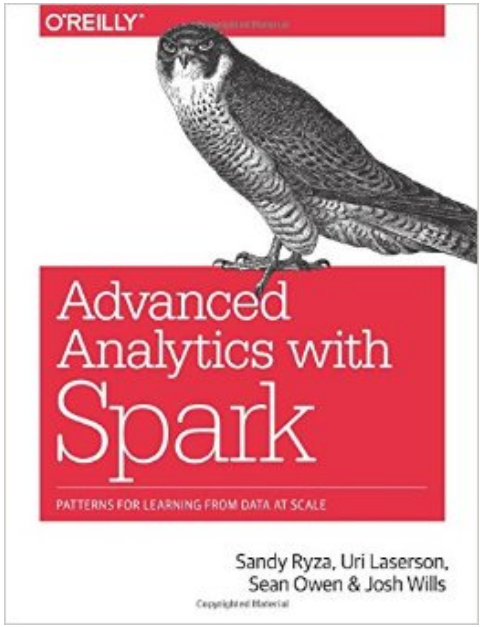
cloudera

# LSA-ing Wikipedia with Spark

Sandy Ryza | Senior Data Scientist

# Me

- Data scientist at Cloudera
- Recently lead Cloudera's Apache Spark development
- Author of Advanced Analytics with Spark




