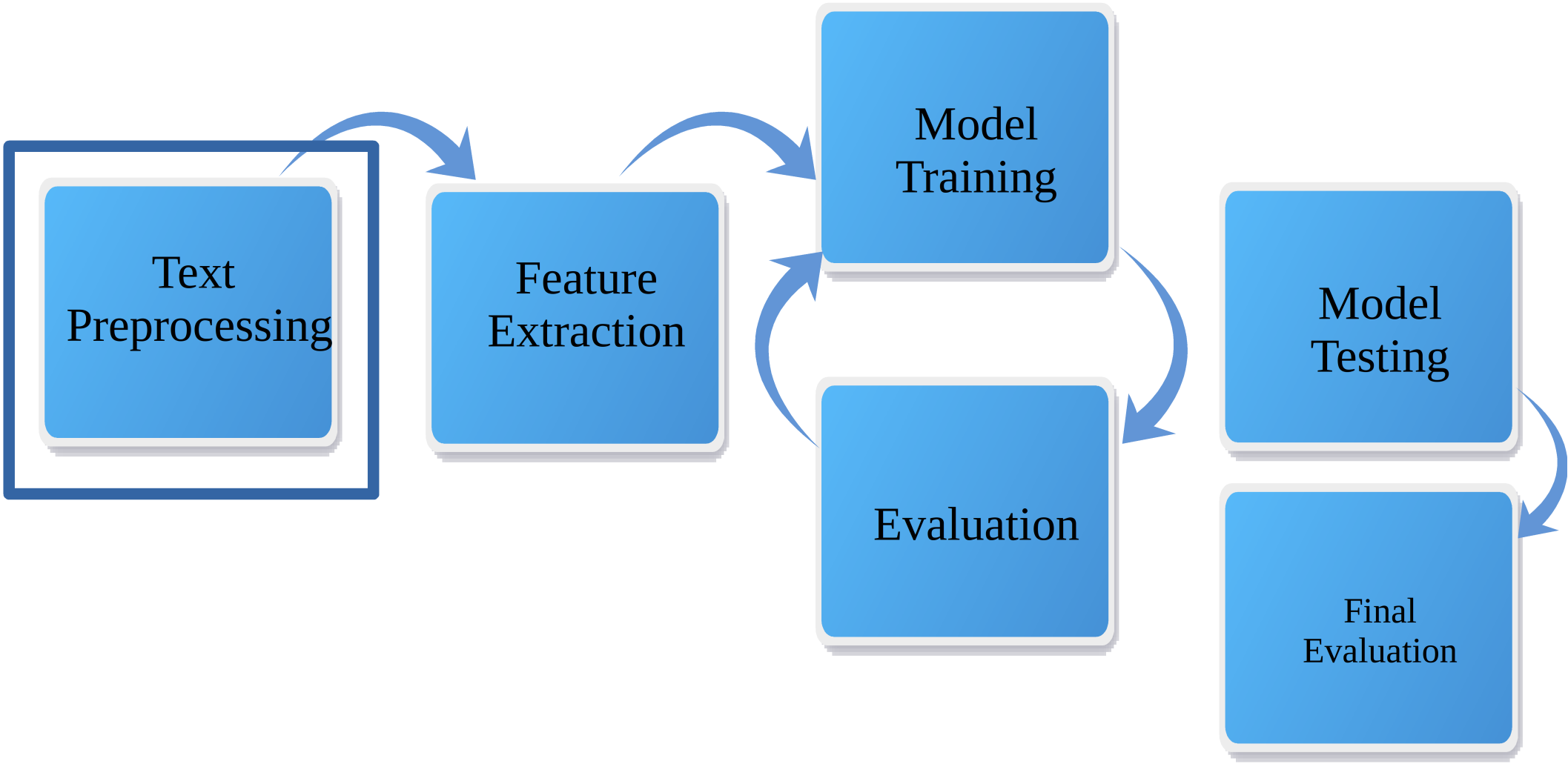
A.Tsakalidis@warwick.ac.uk

Exercise 2



A.Tsakalidis@warwick.ac.uk

Exercise 2

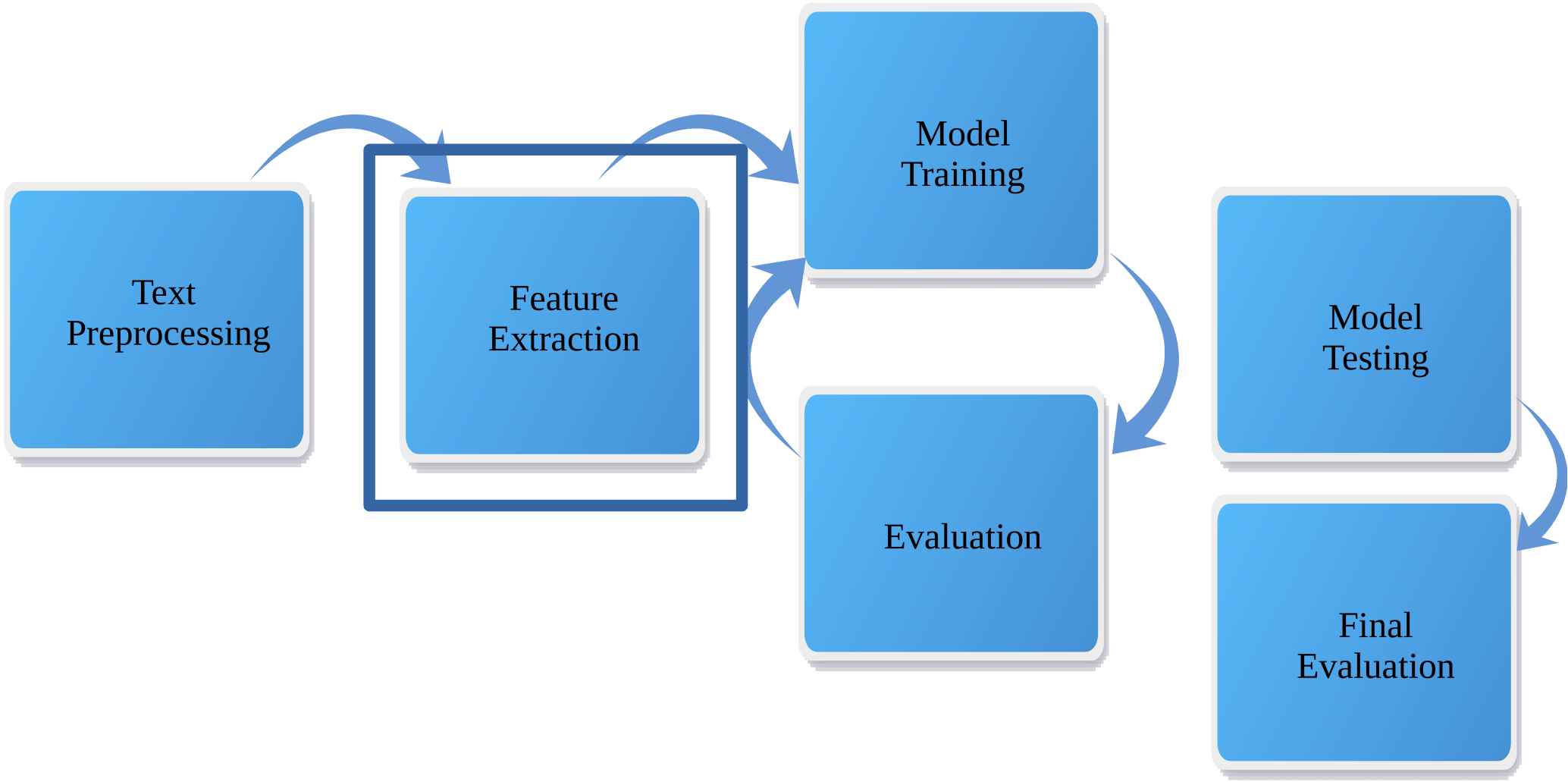


A.Tsakalidis@warwick.ac.uk

Step 1: Text preprocessing

- Recap (see previous seminar):
  - Lowercase (all-upper-case tokens might be important):
    - “This is NOT good” → “this is uppercasenot good”
  - Replace elongated words:
    - “This is so cooooooolllll” → “This is so cooolll”
  - Replace mentions/urls/etc. with identifiers:
    - “Listening to the new song by @user https://youtube.com”
    - “listening to the new song by usrmnt urlink”
  - Remove non-alphanumeric
  - Others: smileys replacement, stemming

Exercise 2



A.Tsakalidis@warwick.ac.uk

Step 2: Feature extraction

- Ngrams
  - {binary, tf, tfidf}, {POS}
- Sentiment Lexicons
  - Counts/summation of scores derived from lexicons
- Word Embeddings
  - Use pre-trained vectors
  - Use functions over vector dimensions
- Twitter-specific
  - Hashtags, uppercase words, tweet length...

A.Tsakalidis@warwick.ac.uk

Step 2: ngrams

- Doc#1 (train): “great piece of art have a look”
- Doc#2 (train): “omg this notdoes look great”
- Doc#3 (test): “omg is this real”
- Vocabulary (e.g., binary unigrams) based on the **training set**:

Doc	a	art	great	have	look	notdoes	of	omg	piece	this
Doc#1	1	1	1	1	1	0	1	0	1	0
Doc#2	0	0	1	0	1	1	0	1	0	1
Doc#3	0	0	0	0	0	0	0	1	0	1

- POS tagging: concatenate word+POS when building the vocabulary (e.g., “greatADJ”)
- Resources: sklearn (see the vectorisers: http://bit.ly/2FPfiCt)

A.Tsakalidis@warwick.ac.uk

Step 2: ngrams

- Doc#1 (train): “great piece of art have a look”
- Doc#2 (train): “omg this notdoes look great”
- Doc#3 (test): “omg is this real”
- Vocabulary (e.g., binary unigrams) based on the **training set**:

Doc	a	art	great	have	look	notdoes	of	omg	piece	this
Doc#1	1	1	1	1	1	0	1	0	1	0
Doc#2	0	0	1	0	1	1	0	1	0	1
Doc#3	0	0	0	0	0	0	0	1	0	1

- POS tagging: concatenate word+POS when building the vocabulary (e.g., “greatADJ”)
- Resources: sklearn (see the vectorisers: http://bit.ly/2FPfiCt)

Step 2: word embeddigs

- Inputs:
  - preprocessed tweet with *n* words;
  - word embeddings: {word→[vals]}
- Output: tweet with *n*-by-D scores (D: dimension of embeddings)
- **Features**: functions over each of the D resulting dimensions
  - average/max/min/multiplication/summation
  - each applied function generates D features for a tweet
- Resources (pre-trained word vectors):
  - Glove: https://nlp.stanford.edu/projects/glove/
  - word2vec: https://code.google.com/archive/p/word2vec/
  - Others: http://ir.hit.edu.cn/~dytang/

A.Tsakalidis@warwick.ac.uk

Step 2: sentiment lexicons

- Inputs:
  - preprocessed tweet with *n* words;
  - lexicon_1: {word→[vals]} (e.g., D-dimensional)
  - lexicon_2: {word→label} (e.g., binary)
  - …
- **Features**:
  - D-dimensional (lexicon_1): summation of each dimension
  - 2-dimensional (lexicon_2): number of (pos/neg) words
  - …
  - Use the concatenation of all of these features

A.Tsakalidis@warwick.ac.uk

Step 2: sentiment lexicons

- Resources:
  - http://sentiwordnet.isti.cnr.it/
  - https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html
  - http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/
  - http://www.wjh.harvard.edu/~inquirer/
  - http://liwc.wpengine.com/

- Beware: different feature type per lexicon!
  - Prefer counts for binary lexicons
  - Prefer overall (or per-class) sum for lexicons with real values

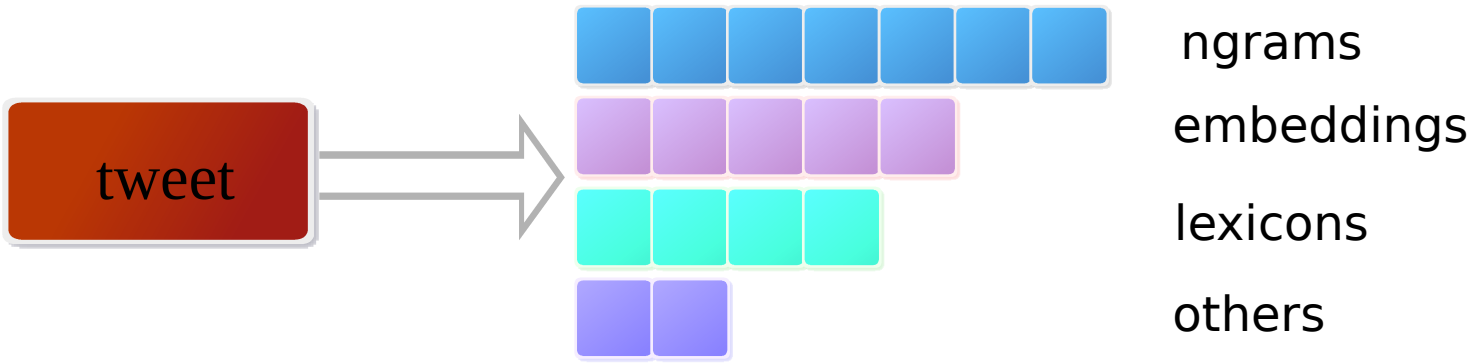
Step 2: other features

- Twitter-specific features:
  - #elongated words
  - #all-upper-case words
  - presence/absence of negation
  - presence/absence of URL (binary)
  - presence/absence of user mention (binary)
  - length of tweet (in chars or words)
  - #POS tags (e.g., #adjectives)
  - …
- Use the concatenation of all of these features

A.Tsakalidis@warwick.ac.uk

Step 2: Summary

- Extract different feature sets



- Use them individually or in concatenation
  - e.g., ngrams+embeddings+lexicons, lexicons+embeddings...
- Feed them into a classifier (see next)
- Validate on train/dev set; select best features/classifier
- If your features are bad, your model will be bad!

A.Tsakalidis@warwick.ac.uk

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

word	label
good	pos
bad	neg
awesome	pos
awful	neg

Lexicons

word	happy	sad	surprise	anger
soo	0.3	0.1	0.2	0.2
bad	0	0.9	0.2	0.4
awesome	0.9	0	0.3	0

Word Vectors

word	dim1	dim2	dim3
actually	0.1	0.2	0.1
awesome	0.3	0.2	0.3
bad	0.8	0.6	0.6
I	0.1	0.2	0.3
is	0.1	0.1	0.1
music	0.3	0.8	0.2
so	0.1	0.2	0.5
think	0.6	0.5	0.5
this	0.1	0.2	0.8

A.Tsakalidis@warwick.ac.uk

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

ngram features (assume binary unigrams/bigrams)?

