Text Analytics Fundamentals

Data Science Dojo

Structured vs. Unstructured Data

- Structured – Tabular data
- Semi-structured – Non-tabular data with some meta-data
  - Ex: JSON, XML
- Unstructured – Non-tabular data with no meta-data

Structured – tabular data

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
1	1	0	3	Braund, Mr. Owen Harris	male	22.00	1	0	A/5 21171	7.2500	
2	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38.00	1	0	PC 17599	71.2833	C85
3	3	1	3	Heikkinen, Miss. Laina	female	26.00	0	0	STON/O2. 3101282	7.9250	
4	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.00	1	0	113803	53.1000	C123
5	5	0	3	Allen, Mr. William Henry	male	35.00	0	0	373450	8.0500	
6	6	0	3	Moran, Mr. James	male	NA	0	0	330877	8.4583	
7	7	0	1	McCarthy, Mr. Timothy J	male	54.00	0	0	17463	51.8625	E46
8	8	0	3	Palsson, Master. Gosta Leonard	male	2.00	3	1	349909	21.0750	
9	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.00	0	2	347742	11.1333	
10	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.00	1	0	237736	30.0708	
11	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.00	1	1	PP 9549	16.7000	G6
12	12	1	1	Bonnell, Miss. Elizabeth	female	58.00	0	0	113783	26.5500	C103
13	13	0	3	Saundercock, Mr. William Henry	male	20.00	0	0	A/5. 2151	8.0500	
14	14	0	3	Andersson, Mr. Anders Johan	male	39.00	1	5	347082	31.2750	
15	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14.00	0	0	350406	7.8542	

Semi-structured data

```
<html>
<head>
<title>CSS Experiments</title>
<link rel="stylesheet" href="styles.css" type="text/css" media="all">
</head>
<body>
<div id="menu">
<ul>
    <li><a href="http://abduzeedo.com/">Home</a></li>
    <li><a href="http://abduzeedo.com/tutorials">Tutorials</a></li>
    <li><a href="http://abduzeedo.com/tags/interview">Interviews</a></li>
    <li><a href="http://abduzeedo.com/tags/wallpaper">Wallpapers</a></li>
</ul>
    <input type="" name="" value="" />
    </div>
    <div id="flickr_badge_uber_wrapper">
        <div id="flickr_badge_wrapper">
            <script type="text/javascript" src="http://www.flickr.com/
              badge_code_v2.gne?
              count=12&display=latest&size=s&layout=x&source=user_set&user=764
              66518%40N00&set=72157604672645588&context=in
              %2Fset-72157604672645588%2F"></script>
        </div>
    </div>

</body>
</html>
```

Unstructured data

TIME @TIME · 52s

An earlier version of this story incorrectly stated that the National Weather Service mistakenly sent a tsunami warning to phones. The warning was sent by third-party weather apps, not by the National Weather Service. The tweet was since deleted