cloudera

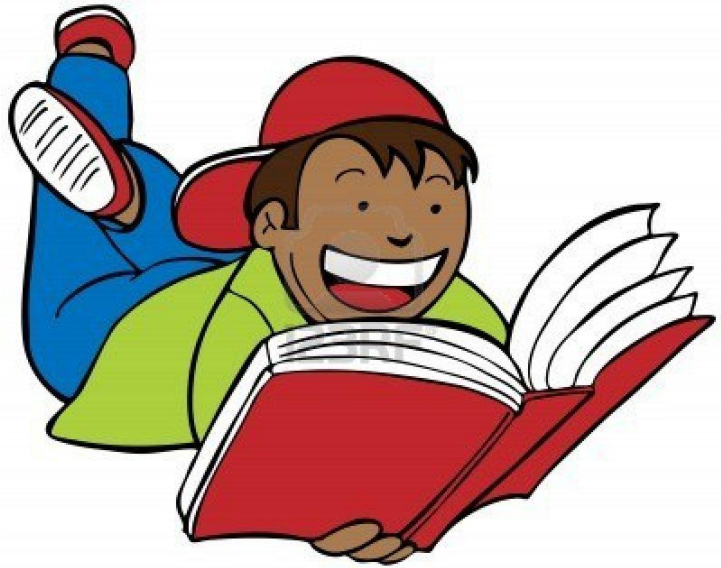
cloudera

# LSA-ing Wikipedia with Spark

Sandy Ryza | Senior Data Scientist

# Latent Semantic Analysis

- Fancy name for applying a matrix decomposition (SVD) to text data



cloudera

WIKIPEDIA

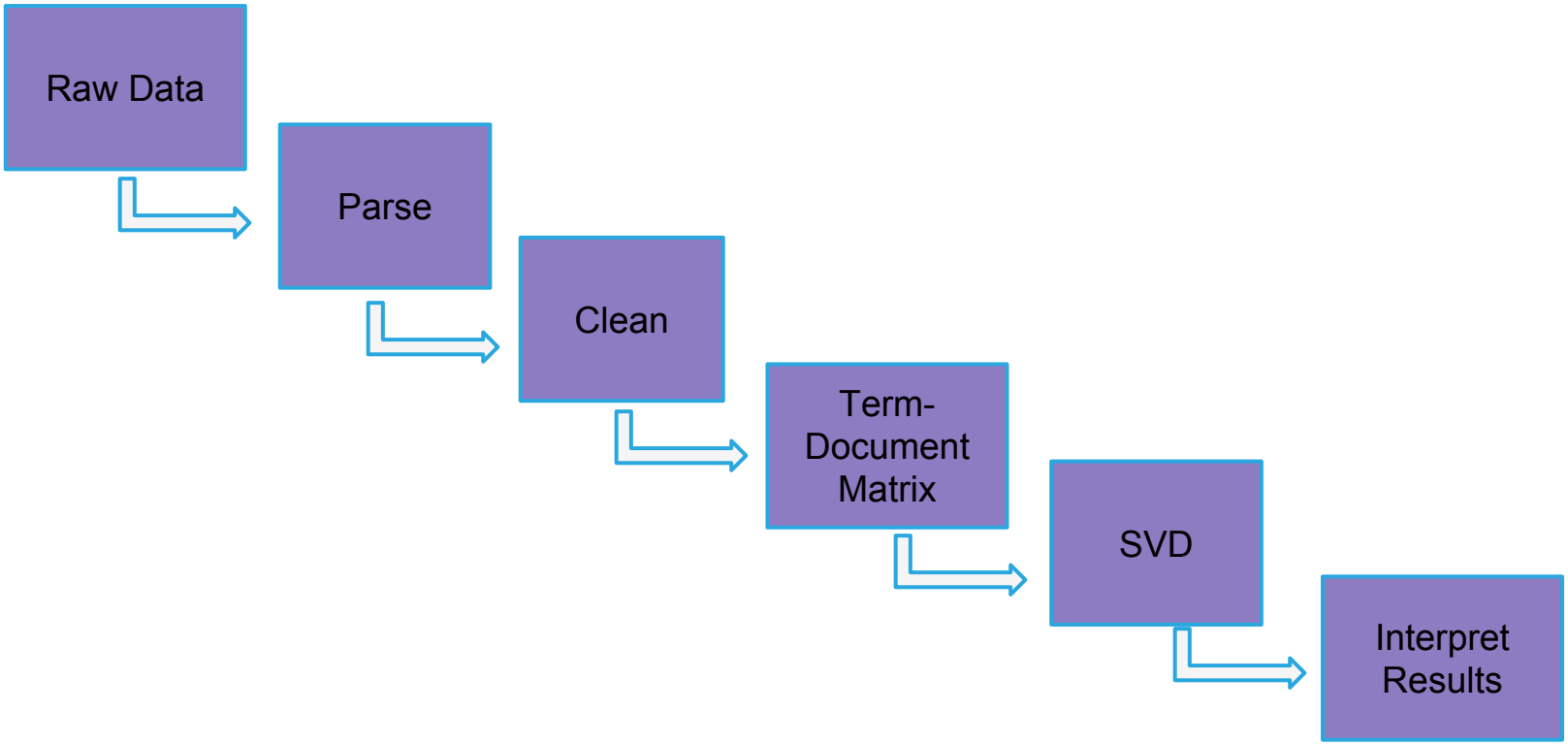
*The Free Encyclopedia*

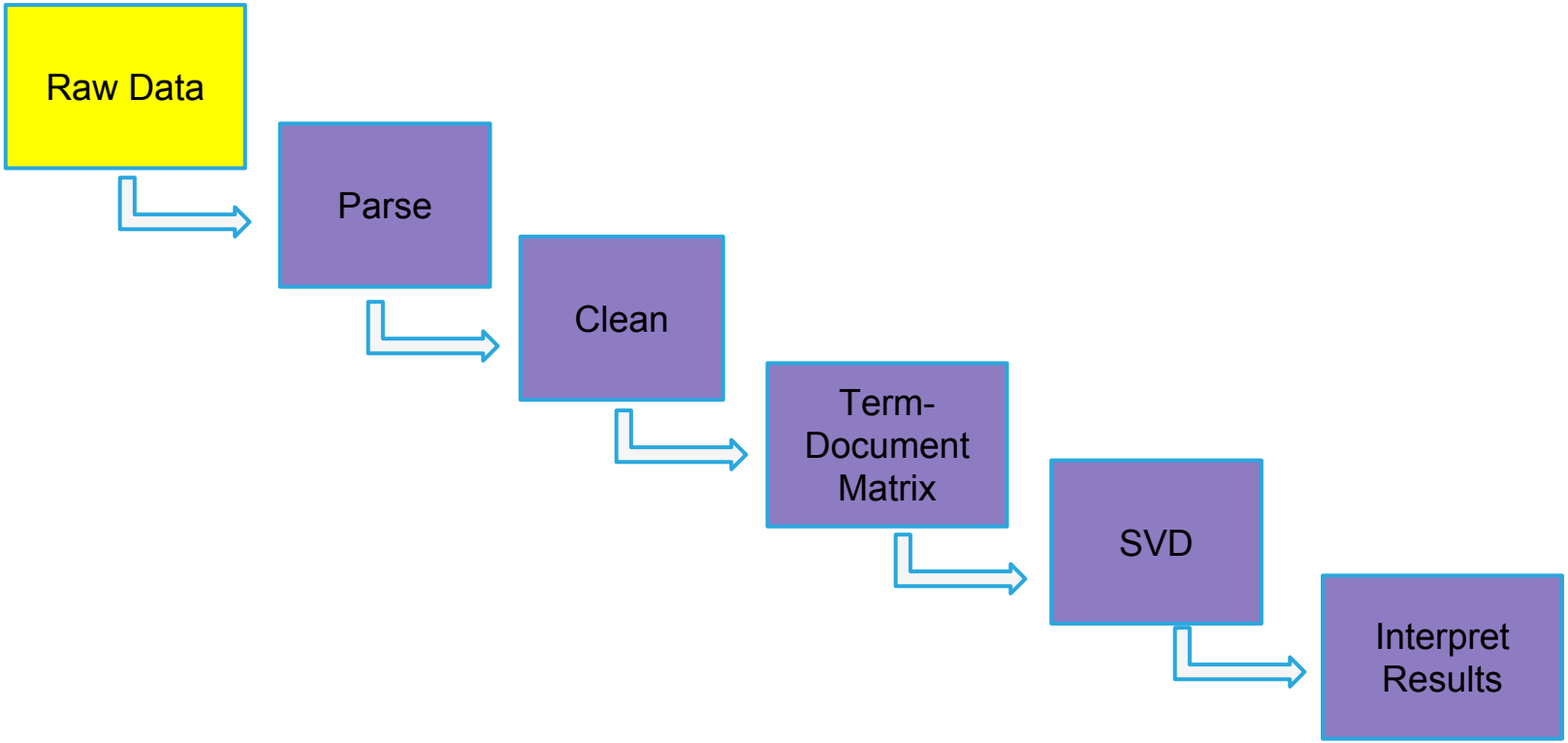






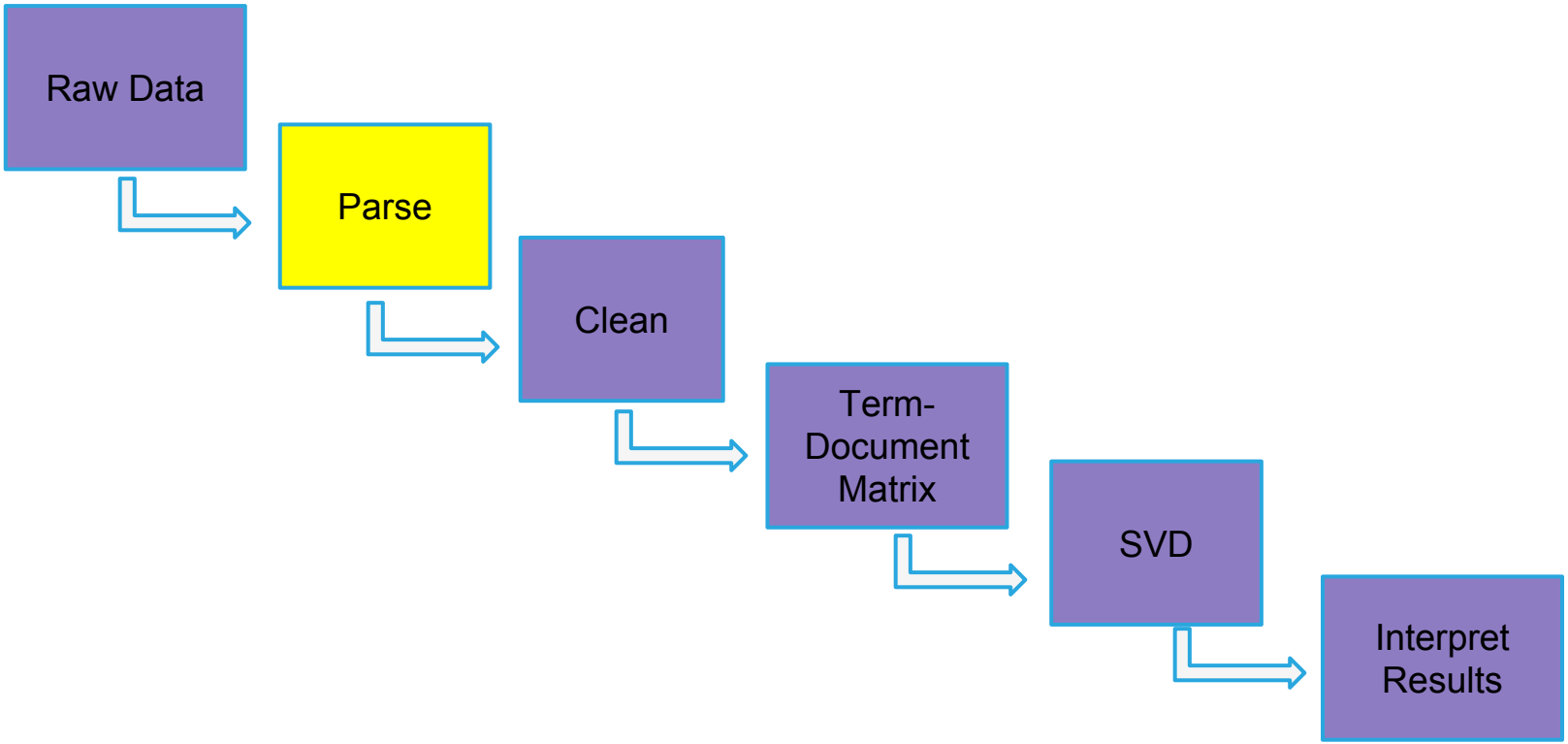


# Wikipedia Content Data Set

- http://dumps.wikimedia.org/enwiki/latest/
- XML-formatted
- 46 GB uncompressed

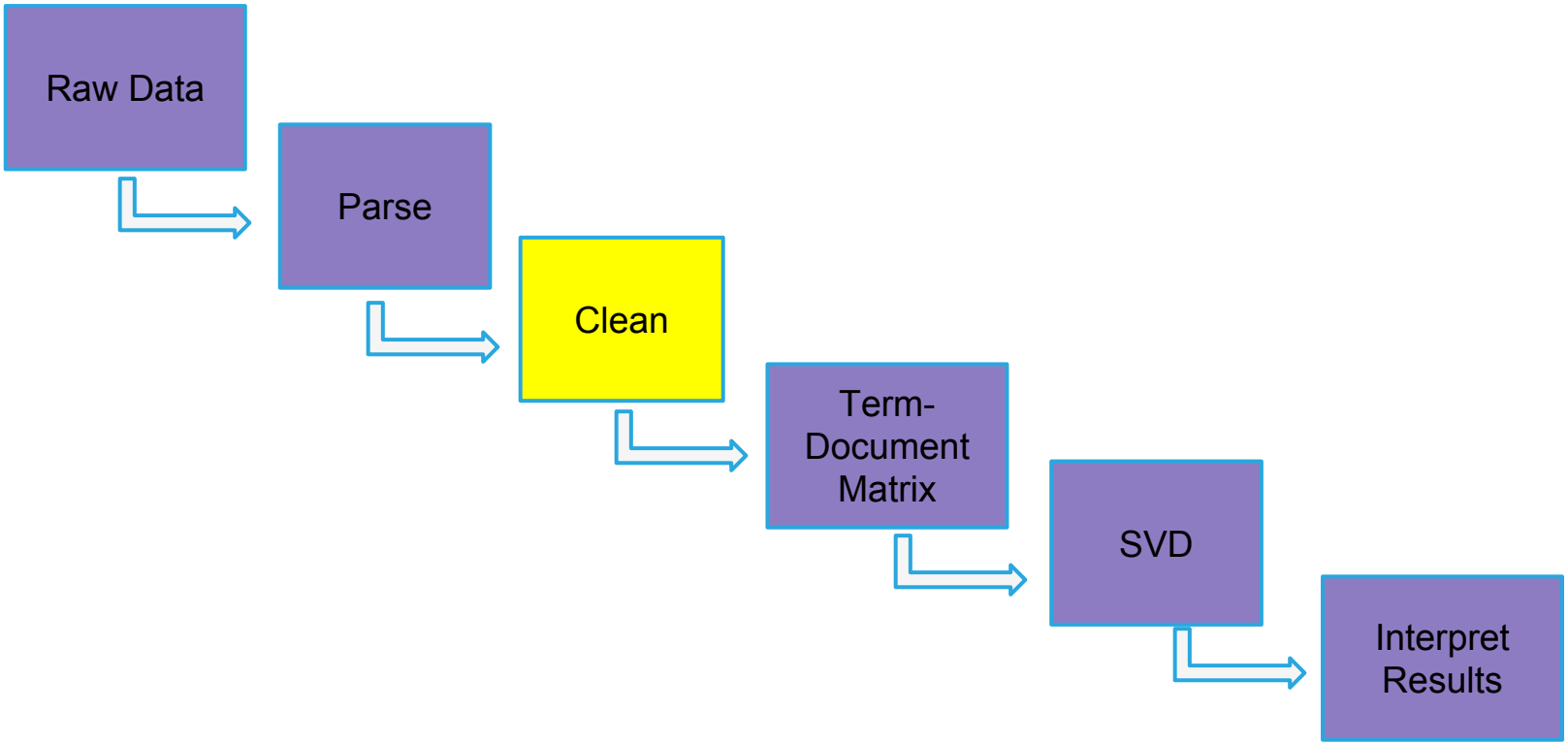
```
<page>
<title>Anarchism</title>
<ns>0</ns>
<id>12</id>
<revision>
