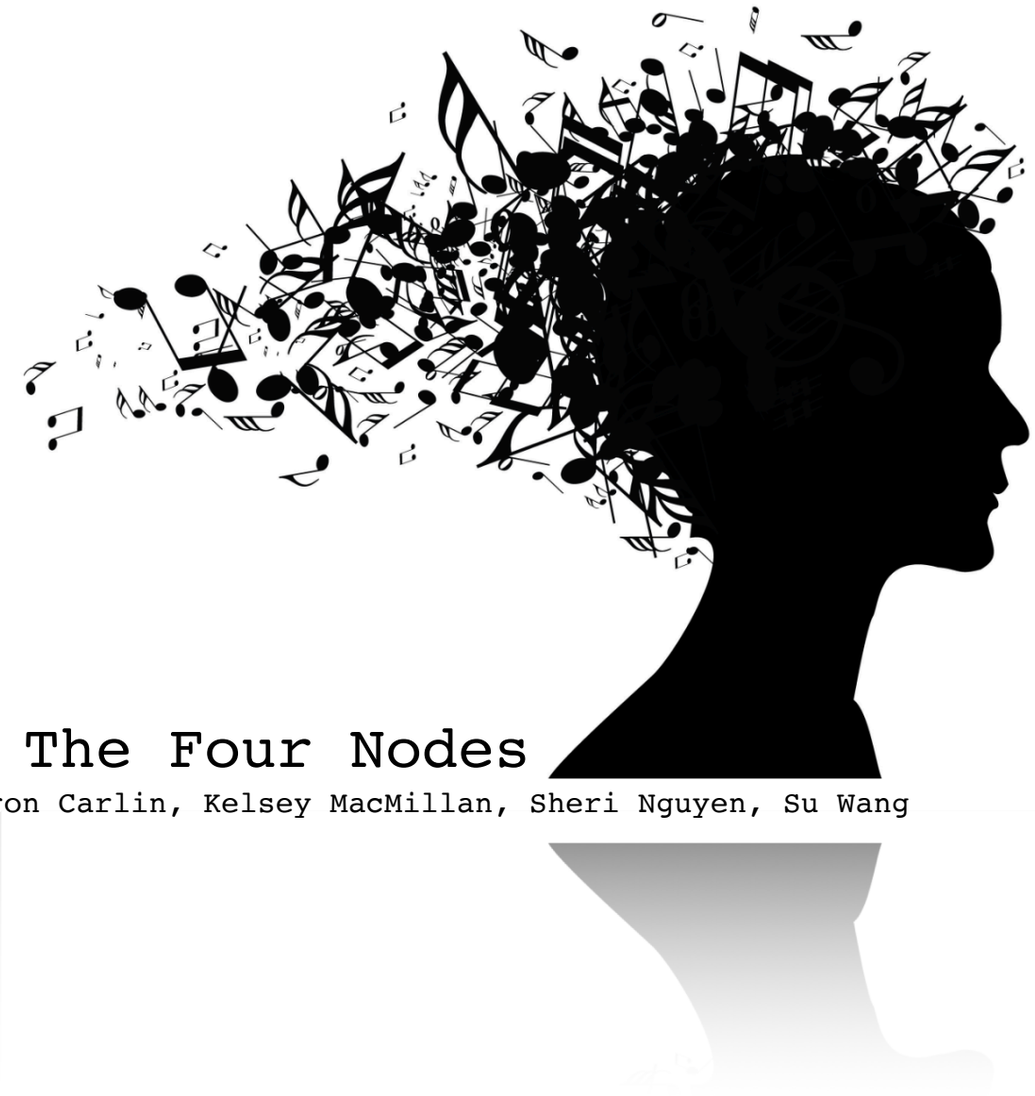
Song Lyric Classification & Relationships



By The Four Nodes

Cameron Carlin, Kelsey MacMillan, Sheri Nguyen, Su Wang

Data and Implementation

Where:

