Feature extraction from texts and images

Solely text/images competitions

Feature extraction from texts and images

Common features + text

Titanic dataset

A1 | survived

	A	B	C	D	E	F
1	survived	pclass	name	sex	age	sibsp
2	0	3	Braund, Mr. Owen Harris	male	22	1
3	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1
4	1	3	Heikkinen, Miss. Laina	female	26	0
5	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1
6	0	3	Allen, Mr. William Henry	male	35	0
7	0	3	Moran, Mr. James	male		0
8	0	1	McCarthy, Mr. Timothy J	male	54	0
9	0	3	Palsson, Master. Gosta Leonard	male	2	3
10	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0
11	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1
12	1	3	Sandstrom, Miss. Marguerite Rut	female	4	1
13	1	1	Bonnell, Miss. Elizabeth	female	58	0

Feature extraction from texts and images

Common features + images/text

TRADESHIFT®

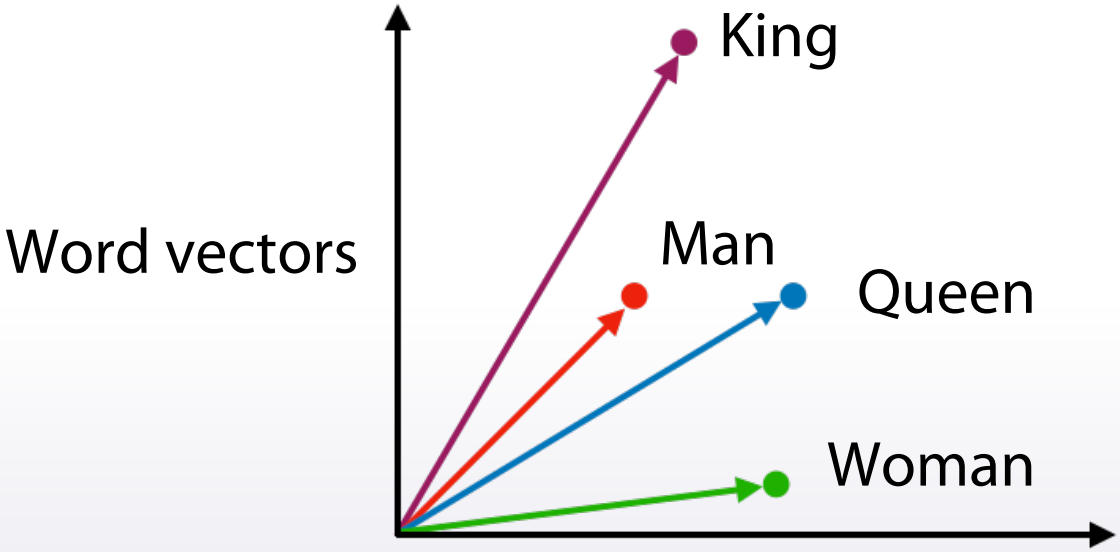
Text -> vector

1. Bag of words:

The dog is on the table

0	0	1	1	0	1	1	1
are	cat	dog	is	now	on	table	the

2. Embeddings (~word2vec):



Bag of words

(excited) Hi everyone!	I'm so excited about this course!	So excited. SO EXCITED. EXCITED, I AM!

CountVectorizer

hi	every one	I'm	so	excited	about	this	course
1	1			1			
		1	1	1	1	1	1
		1	2	3			

`sklearn.feature_extraction.text.CountVectorizer`

Bag of words: TFiDF

Bag of words: TFiDF

Term frequency

```
tf = 1 / x.sum(axis=1) [:,None]
x = x * tf
```

Bag of words: TFiDF

Term frequency

```
tf = 1 / x.sum(axis=1) [:,None]
x = x * tf
```

Inverse Document Frequency

```
idf = np.log(x.shape[0] / (x > 0).sum(0))
x = x * idf
```

Bag of words: TFiDF

Term frequency

tf = 1 / x.sum(axis=1) [:,None]

x = x * tf

Inverse Document Frequency

idf = np.log(x.shape[0] / (x > 0).sum(0))

x = x * idf

`sklearn.feature_extraction.text.TfidfVectorizer`

Bag of words: TF

(excited) Hi everyone!	I'm so excited about this course!	So excited. SO EXCITED. EXCITED, I AM!