<id>584215651</id>
<parentid>584213644</parentid>
<timestamp>2013-12-02T15:14:01Z</timestamp>
<contributor>
<username>AnomieBOT</username>
<id>7611264</id>
</contributor>
<comment>Rescuing orphaned refs (&quot;autogenerated1&quot; from rev
584155010; &quot;bbc&quot; from rev 584155010)</comment>
<text xml:space="preserve">{{Redirect|Anarchist|the fictional character|
Anarchist (comics)}}
{{Redirect|Anarchists}}
{{pp-move-indef}}
{{Anarchism sidebar}}

'''Anarchism''' is a [[political philosophy]] that advocates [[stateless society|
stateless societies]] often defined as [[self-governance|self-governed]] voluntary
institutions,&lt;ref&gt;&quot;ANARCHISM, a social philosophy that rejects
authoritarian government and maintains that voluntary institutions are best suited
to express man's natural social tendencies&quot; George Woodcock.
&quot;Anarchism&quot; at The Encyclopedia of Philosophy&lt;/ref&gt;&lt;ref&gt;
&quot;In a society developed on these lines, the voluntary associations which
already now begin to cover all the fields of human activity would take a still
greater extension so as to substitute
...
```



```
import org.apache.mahout.text.wikipedia.XmlInputFormat
import org.apache.hadoop.conf.Configuration
import org.apache.hadoop.io._

val path = "hdfs:///user/ds/wikidump.xml"
val conf = new Configuration()
conf.set(XmlInputFormat.START_TAG_KEY, "<page>")
conf.set(XmlInputFormat.END_TAG_KEY, "</page>")
val kvs = sc.newAPIHadoopFile(path,
  classOf[XmlInputFormat],
  classOf[LongWritable],
  classOf[Text],
  conf)
val rawXmls = kvs.map(p => p._2.toString)
```



# Lemmatization

“the boy’s cars are different colors”

“the boy car be different color”



# CoreNLP

```
def createNLPPipeline(): StanfordCoreNLP = {
  val props = new Properties()
  props.put("annotators", "tokenize, ssplit, pos, lemma")
  new StanfordCoreNLP(props)
}
```