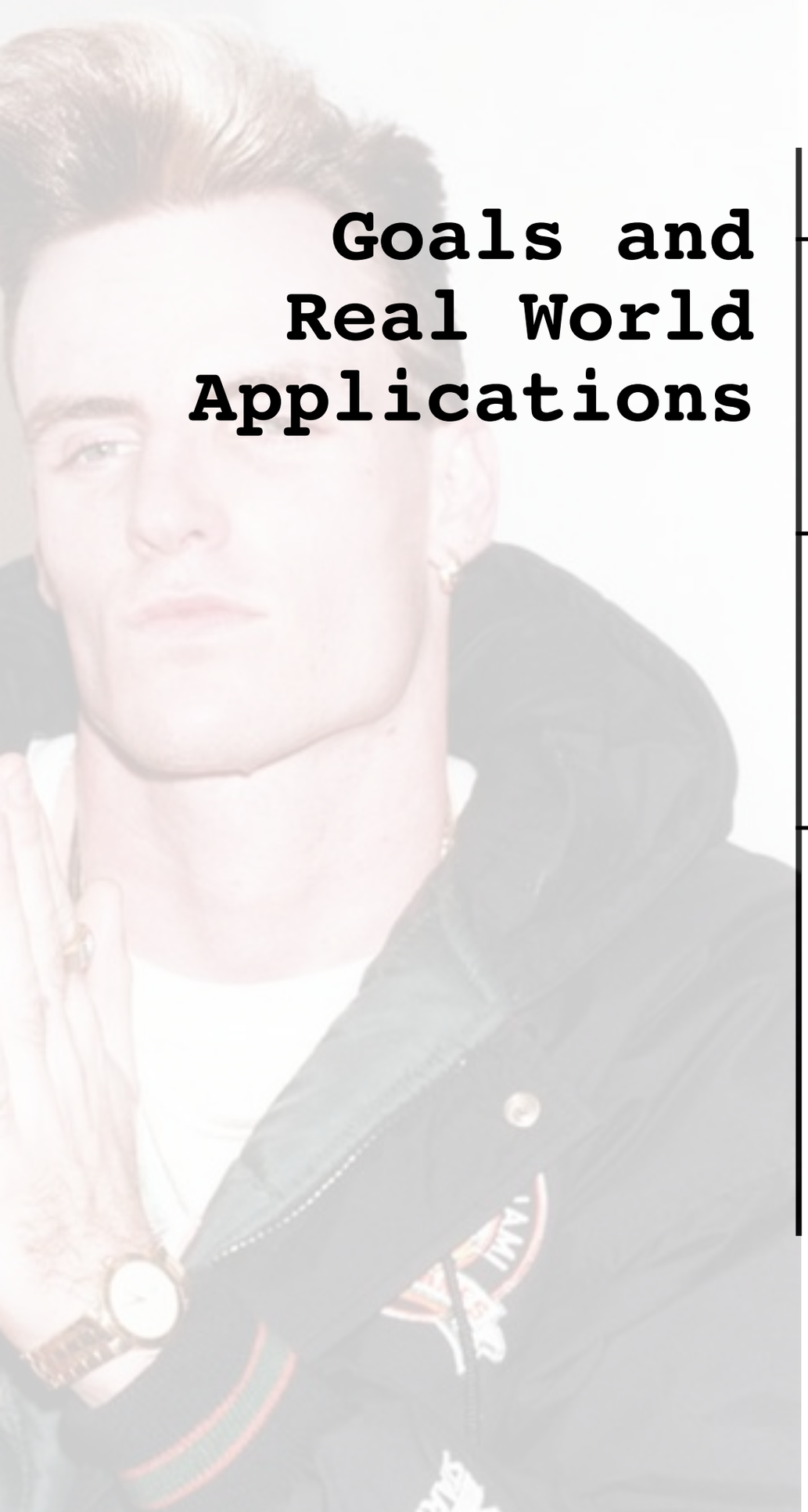
- Kaggle
  From LyricsFreak
  https://www.kaggle.com/mousehead/songlyrics

Description:

- 57,650 song lyrics
- Fields: Artist, Song Name, Link, Lyrics

EMR Suitability:

- Faster operations compared to local machine
- Data loading using S3, SparkML algorithms (particularly K-Means clustering)



Goals and Real World Applications

- **Goal:**
  - Analyze potential relationships between songs and artists
  - Classify a song's genre based on the words in its lyrics

- **How:**
  - Convert song lyrics into vectors using Word2Vec
  - Perform K-Means clustering
  - Describe common themes within clusters