A Tsunami Warning Blared on Phones Across the Country This Morni…

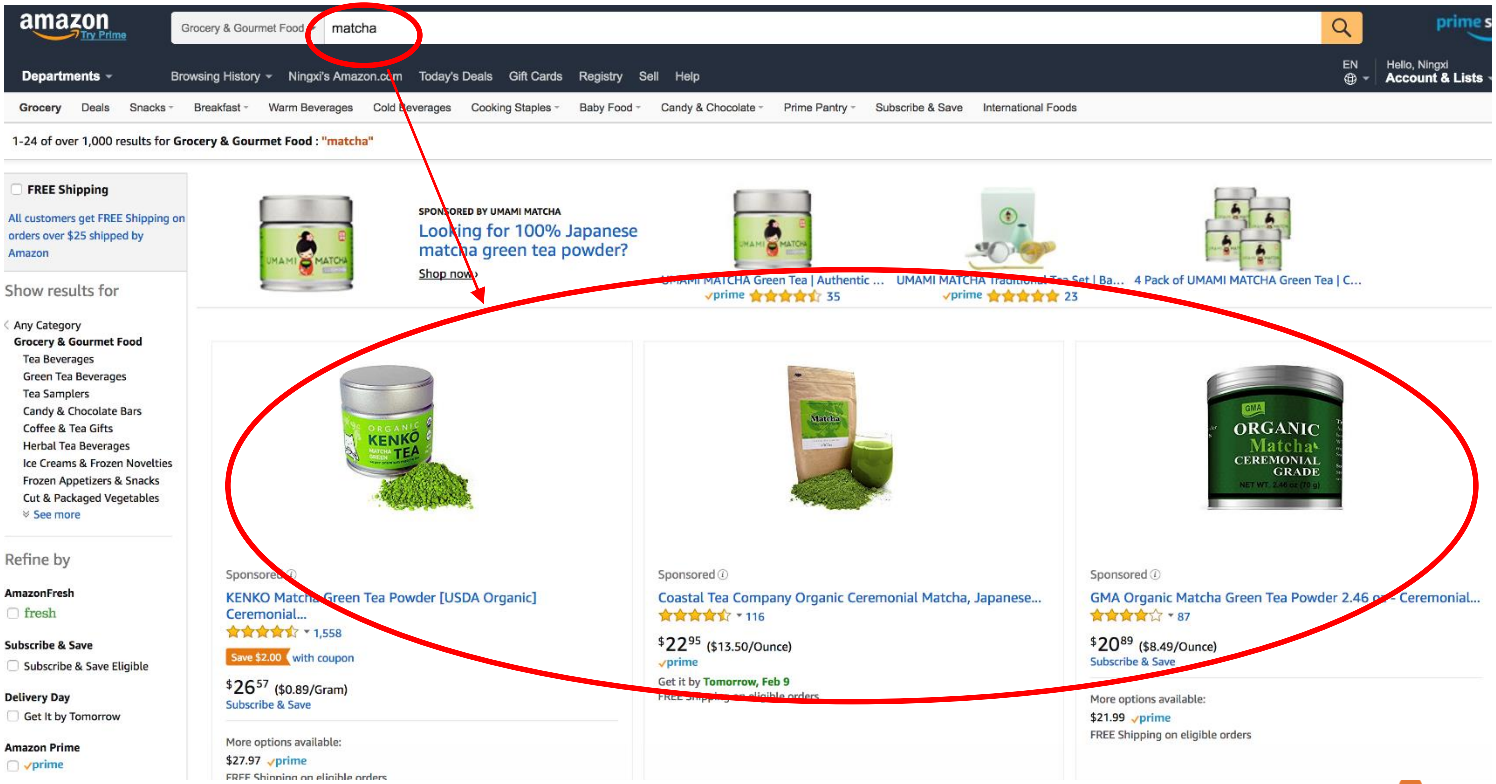
"Please note there is NO TSUNAMI THREAT"

time.com

Text Analytics in Business

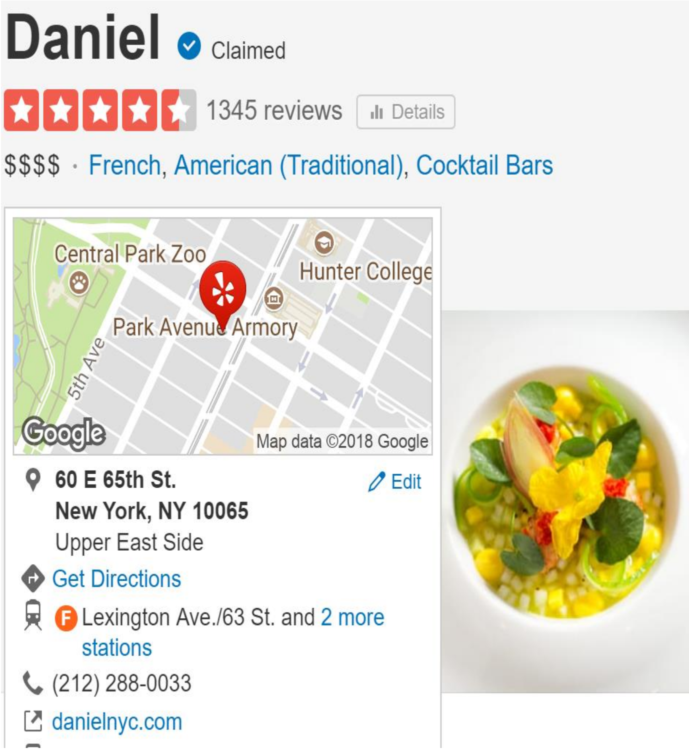
- **Information Retrieval (IR)**
  - Find documents which match a query
- **Sentiment Analysis**
  - Determine "emotion" of document based on certain words/terms appearing in the document
- **Recommendation Engines (Similarity)**
  - Recommend entities based on certain attributes
- **Topic Modelling**
  - Reduce document to topics