A.Tsakalidis@warwick.ac.uk

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

Doc1 unigrams:

Doc1 bigrams:

A.Tsakalidis@warwick.ac.uk

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

Doc1 unigrams: {i, think, this, is, awesome, music, urlink}

Doc1 bigrams: {i think, think this, this is, is awesome, awesome music, music urlink}

A.Tsakalidis@warwick.ac.uk

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

Doc1 unigrams: {i, think, this, is, awesome, music, urlink}

Doc1 bigrams: {i think, think this, this is, is awesome, awesome music, music urlink}

|V|=13

	i	think	this	is	awesome	music	urlink	i think	think this	this is	is awesome	awesome music
#1	1	1	1	1	1	1	1	1	1	1	1	1
#2												

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

Doc1 unigrams: {i, think, this, is, awesome, music, urlink}

Doc1 bigrams: {i think, think this, this is, is awesome, awesome music, music urlink}

|V|=13

	i	think	this	is	awesome	music	urlink	i think	think this	this is	is awesome	awesome music
#1	1	1	1	1	1	1	1	1	1	1	1	1
#2			1	1						1		

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

word	label
good	pos
bad	neg
awesome	pos
awful	neg

word	happy	sad	surprise	anger
soo	0.3	0.1	0.2	0.2
bad	0	0.9	0.2	0.4
awesome	0.9	0	0.3	0

Which lexicon features would you use?

A.Tsakalidis@warwick.ac.uk

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

word	label
good	pos
bad	neg
awesome	pos
awful	neg

word	happy	sad	surprise	anger
soo	0.3	0.1	0.2	0.2
bad	0	0.9	0.2	0.4
awesome	0.9	0	0.3	0

	?	?	?	?	?	?
#1	?	?	?	?	?	?
#2	?	?	?	?	?	?

A.Tsakalidis@warwick.ac.uk

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

word	label
good	pos
bad	neg
awesome	pos
awful	neg

word	happy	sad	surprise	anger
soo	0.3	0.1	0.2	0.2
bad	0	0.9	0.2	0.4
awesome	0.9	0	0.3	0

	pos	neg	happy	sad	surprise	anger
#1						
#2						

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

word	label
good	pos
bad	neg
awesome	pos
awful	neg

word	happy	sad	surprise	anger
soo	0.3	0.1	0.2	0.2
bad	0	0.9	0.2	0.4
awesome	0.9	0	0.3	0

	pos	neg	happy	sad	surprise	anger
#1	1	0	0.9	0.0	0.3	0.0
#2	0	1	0.3	1.0	0.4	0.6

A.Tsakalidis@warwick.ac.uk

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

word	dim1	dim2	dim3
actually	0.1	0.2	0.1
awesome	0.3	0.2	0.3
bad	0.8	0.6	0.6
I	0.1	0.2	0.3
is	0.1	0.1	0.1
music	0.3	0.8	0.2
so	0.1	0.2	0.5
think	0.6	0.5	0.5
this	0.1	0.2	0.8

Let's extract some word embeddings features...

(Remember: functions over dimensions)

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

word	dim1	dim2	dim3
actually	0.1	0.2	0.1
awesome	0.3	0.2	0.3
bad	0.8	0.6	0.6
I	0.1	0.2	0.3
is	0.1	0.1	0.1
music	0.3	0.8	0.2
so	0.1	0.2	0.5
think	0.6	0.5	0.5
this	0.1	0.2	0.8

Doc#1	Dim1	Dim2	Dim3
I	0.1	0.2	0.3
think	0.6	0.5	0.5
this	0.1	0.2	0.8
is	0.1	0.1	0.1
awesome	0.3	0.2	0.3
music	0.3	0.8	0.2
urlink	–	–	–

A.Tsakalidis@warwick.ac.uk

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

word	dim1	dim2	dim3
actually	0.1	0.2	0.1
awesome	0.3	0.2	0.3
bad	0.8	0.6	0.6
I	0.1	0.2	0.3
is	0.1	0.1	0.1
music	0.3	0.8	0.2
so	0.1	0.2	0.5
think	0.6	0.5	0.5
this	0.1	0.2	0.8

Doc#1	Dim1	Dim2	Dim3
I	0.1	0.2	0.3
think	0.6	0.5	0.5
this	0.1	0.2	0.8
is	0.1	0.1	0.1
awesome	0.3	0.2	0.3
music	0.3	0.8	0.2
urlink	–	–	–
MIN			
SUM			

A.Tsakalidis@warwick.ac.uk

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

word	dim1	dim2	dim3
actually	0.1	0.2	0.1
awesome	0.3	0.2	0.3
bad	0.8	0.6	0.6
I	0.1	0.2	0.3
is	0.1	0.1	0.1
music	0.3	0.8	0.2
so	0.1	0.2	0.5
think	0.6	0.5	0.5
this	0.1	0.2	0.8