- **Potential Applications:**
  - Develop song recommendation systems for popular music streaming apps such as Spotify and SoundCloud

Process

- **AWS EC2**
  - Download the data
  - Clean data using Pandas
  - Save cleaned data to S3 bucket
- **AWS EMR**
  - Spin-up EMR instance running YARN
  - Load data from S3 bucket as RDD
  - Convert to Spark DataFrame
- **SparkML**
  - Use Word2Vec to create feature vectors for each song
  - Save feature vectors as JSON table to S3 bucket
  - Use K-Means clustering to group songs using Word2Vec features
  - Save clusters as JSON table to S3 bucket
- **AWS S3**
  - Load data from S3
  - Perform sample analyses

Setting up EC2

Type

- Instance: t2-micro
- Note: For greater performance demands, upgrade to appropriate size, i.e. t2.xlarge

How

- Set up and start instance via AWS web app
- Install below within instance

Tools for Data Processing

Pandas Installation within instance

```
$sudo yum update
$sudo yum groupinstall “Development Tools”
$sudo yum install python-devel libpng-devel freetype-devel
$pip install --upgrade pip
$sudo pip install pandas
```

Anaconda Installation within instance

```
$wget http://repo.continuum.io/archive/Anaconda3-4.1.1-Linux-x86_64.sh
$sh Anaconda3—4.1.1-Linux-x86_64.sh
```

Get NLTK stopwords to use with scripts

```
$wget "https://gist.githubusercontent.com/sebleier/554280/raw/7e0e4a1ce04c2bb7bd41089c9821dbcf6d0c786c/NLTK's%2520list%2520of%2520english%2520stopwords"
$mv NLTK\'s%20list%20of%20english%20stopwords stopwords
```

Data Retrieval

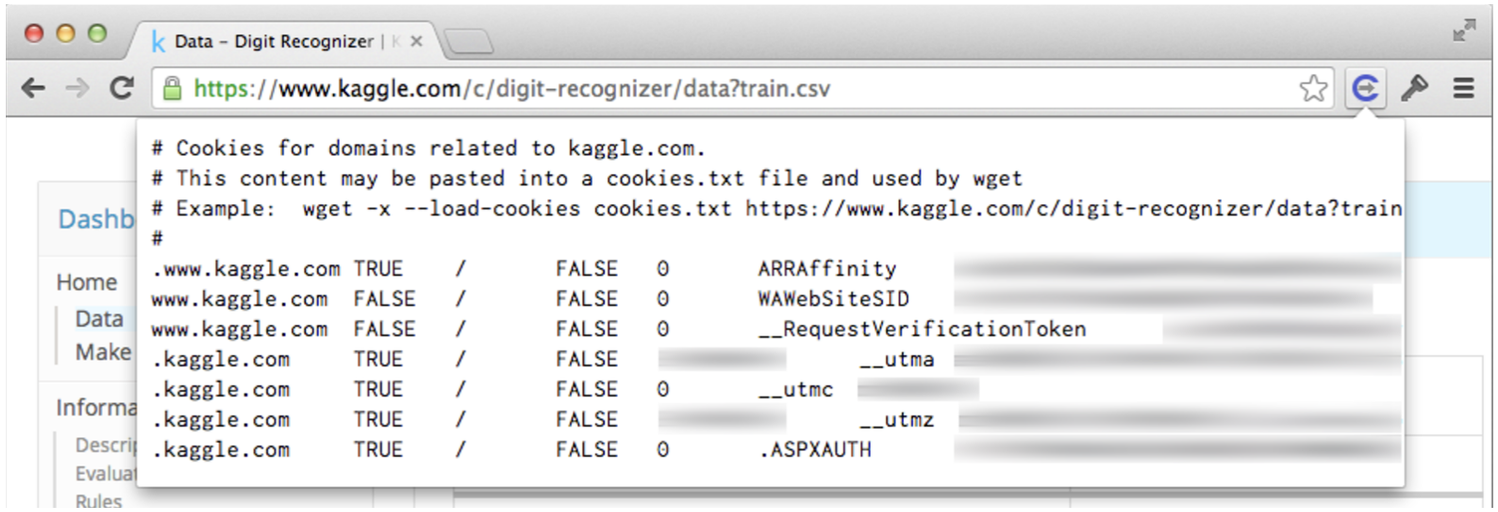
- **Kaggle CLI within EC2 instance**

  $ kg download -u `username` -p `password` -c `competition` -f `filename`

  - Doesn't work because dataset didn’t come from a competition

- **wget**
  - Does not work directly with Kaggle
  - Workaround
    - Use ‘cookie.txt export’ Chrome Extension
    - Save ‘cookie.txt’ in vi
    - Run below command line and load data onto S3