Information Retrieval



datasciencedojo
data science for everyone

Sentiment Analysis



1/5/2018

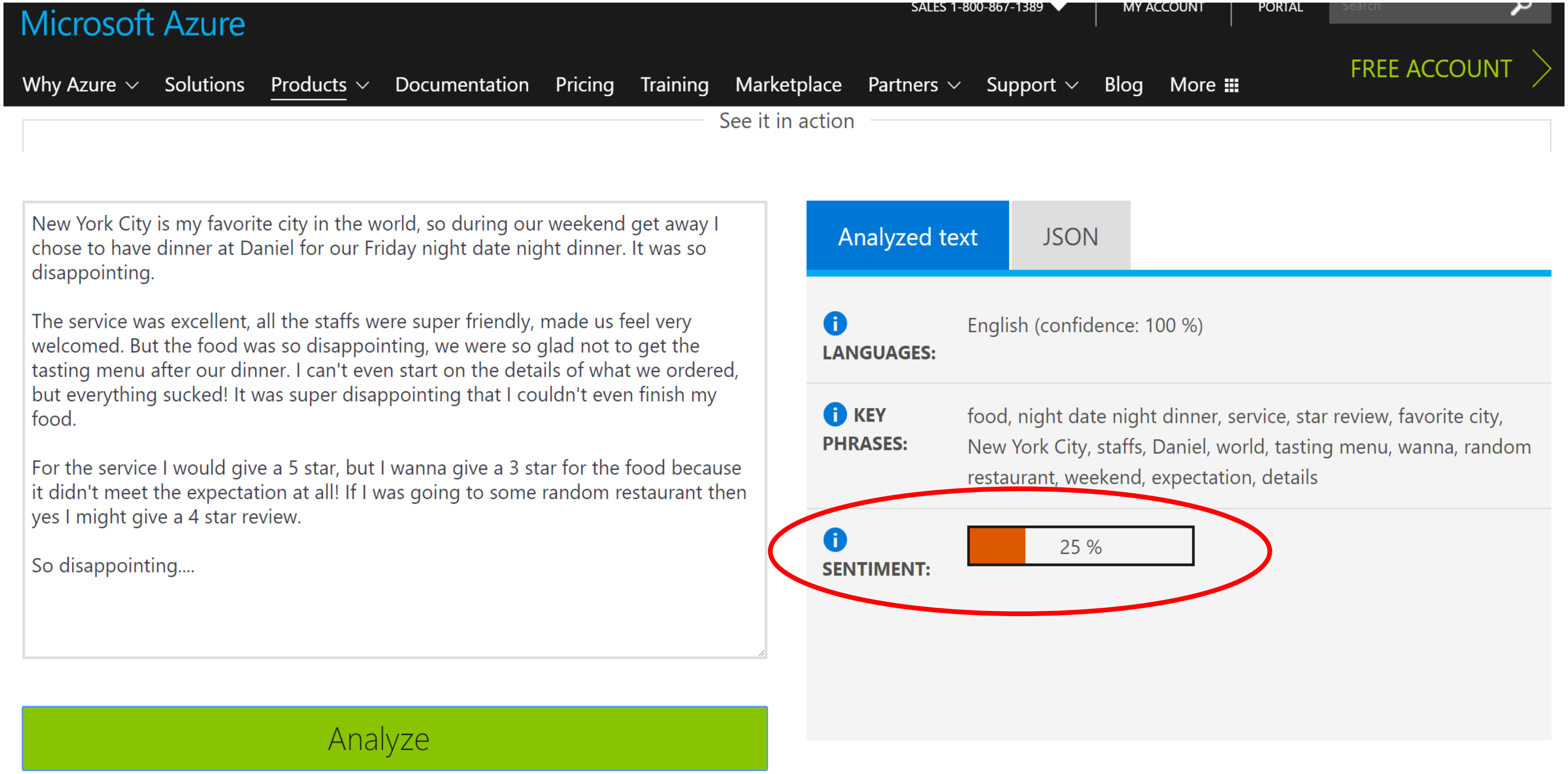
1 check-in

New York City is my favorite city in the world, so during our weekend get away I chose to have dinner at Daniel for our Friday night date night dinner. It was so disappointing.