Doc#1	Dim1	Dim2	Dim3
I	0.1	0.2	0.3
think	0.6	0.5	0.5
this	0.1	0.2	0.8
is	0.1	0.1	0.1
awesome	0.3	0.2	0.3
music	0.3	0.8	0.2
urlink	–	–	–
MIN	**0.1**	**0.1**	**0.1**
SUM	**1.5**	**2.0**	**2.2**

A.Tsakalidis@warwick.ac.uk

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

word	dim1	dim2	dim3
actually	0.1	0.2	0.1
awesome	0.3	0.2	0.3
bad	0.8	0.6	0.6
I	0.1	0.2	0.3
is	0.1	0.1	0.1
music	0.3	0.8	0.2
so	0.1	0.2	0.5
think	0.6	0.5	0.5
this	0.1	0.2	0.8

#1	0.1	0.1	0.1	1.5	2.0	2.2
#2	0.1	0.1	0.1	1.1	1.1	1.6

A.Tsakalidis@warwick.ac.uk

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

word	dim1	dim2	dim3
actually	0.1	0.2	0.1
awesome	0.3	0.2	0.3
bad	0.8	0.6	0.6
I	0.1	0.2	0.3
is	0.1	0.1	0.1
music	0.3	0.8	0.2
so	0.1	0.2	0.5
think	0.6	0.5	0.5
this	0.1	0.2	0.8

#1	0.1	0.1	0.1	1.5	2.0	2.2
#2	0.1	0.1	0.1	1.1	1.1	1.6

Any other features?

A.Tsakalidis@warwick.ac.uk

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

	length	link	mention	exclmtn	negation	smiley	frown	elong
#1	7	1	0	1	0	0	0	0
#2	6	0	1	0	0	0	0	1

A.Tsakalidis@warwick.ac.uk

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	0	0	0	0	0	1	0	0

1	0	.9	0	.3	0
0	1	.3	1	.4	.6

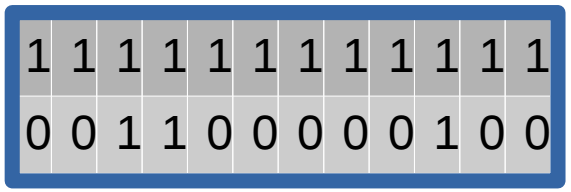
.1	.1	.1	1.5	2.0	2.2
.1	.1	.1	1.1	1.1	1.6

7	1	0	1	0	0	0	0
6	0	1	0	0	0	0	1

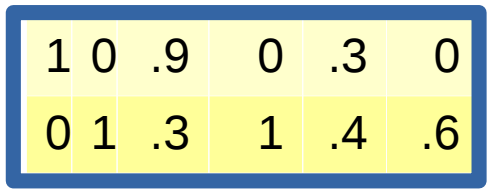
Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

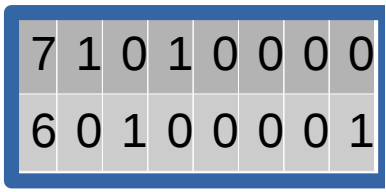


ngrams



lexicons

embeddings



others

Use these feature sources individually

A.Tsakalidis@warwick.ac.uk

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

1	1	1	1	1	1	1	1	1	1	1	1	1	0	.9	0	.3	0
0	0	1	1	0	0	0	0	0	1	0	0	0	1	.3	1	.4	.6

ngrams + lexicons

.1	.1	.1	1.5	2.0	2.2
.1	.1	.1	1.1	1.1	1.6

7	1	0	1	0	0	0	0
6	0	1	0	0	0	0	1

Use these feature sources individually

or concatenated

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	0	0	0	0	0	1	0	0

1	0	.9	0	.3	0
0	1	.3	1	.4	.6

.1	.1	.1	1.5	2.0	2.2
.1	.1	.1	1.1	1.1	1.6

7	1	0	1	0	0	0	0
6	0	1	0	0	0	0	1

lexicons + embeddings + others

Use these feature sources individually

or concatenated

Step 2: example

"I think this is AWESOME! #music http://url.com"

"@user123 This is actually soooo bad..."

1	1	1	1	1	1	1	1	1	1	1	1	1	0	.9	0	.3	0	.1	.1	.1	1.5	2.0	2.2	7	1	0	1	0	0	0	0
0	0	1	1	0	0	0	0	0	1	0	0	0	1	.3	1	.4	.6	.1	.1	.1	1.1	1.1	1.6	6	0	1	0	0	0	0	1