    $ wget -x --load-cookies cookies.txt https://www.kaggle.com/mousehead/songlyrics/downloads/songlyrics.zip



Data Cleaning

- **Spark CSV Parser**
  - Not suitable
  - Cannot handle data containing both commas and newlines
- **Pandas**
  - Stripped white spaces
  - Removed newlines and punctuations
  - Replaced multiple spaces with single spaces
  - Lowered all characters
  - Saved clean Pandas dataframe to ‘cleanedsongdata.csv’ and loaded onto S3

EMR Set Up

- Set up cluster and edit security groups

Add step | Resize | Clone | Terminate | AWS CLI export

Cluster: My cluster Starting Configuring cluster software

Connections: Enable Web Connection – Hue, Ganglia, Resource Manager … (View All)
Master public DNS: ec2-35-165-144-111.us-west-2.compute.amazonaws.com SSH
Tags: -- View All / Edit

Summary

ID: j-2J9912LG454QV
Creation date: 2017-01-17 16:19 (UTC-8)
Elapsed time: 1 minute
Auto-terminate: No
Termination protection: Off Change

Configuration Details

Release label: emr-5.2.1
Hadoop distribution: Amazon 2.7.3
Applications: Ganglia 3.7.2, Spark 2.0.2, Zeppelin 0.6.2
Log URI: s3://aws-logs-753835447179-us-west-2/elasticmapreduce/
EMRFS consistent view: Disabled

Security and Access

Key name: msan694
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Visible to all users: All Change
Security groups for Master: sg-ba4199c2 (ElasticMapReduce-master)
Security groups for Core & Task: sg-bb4199c3 (ElasticMapReduce-slave)

S3 to Spark DataFrame pipelime

S3 User Authentication

```
sc._jsc.hadoopConfiguration()\
.set("fs.s3n.awsAccessKeyId", "<access_key_id>")


sc._jsc.hadoopConfiguration()\
.set("fs.s3n.awsSecretAccessKey", "<secret_access_key>")
```

CSV to RDD

```
rdd = sc.textFile("s3n://songdatamsan694/cleanedsongdata.csv")
```

Split into Tuples

```
rdd_tuples = rdd.map(lambda x: x.split(",")) \
                .filter(lambda x: x[1]!='artist') \
                .map(lambda x: (x[1],x[2],x[3].split(" ")))
```

Convert RDD to DataFrame

```
df = rdd_tuples.toDF(['artist','song','text'])
```

Word2Vec with SparkML

SparkML Word2Vec Transformer

- Learns vector representation of words via skip-gram algorithm
- Converts every word in each song to a numerical vector
- Returns column with average vector for each song

```
+------+--------------------+--------------------+
|artist|                song|                text|
+------+--------------------+--------------------+
|  abba|ahes my kind of girl|[look, at, her, f...|
|  abba|     andante andante|[take, it, easy, ...|
|  abba|      as good as new|[ill, never, know...|
|  abba|                bang|[making, somebody...|
|  abba|      bangaboomerang|[making, somebody...|
+------+--------------------+--------------------+
only showing top 5 rows
```

Learn Mapping from words → vectors

```
word2Vec = Word2Vec(vectorSize=50, minCount=0, inputCol="text", outputCol="vector")
model = word2Vec.fit(df)
result = model.transform(df)
```

Upload vector JSON table to S3

```
result.write.mode("overwrite").format('json')\
.saveAsTable('songs_with_vec', path='s3n://songdatamsan694/songs_with_vecs')
```