The service was excellent, all the staffs were super friendly, made us feel very welcomed. But the food was so disappointing, we were so glad not to get the tasting menu after our dinner. I can't even start on the details of what we ordered, but everything sucked! It was super disappointing that I couldn't even finish my food.

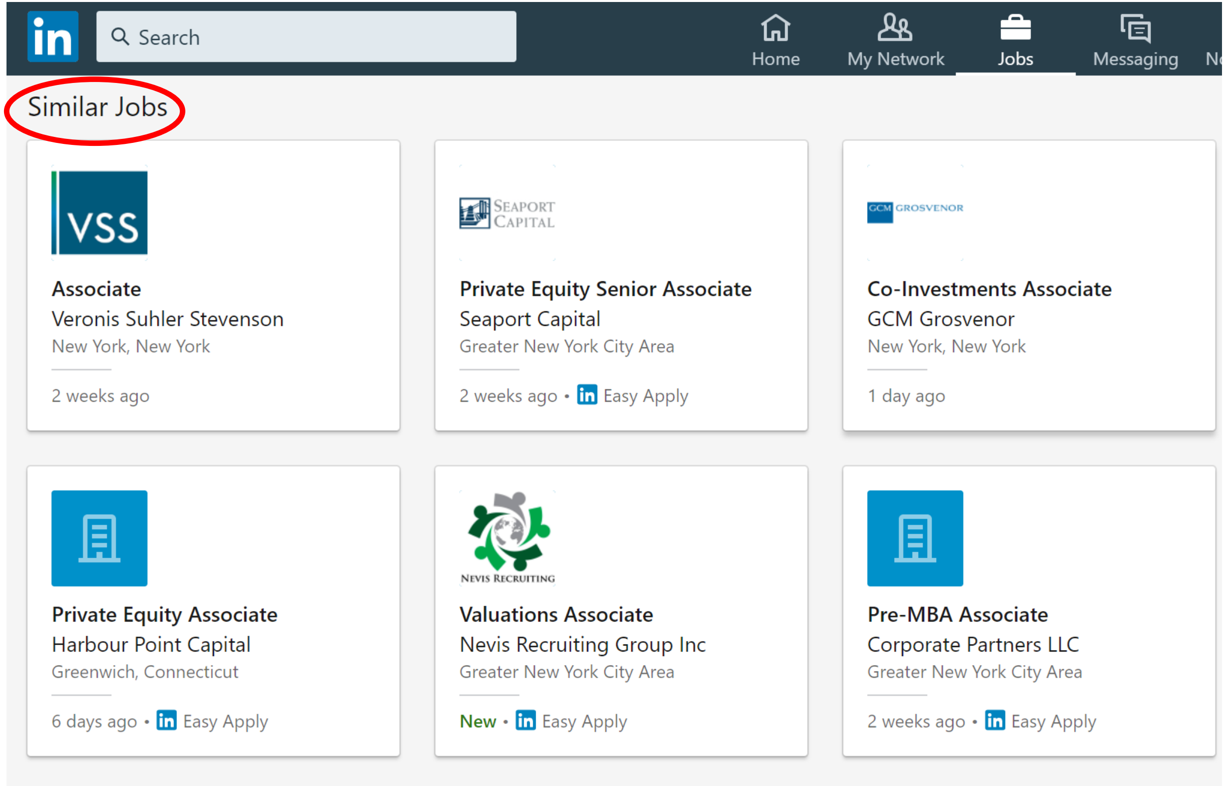
For the service I would give a 5 star, but I wanna give a 3 star for the food because it didn't meet the expectation at all! If I was going to some random restaurant then yes I might give a 4 star review.