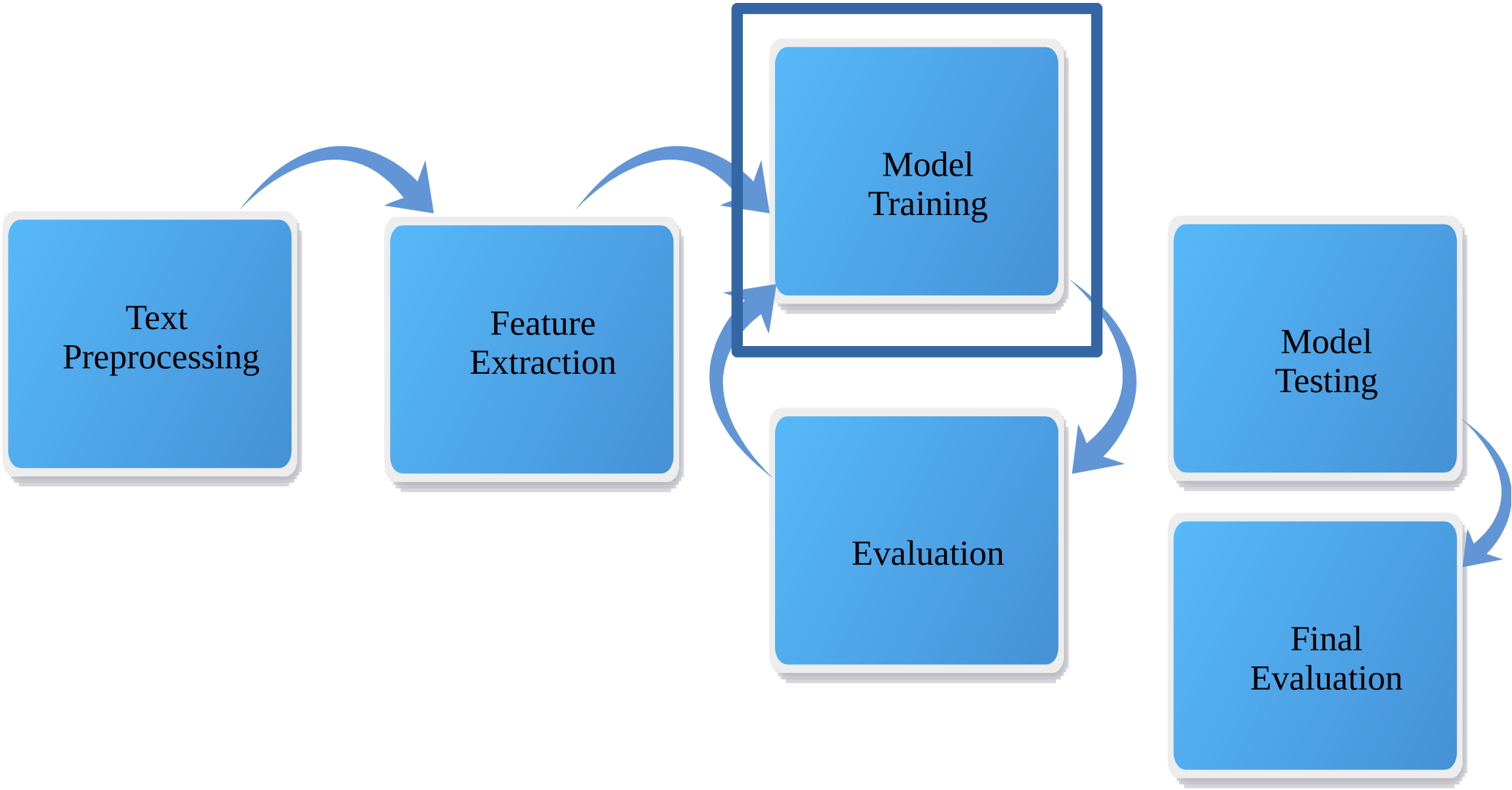
ngrams + lexicons + embeddings + others

Use these feature sources individually

or concatenated

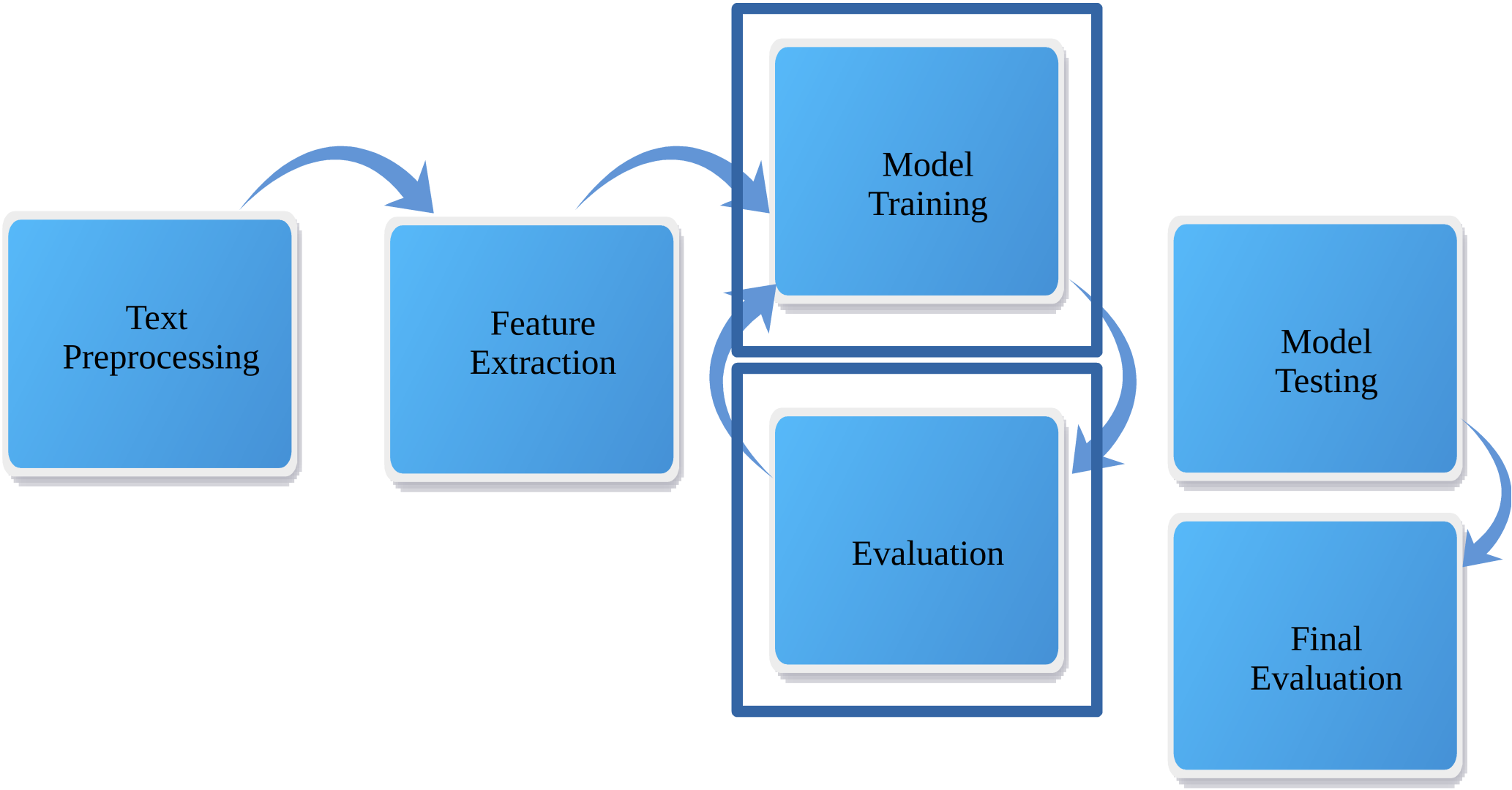
A.Tsakalidis@warwick.ac.uk

Exercise 2



A.Tsakalidis@warwick.ac.uk

Exercise 2



A.Tsakalidis@warwick.ac.uk

Step 3: training

- Start with simple models
  - Support Vector Machine
  - Random Forest
  - Logistic Regression
  - Naive Bayes

- Recommended:
  - sklearn (http://scikit-learn.org/stable/)

- Proceed with deep learning – if time allows to do so

A.Tsakalidis@warwick.ac.uk

Step 3: training

- Train different classifiers on different feature sets
- Report macro-average f-score (pos/neg), e.g.:

	ngram	embed	lex	ngram + embed	ngram + lex	embed + lex	ngram + other	lex + other	all
Majority									
SVM									
RF									
LR									

- Questions:
  - How do you select the best parameters per model?
  - How do you select the best model?

A.Tsakalidis@warwick.ac.uk

Step 3: tuning

- Train all models on the training set
  - Tune parameters: 10-fold cross-validation
    - Regularisers, kernel selection for SVM, number of trees in RF...
    - Hint: GridSearchCV in sklearn
  - Select final feature sets to use on test set

- What is the development set for?
  - Parameter tuning (instead of CV)
  - Accuracy estimation (pseudo-test set)
  - More data for training

A.Tsakalidis@warwick.ac.uk

Step 3: tuning dangers

- Overfitting
  - You do NOT know the class distribution of the test sets
    - It might be very different compared to your training data
  - Do NOT try to overfit your models
    - a small accuracy boost in 10-fold CV/dev set might result into a large decrease in the test set!

- Evaluation: macro-avg f-score of positive/negative classes
  - Do NOT ignore the neutral class!