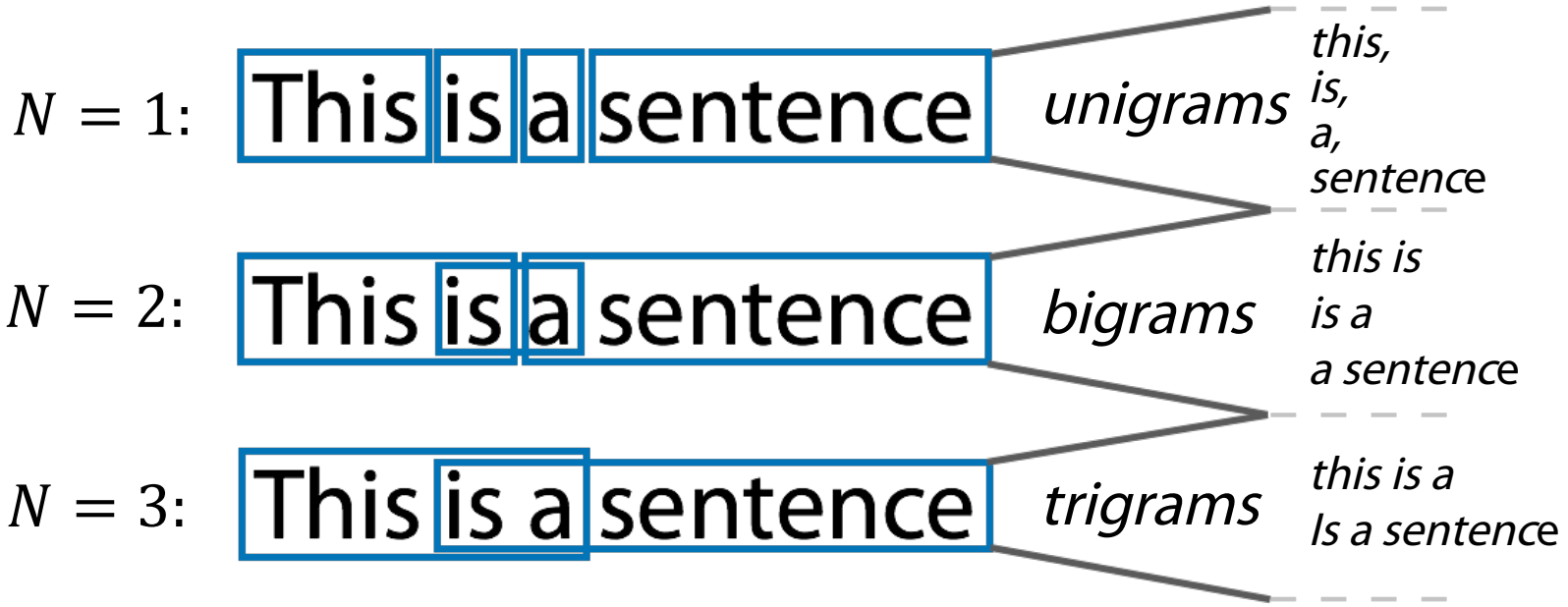
hi	every one	I'm	so	excited	about	this	course
0.33	0.33			0.33			
		0.16	0.16	0.16	0.16	0.16	0.16
		0.16	0.33	0.5			

Bag of words: TF+iDF

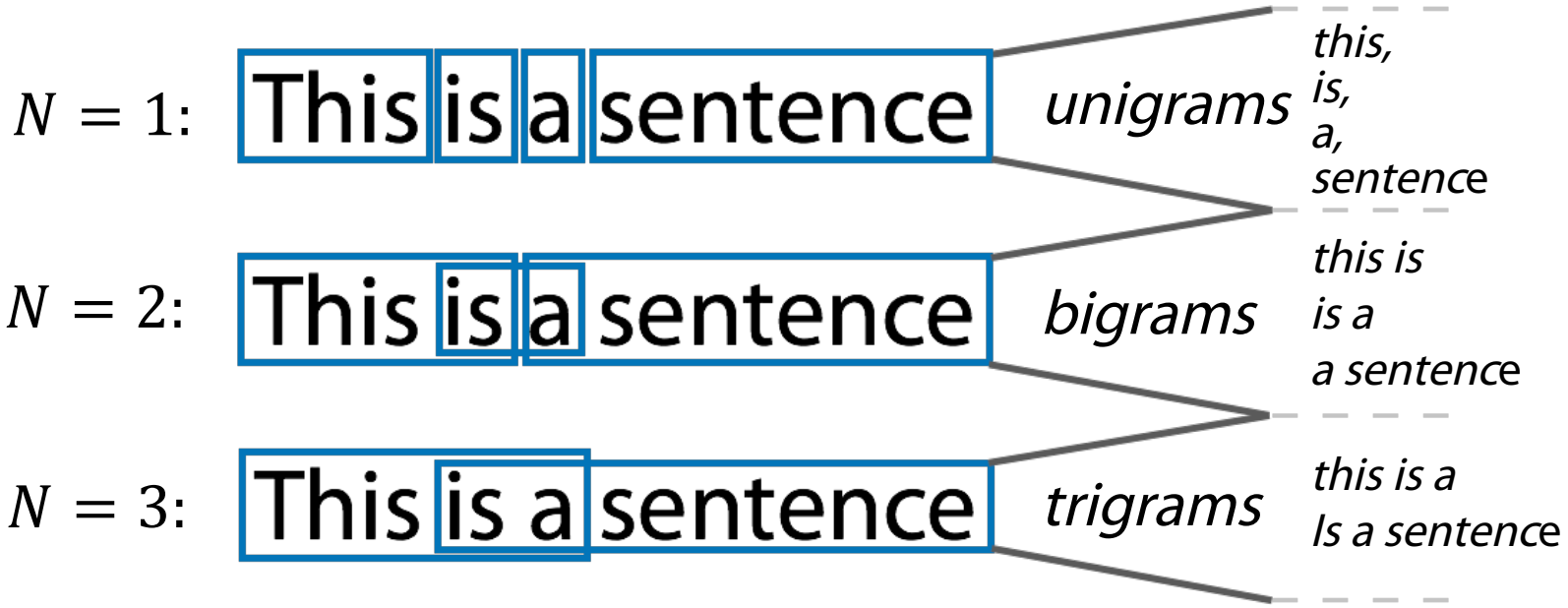
(excited) Hi everyone!	I'm so excited about this course!	So excited. SO EXCITED. EXCITED, I AM!