So disappointing....



datasciencedojo
data science for everyone

Recommendation Engines



"Associate" appears in all postings, and all postings share words that may be related ("private equity," "investment," "valuations," "MBA," "capital," etc)

Topic Modelling

The New York Times

music	book	art	game	show
band	life	museum	knicks	film
songs	novel	show	nets	television
rock	story	exhibition	points	movie
album	books	artist	team	series
jazz	man	artists	season	says
pop	stories	paintings	play	life
song	love	painting	games	man
singer	children	century	night	character
night	family	works	coach	know

theater	clinton	stock	restaurant	budget
play	bush	market	sauce	tax
production	campaign	percent	menu	governor
show	gore	fund	food	county
stage	political	investors	dishes	mayor
street	republican	funds	street	billion
broadway	dole	companies	dining	taxes
director	presidential	stocks	dinner	plan
musical	senator	investment	chicken	legislature
directed	house	trading	served	fiscal

Text Analytics Fundamentals

- **Token:** A specific word in the document
- **Term**: The version of a word set that is in the dictionary
- **Corpus**: All of the documents.

Text Analytics Fundamentals

- How do we turn unstructured data into structured data?
  - Create columns based on document content
  - Each **term** in document creates a column
    - Column types: binary, word count, TF-IDF
  - Do we want to count every word?
    - Stop words
    - Stemming and lemmatization

Term – Dictionary Example

unstructured text data

CFA Institute®
@CFAinstitute
Following

"You are not a robo-adviser," says @meirstatman, "Your advantage is not in beating the market . . . Your advantage is in creating this bond, this emotional bond, with your clients," via @laurenfosternyc

pre-processing

lower case

remove stop words, punctuation, etc

stemming

build dictionary

document

token
robo-adviser
advantage
beating
market
creating
bond
emotional
clients

dictionary

term
robo-adviser
advantage
beat
market
creat
bond
emotion
client

datasciencedojo
data science for everyone

Stemming & Lemmatization

- **Stemming**: Convert tokens to terms by removing letters via heuristic
  - Both simple (Levins) and complex (Porter)

- **Lemmatization**: Classify tokens into terms using a linguistic analysis
  - **Lemma**: the base (dictionary) form of a word
  - Can be done using dictionary look-up, machine learning on annotated corpus

Stemming / Lemmatizing Example

Token	Stemmed term	Lemmatized term
Stemming is funnier than lemmatizing says the Barcelona loving data scientists	Stem is funnier than lemmas say the Barcelona love data scientist	stem be funny than lemmatizing say the barcelona love data scientist

Document Vectorization

Terms in the documents

	team	coach	play	ball	score	game	win	lost	timeout	season
d_1	3	0	5	0	2	6	0	2	0	2
d_2	0	7	0	2	1	0	0	3	0	0
d_3	0	1	0	0	1	2	2	0	3	0

Documents 1 to 3

dictionary

term
team
coach
play
ball
score
game
win
lost
timeout
season

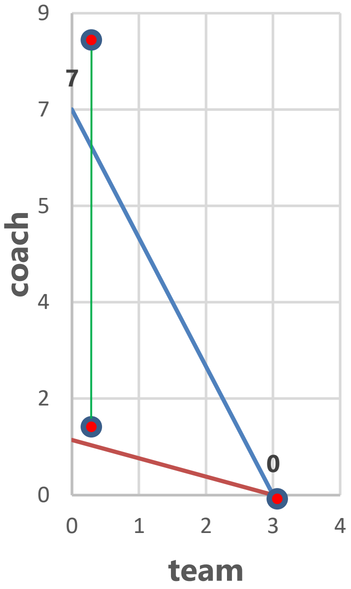
Document Vectorization

- Each document becomes a vector
- Allows use of numeric analysis

	team	coach	play	ball	score	game	win	lost	timeout	season
d_1	3	0	5	0	2	6	0	2	0	2
d_2	0	7	0	2	1	0	0	3	0	0
d_3	0	1	0	0	1	2	2	0	3	0

Document Similarity Measure

	Team	Coach
d_1	3	0
d_2	0	7
d_3	0	1



Distance between documents is calculated as:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Binary Document Vectorization

- Each document has a 1 if the word appears in it and a 0 if not

	team	coach	play	ball	score	game	win	lost	timeout	season
d_1	1	0	1	0	1	1	0	1	0	1
d_2	0	1	0	1	1	0	0	1	0	0
d_3	0	1	0	0	1	1	1	0	1	0

Drawbacks of Vectorization

- Not every word has similar importance
- Longer documents have a higher chance to have random unimportant words