    a) it affects the macro-avg pos/neg performance anyway

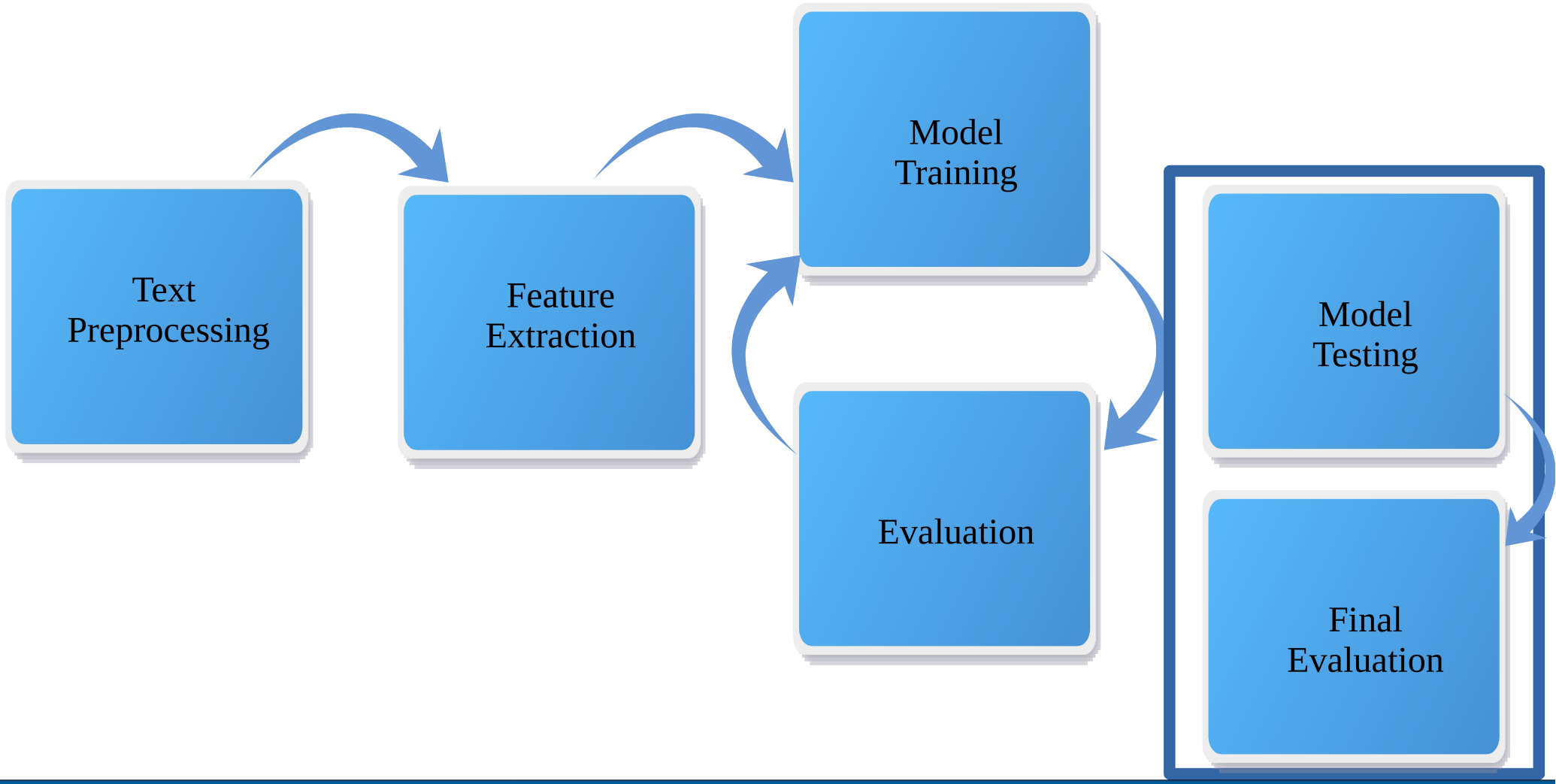
    b) what if the test set has much more/less neutral tweets...?

Step 4: evaluation

- Selected best features/combinations of features?
- Performance consistent across folds/sets?
- Different classifiers performing better at different classes?
  - Ensemble...?
- Error anaysis
  - Common characteristics of missclassified tweets?
  - Confusion matrix: frequently missclassified labels?
- Select best model/parameters/features for the test set
  - Retrain model on training (+dev?) set; only one shot

A.Tsakalidis@warwick.ac.uk

Exercise 2



A.Tsakalidis@warwick.ac.uk

Step 5: testing

- Exactly the same pre-processing steps
- Exactly the same features as in the training set
  - do NOT add ngrams you have not seen in the training set
- Simply apply the best model to the test sets
- Make sure you include the results in your submission
- Include detailed evaluation on the three test sets
- Do NOT try to overfit!
  - We will be running your code on different test sets!

A.Tsakalidis@warwick.ac.uk

Summary

- Text Preprocessing
  - Dealing with: user mentions, hashtags, URLS, non-alphanumeric chars, elongated words, all upper-case words, emoticons...
- Feature Extraction
  - ngrams, lexicons, word embeddings, twitter-specific
- Training/Tuning/Evaluation Process
  - Train different models on different combinations of feature sets
  - **Evaluate!** Error analysis, detailed evaluation, argue about your final choice!
- Testing Process
  - Apply model to test sets

A.Tsakalidis@warwick.ac.uk

Datasets and Marking

- Training Set: Extract features, train classifiers, evaluate using 10-fold CV
- Dev Set: use for *(a)* for training, along with Training Set; *(b)* tuning parameters; or *(c)* for testing (getting an estimation of performance)
- Test Set: Extract the same features & apply your best model

- **Marking** (see the assignment for details!):
  - Make sure your **code** runs (+on different test sets) (20/85)
  - Write a descent **report** (20/85)
  - **Performance** on 5 test sets (20/85)
  - **Feature extraction** approach (15/85)
  - **Methodological** novelty (10/85)

A.Tsakalidis@warwick.ac.uk

#thank @you :)

Any questions?

A.Tsakalidis@warwick.ac.uk