Create Song Clusters with SparkML

SparkML

- K-Means clustering by Word2Vec features

```
kmeans = KMeans(k=20, seed=1, featuresCol='vector',predictionCol='cluster')
model = kmeans.fit(songs_with_vecs)
clusters = model.transform(songs_with_vecs)
```

artist	song	text	cluster
abba	ahes my kind of girl	[look, at, her, f...	12
abba	andante andante	[take, it, easy, ...	16
abba	as good as new	[ill, never, know...	19
abba	bang	[making, somebody...	7
abba	bangaboomerang	[making, somebody...	7
abba	burning my bridges	[well, you, hoot,...	9
abba	cassandra	[down, in, the, s...	0
abba	chiquitita	[chiquitita, tell...	8
abba	crazy world	[i, was, out, wit...	8
abba	crying over you	[im, waitin, for,...	16
abba	dance	[oh, my, love, it...	8
abba	dancing queen	[you, can, dance,...	9
abba	disillusion	[changing, moving...	8
abba	does your mother ...	[youre, so, hot, ...	16
abba	dream world	[agnetha, were, n...	7
abba	dum dum diddle	[i, can, hear, ho...	2
abba	eagle	[they, came, flyi...	18
abba	every good man	[every, good, man...	12
abba	fernando	[can, you, hear, ...	0
abba	fernando in spanish	[puedes, escuchar...	11

only showing top 20 rows

Save cluster assignments to S3

```
clusters.write.mode("overwrite").format('json')\
.saveAsTable('songs_with_clusters', path='s3n://songdatamsan694/songs_with_clusters')
```

JSON tables from S3 to Spark DataFrame

S3 User Authentication

```
sc._jsc.hadoopConfiguration()\
.set("fs.s3n.awsAccessKeyId", "<access_key_id>")

sc._jsc.hadoopConfiguration()\
.set("fs.s3n.awsSecretAccessKey", "<secret_access_key>")
```

Load .JSON as DataFrame

```
data = spark.read.json(path='s3n://songdatamsan694/songs_with_vecs')
```

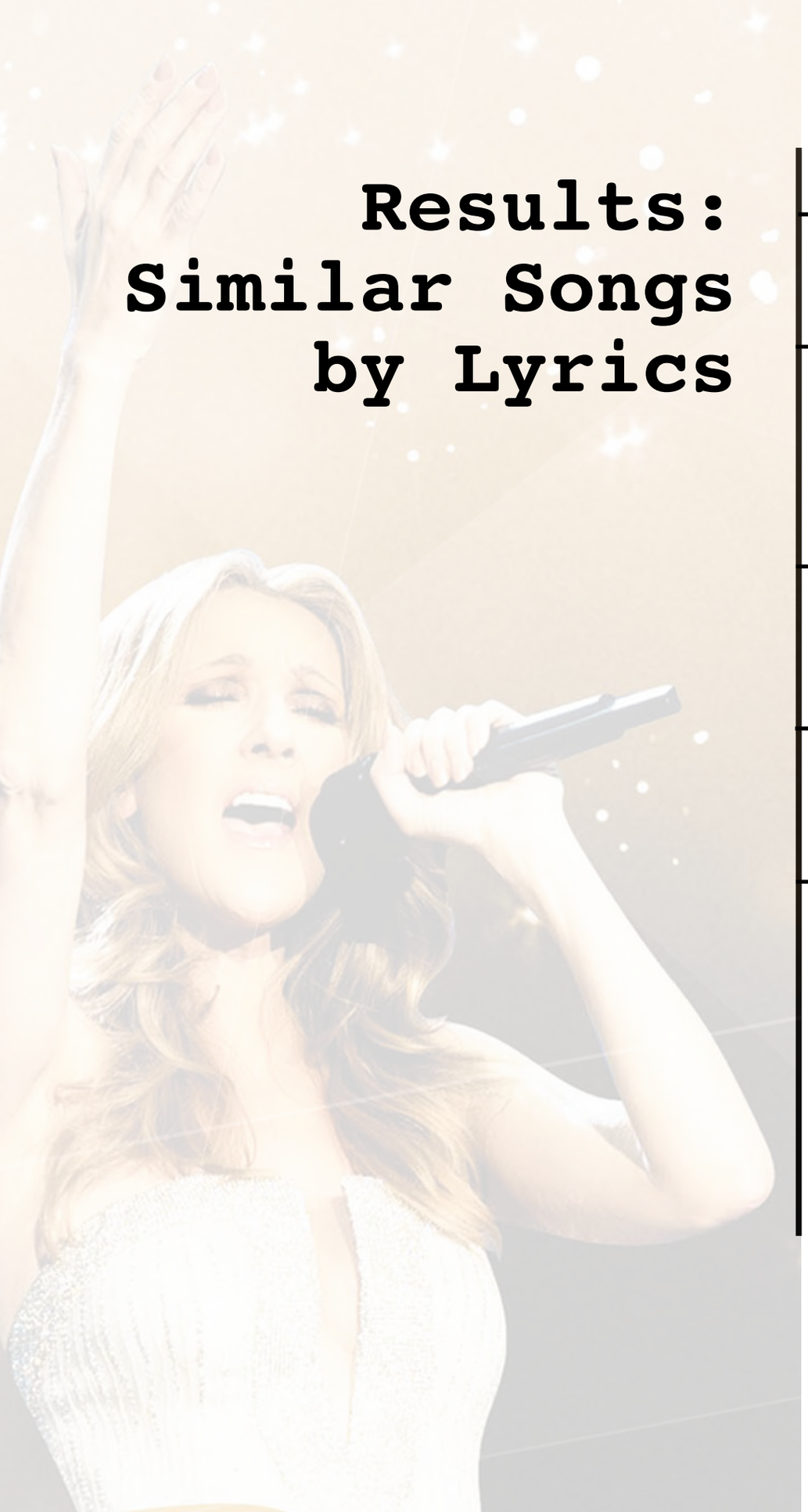
Convert Word2Vec back to DenseVector

```
toVector = udf(lambda x: Vectors.dense(x[1]), VectorUDT())
songs_with_vecs = data.select('artist','song','text',toVector('vector')\
                              .alias('vector')).cache()
```