TF-IDF

- Calculates term importance based on its occurrence in a given document
- But balanced with its prevalence elsewhere in the pool of documents
- The more frequently it appears in any particular document, the more important it becomes
- Frequent appearances in other documents reduces its importance

Term Frequency (TF)

- Measures how often a term appears (density in a document) in a given document
  - Assumes important terms appear more often
  - Normalized to account for document length

Term Frequency (TF)

- Let *freq(t,d)* number of occurrences of keyword *t* in document *d*
- Let *max{freq(w,d)}* denote the highest number of occurrences of another keyword of *d*
- $TF(t,d) = \dfrac{freq(t,d)}{\max\{freq(w,d) : w \in d\}}$

(Frequency of a particular term in a document divided by the maximum frequency of any word in that document)

Term Frequency (TF)

CFA Institute®
@CFAinstitute

Following

“You are not a robo-adviser,” says @meirstatman, “Your advantage is not in beating the market . . . Your advantage is in creating this bond, this emotional bond, with your clients," via @laurenfosternyc

$$\max\{freq(w,d) : w \in d\} = 2$$

TF (advantage) = 2/2 = 1

TF (market) = ½ = 0.5

Inverse Document Frequency

- Aims to reduce the weight of terms that appear in many other documents
- Assumes terms that appear in many documents are less distinguishing

Inverse Document Frequency

- N: number of all recommendable documents
- n(t): number of documents in which keyword *t* appears
- $IDF(t) = log\frac{N}{n(t)}$

datasciencedojo
data science for everyone

IDF Example

- Given 1000 documents (could be tweets, articles, etc)
- The term "coffee" appears in 10 out of 1000 documents
- The term "mug" appears in all 1000 documents

IDF (coffee) = log 1000/10 = log 100 = 2

IDF (mug) = log 1000/1000 = log 1 = 0

Calculating TF-IDF

- Compute the overall importance of keywords
  - Given a keyword *t* and a document *d*

$$TF\text{-}IDF\,(t,d) = TF(t,d) * IDF(t)$$

TF-IDF Exercise

- **D1** = "If it walks like a duck and quacks like a duck, it must be a duck."
- **D2** = "Beijing Duck is mostly prized for the thin, crispy duck skin with authentic versions of the dish serving mostly the skin."
- **D3** = "Bugs' ascension to stardom also prompted the Warner animators to recast Daffy Duck as the rabbit's rival, intensely jealous and determined to steal back the spotlight while Bugs remained indifferent to the duck's jealousy, or used it to his advantage. This turned out to be the recipe for the success of the duo."
- **D4** = "6:25 PM 1/7/2007 blog entry: I found this great recipe for Rabbit Braised in Wine on cookingforengineers.com."
- **D5** = "Last week Li has shown you how to make the Sechuan duck. Today we'll be making Chinese dumplings (Jiaozi), a popular dish that I had a chance to try last summer in Beijing. There are many recipies for Jiaozi."

- **Dictionary:** {beijing, dish, duck, rabbit, recipe}

Creating the TF Matrix: Step 1

Step 1: Count the word frequency per document.

	beijing	dish	duck	rabbit	recipe
D1	0	0	3	0	0
D2	1	1	2	0	0
D3	0	0	2	1	1
D4	0	0	0	1	1
D5	1	1	1	0	1

Creating the TF Matrix: Step 2

Step 2: Normalize the counts by the most frequency word.

Normalized Frequency: $TF(t,d) = \frac{freq(t,d)}{\max\{freq(w,d): w \in d\}}$

	beijing	dish	duck	rabbit	recipe
D1	0 / 3	0 / 3	3 / 3	0 / 3	0 / 3
D2	1 / 2	1 / 2	2 / 2	0 / 2	0 / 2
D3	0 / 2	0 / 2	2 / 2	1 / 2	1 / 2
D4	0 / 1	0 / 1	0 / 1	1 / 1	1 / 1
D5	1 / 1	1 / 1	1 / 1	0 / 1	1 / 1

Creating the IDF Vector

TF Matrix

	beijing	dish	duck	rabbit	recipe
D1	0	0	1	0	0
D2	0.5	0.5	1	0	0
D3	0	0	1	0.5	0.5
D4	0	0	0	1	1
D5	1	1	1	0	1