hi	every one	I'm	so	excited	about	this	course
0.36	0.36			0			
		0.06	0.06	0	0.18	0.18	0.18
		0.06	0.13	0			

N-grams


$N = 1$:
This is a sentence
unigrams
this,
is,
a,
sentence
$N = 2$:
This is a sentence
bigrams
this is
is a
a sentence
$N = 3$:
This is a sentence
trigrams
this is a
Is a sentence

N-grams



`sklearn.feature_extraction.text.CountVectorizer:`
`Ngram_range, analyzer`

Texts preprocessing

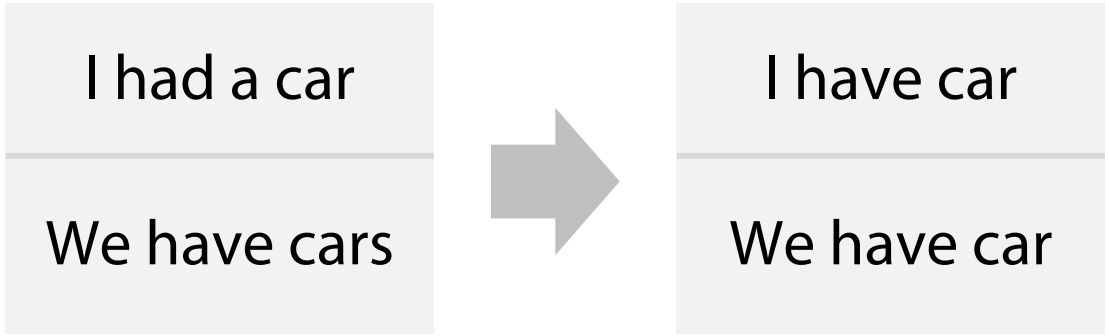
1. Lowercase
2. Lemmatization
3. Stemming
4. Stopwords

Texts preprocessing: lowercase

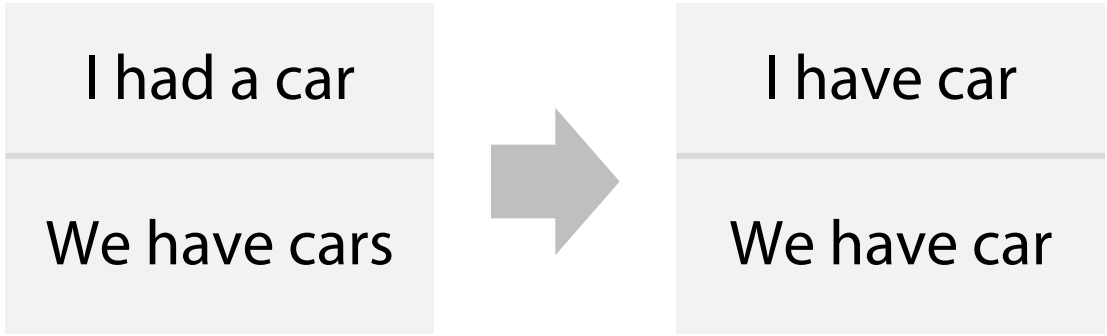
Very, very sunny.
Sunny... Sunny!

Very	very	Sunny	sunny
1	1	0	1
0	0	2	0

Texts preprocessing: lemmatization and stemming



Texts preprocessing: lemmatization and stemming



Stemming:
democracy, democratic, and democratization -> democr

Lemmatization:
democracy, democratic, and democratization -> democracy

Texts preprocessing: lemmatization and stemming

I had a car	I have car
We have cars	We have car

Stemming:
democracy, democratic, and democratization -> democr

Saw -> s

Lemmatization:
democracy, democratic, and democratization -> democracy

Saw -> see or saw (depending on context)

Texts preprocessing: stopwords

Examples:

1. Articles or prepositions
2. Very common words

Texts preprocessing: stopwords

Examples:

1. Articles or prepositions
2. Very common words

NLTK, Natural Language Toolkit library for python

`sklearn.feature_extraction.text.CountVectorizer: max_df`

Conclusion

Pipeline of applying BOW

1. Preprocessing:

   Lowercase, stemming, lemmatization, stopwords

2. Bag of words:

   Ngrams can help to use local context

3. Postprocessing: TFiDF