- Cache DataFrame to use in K-Means (iterative)

```
+------+--------------------+--------------------+--------------------+
|artist|                song|                text|              vector|
+------+--------------------+--------------------+--------------------+
|  abba|ahes my kind of girl|[look, at, her, f...|[-0.0024252008445...|
|  abba|     andante andante|[take, it, easy, ...|[-0.0568260343315...|
|  abba|      as good as new|[ill, never, know...|[-0.0306186712032...|
|  abba|                bang|[making, somebody...|[-0.0459822696028...|
```

Results: Similar Songs by Lyrics



DataFrame → TempTable for SQL querying

```
songs_with_vecs.registerTempTable('songs_with_vecs')
```

Select song to compare

```
song_vec = sqlContext.sql\
("SELECT vector from songs_with_vecs\
WHERE artist = 'abba' AND song = 'dancing queen'")
song_vec.show()
```

Broadcast song to compare to others

```
song_vec_broadcast = sc.broadcast(song_vec.collect()[0][0])
song_vec_broadcast.value
```

UDF → Calculate distance between vectors

```
sqlContext.registerFunction('vDistance',\
lambda v: float(np.linalg.norm(v-song_vec_broadcast.value)), FloatType())
```

Calculate distance to all other songs

```
sqlContext.sql\
("SELECT artist, song, vDistance(vector) AS distance\
 FROM songs_with_vecs ORDER BY vDistance(vector) ASC").show()
```

Results:
Similar Songs by Lyrics

artist	song	distance
abba	dancing queen	0.0
kylie minogue	dancing queen	0.030706976
abba	reina danzante	0.034336165
glee	dancing queen	0.075292096
regine velasquez	dancing queen	0.13296847
ub40	music so nice	0.19429958
ll cool j	ahh lets get ill	0.22084112
madonna	now im following you	0.22240157
elvis costello	ghost train	0.22447343
yello	daily disco	0.22748236
lou reed	modern dance	0.22794528
rolling stones	gloom and doom	0.22795321
ugly kid joe	funky fresh count...	0.22806905
will smith	summertime	0.23010315
nickelback	rockstar	0.23149315
vince gill	next big thing	0.23162898
erasure	sunday girl	0.23194632
neil young	harvest moon	0.23203978
chris rea	twisted wheel	0.23269044
doobie brothers	music man	0.23284152