IDF Vector

Word	IDF
beijing	log(5/2)
dish	log(5/2)
duck	log(5/4)
rabbit	log(5/2)
recipe	log(5/3)

TF-IDF Matrix

We calculate the TF-IDF numbers by multiplying TF and IDF

	beijing	dish	duck	rabbit	recipe
D1	0*log(5/2)	0*log(5/2)	1*log(5/4)	0*log(5/2)	0*log(5/3)
D2	0.5*log(5/2)	0.5*log(5/2)	1*log(5/4)	0*log(5/2)	0*log(5/3)
D3	0*log(5/2)	0*log(5/2)	1*log(5/4)	0.5*log(5/2)	0.5*log(5/3)
D4	0*log(5/2)	0*log(5/2)	0	1*log(5/2)	1*log(5/3)
D5	1*log(5/2)	1*log(5/2)	1*log(5/4)	0*log(5/2)	1*log(5/3)

TF-IDF Search Example

- User searches in our document set
- **Query:** "Beijing duck recipe"
- Calculate TF-IDF of query

Word	IDF
beijing	log(5/2)
dish	log(5/2)
duck	log(5/4)
rabbit	log(5/2)
recipe	log(5/3)

	beijing	dish	duck	rabbit	recipe
Query	1/1 * log(5/2)	0	1/1 * log(5/4)	0	1/1 * log(5/3)

TF-IDF Search Example

- Cosine similarity of query and each doc
  - $D1 = [0, 0, 0.097, 0, 0]$ (D1's TF-IDF score)
  - $Q = [0.398, 0, 0.097, 0, 0.222]$

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

  - $\cos(D1, Q) = \frac{0*0.398+0*0+0.097*0.097+0*0+0*0.222}{\sqrt{0.097^2}*\sqrt{0.398^2+0.097^2+0.222^2}} = 0.208$

Cosine similarities

	beijing	dish	duck	rabbit	recipe	cos(D,Q)
D1	0	0	0.097	0	0	0.208
D2	0.199	0.199	0.097	0	0	0.639
D3	0	0	0.097	0.199	0.111	0.256
D4	0	0	0	0.398	0.222	0.232
D5	0.398	0.398	0.097	0	0.222	0.760
Query	**.398**	**0**	**.097**	**0**	**.222**	1

Final ordered list

- **D5 =** "Last week Li has shown you how to make the Sechuan duck. Today we'll be making Chinese dumplings (Jiaozi), a popular dish that I had a chance to try last summer in Beijing. There are many recipies for Jiaozi."
- **D2 =** "Beijing Duck is mostly prized for the thin, crispy duck skin with authentic versions of the dish serving mostly the skin."
- **D3 =** "Bugs' ascension to stardom also prompted the Warner animators to recast Daffy Duck as the rabbit's rival, intensely jealous and determined to steal back the spotlight while Bugs remained indifferent to the duck's jealousy, or used it to his advantage. This turned out to be the recipe for the success of the duo."
- **D4 =** "6:25 PM 1/7/2007 blog entry: I found this great recipe for Rabbit Braised in Wine on cookingforengineers.com."
- **D1 =** "If it walks like a duck and quacks like a duck, it must be a duck."

N-grams

- Our representations so far have been single terms, known as *unigrams* or *1-grams*.
- There are also *bigrams*, *trigrams*, *4-grams*, *5-grams*, etc.
- N-grams allow us to extend the bags-of-words model to include word ordering

N-grams

- Take the sample document:
  - "If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck."

- A standard data pre-processing pipeline (stop word removal, stemming, etc.) would transform the above into something like:
  - "look like duck swim like duck quack like duck probabl duck"

- Which we could represent as a document-term frequency matrix:

look	like	duck	swim	quack	probabl
1	3	4	1	1	1

Bigrams

- Given the processed document,

"look like duck swim like duck quack like duck probabl duck"

The bigrams for the processed data would be:

look_like	like_duck	duck_swim	swim_like	duck_quack	quack_like	duck_probabl	probabl_duck
1	3	1	1	1	1	1	1

NOTE – We've now more than doubled the total size of our matrix!

Text Analytics Tools

- R – tm, Rstem, openNLP
- Python – NLTK
- Azure – Feature Hashing module

QUESTIONS