only showing top 20 rows

artist	song	distance
kanye west	gold digger	0.0
lil wayne	give head	0.14428505
glee	gold digger	0.14830123
insane clown posse	just like that	0.16482799
insane clown posse	first day out	0.1648633
gucci mane	beat it up	0.16520454
eminem	insane	0.16527808
florida	low	0.16938315
chris brown	cali swag	0.169994
ice cube	you cant fade me	0.17017235
snoop dogg	in love with a thug	0.17313021
insane clown posse	please dont hate me	0.17332664
eminem	maxine	0.17365943
lil wayne	girl you know	0.17532556
lil wayne	earthquake	0.17536649
insane clown posse	cotton candy	0.17632075
kanye west	apologize	0.17746963
eminem	my dads gone crazy	0.17747127
europe	mr goverment man	0.17747204
ll cool j	after school	0.17764296

only showing top 20 rows

Results: Top Artists per Lyric Cluster

- **Register TempTable for SQL Querying**

```
songs_with_clusters.registerTempTable('songs_with_clusters')
```

- **SparkSQL**

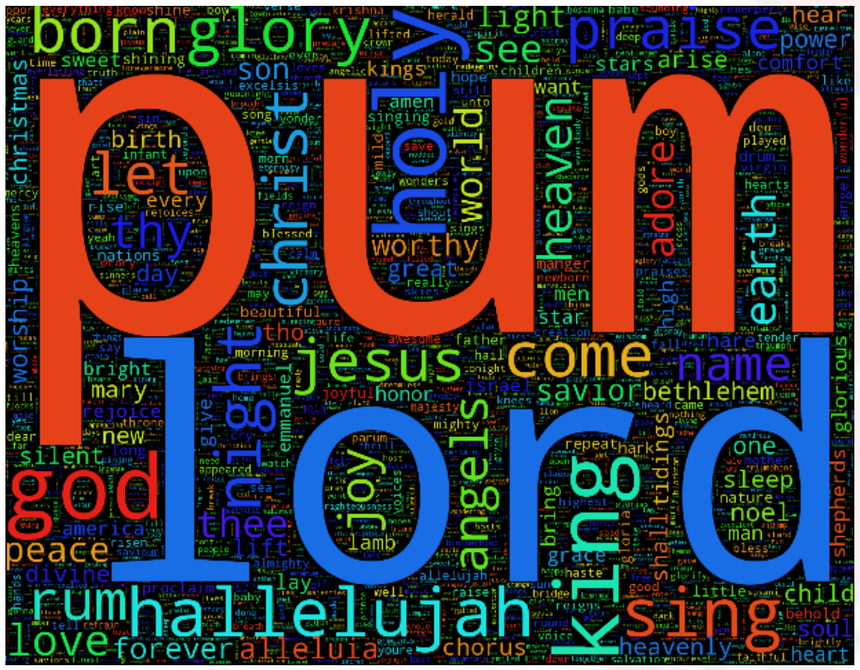
- show top 5 artists per cluster

```
for i in range(10):
    sqlContext.sql\
    ('SELECT artist, count(*) FROM songs_with_clusters\
    WHERE cluster = ' + str(i) + \
    'GROUP BY artst ORDER BY count(*) DESC\
    LIMIT 3').show
```

Results:
Top Artists and Word Clouds per Cluster

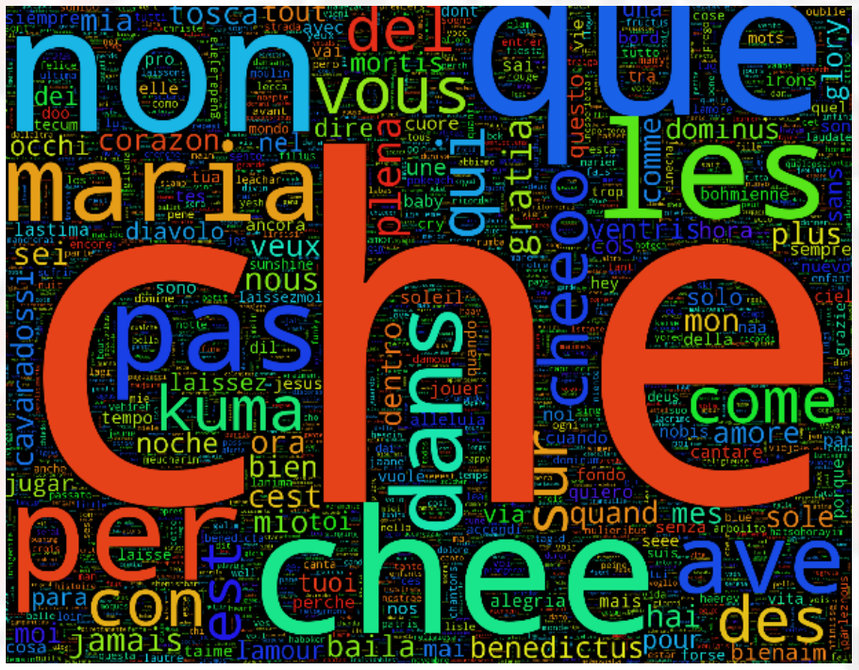
Cluster 3: Christian

artist	count(1)
religious music	34
michael w smith	31
indiana bible col...	30
hillsong	29
christmas songs	24



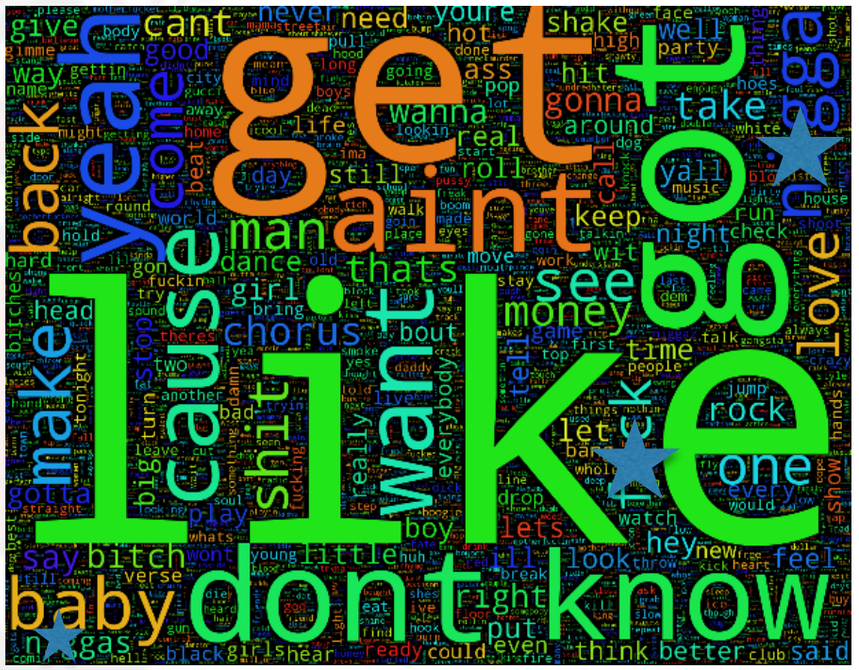
Cluster 4: Latin Pop

artist	count(1)
celine dion	12
andrea bocelli	6
josh groban	6
zucchero	6
neil sedaka	4



Cluster 6: Rap

artist	count(1)
lil wayne	100
fabolous	89
insane clown posse	88
gucci mane	79
ice cube	75



Learning Outcomes

EMR

- Don’t select wrong EMR cluster image

S3

- Connecting to private S3 bucket via pyspark requires authentication

```
sc._jsc.hadoopConfiguration()\
.set("fs.s3n.awsAccessKeyId", "<access_key_id>")

sc._jsc.hadoopConfiguration()\
.set("fs.s3n.awsSecretAccessKey", "<secret_access_key>")
```

Spark

- Spark CSV parser can’t deal with complex CSV conventions
- Spark DataFrames and UDFs don’t play well with all data types

```
sqlContext.registerFunction('vDistance', lambda v:
float(np.linalg.norm(v-
song_vec_broadcast.value)).FloatType())
```

- Saving as .JSON coerces vectors to string array
  - Requires conversion

```
toVector=udf(lambda x: Vectors.dense(x[1]), VectorUDT())
```

Thank You

The Four Nodes