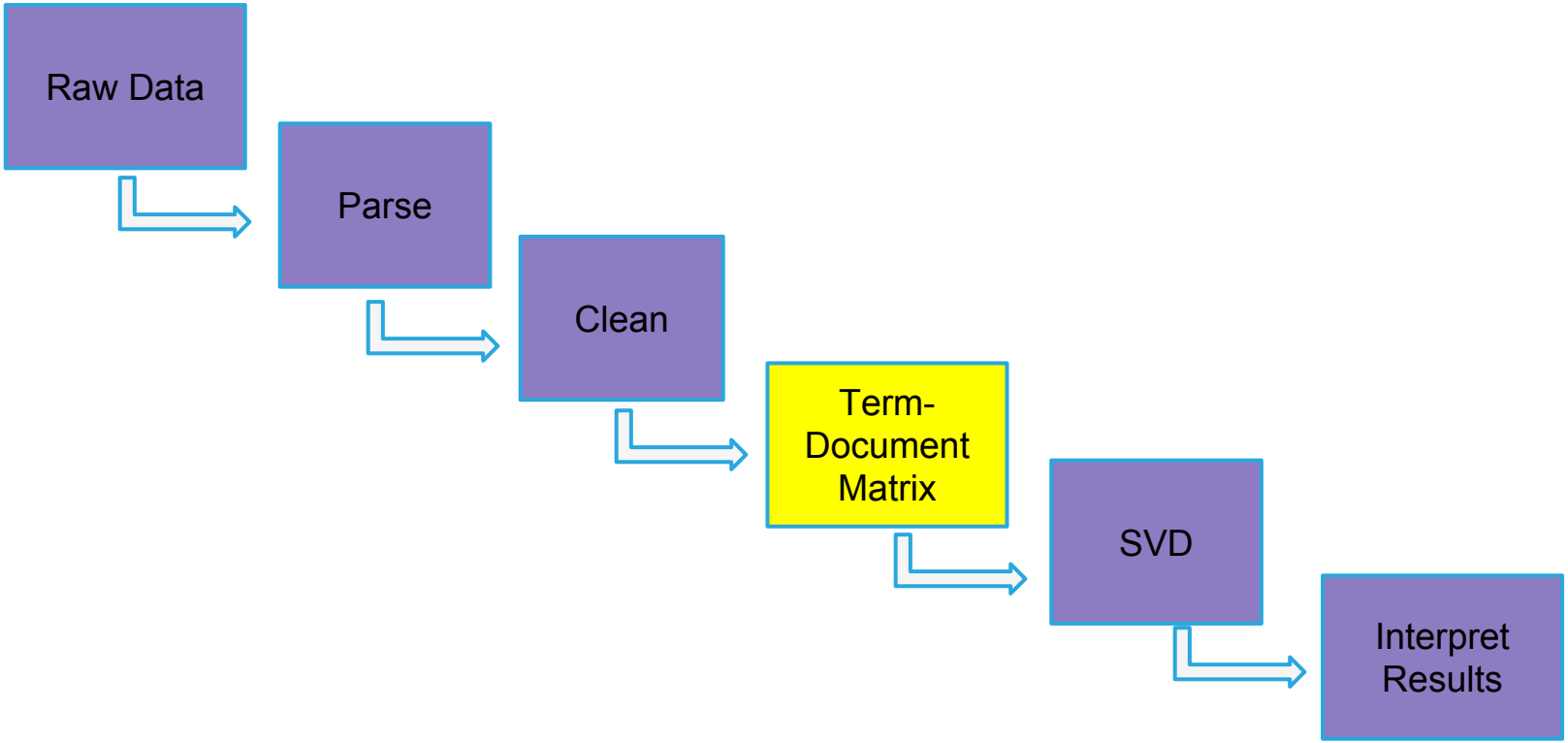
# Stop Words

"**the** boy car **be** different color"

"boy car different color"

cloudera



# Term-Document Matrix

| | Tail | Monkey | Algorithm | Scala |
|---|---|---|---|---|
| Document 1 | 1.5 | 1.8 | | |
| Document 2 | | | 2.0 | 4.3 |
| Document 3 | | 1.4 | 6.7 | |
| Document 4 | | | | 1.6 |
| Document 5 | 1.2 | | | |

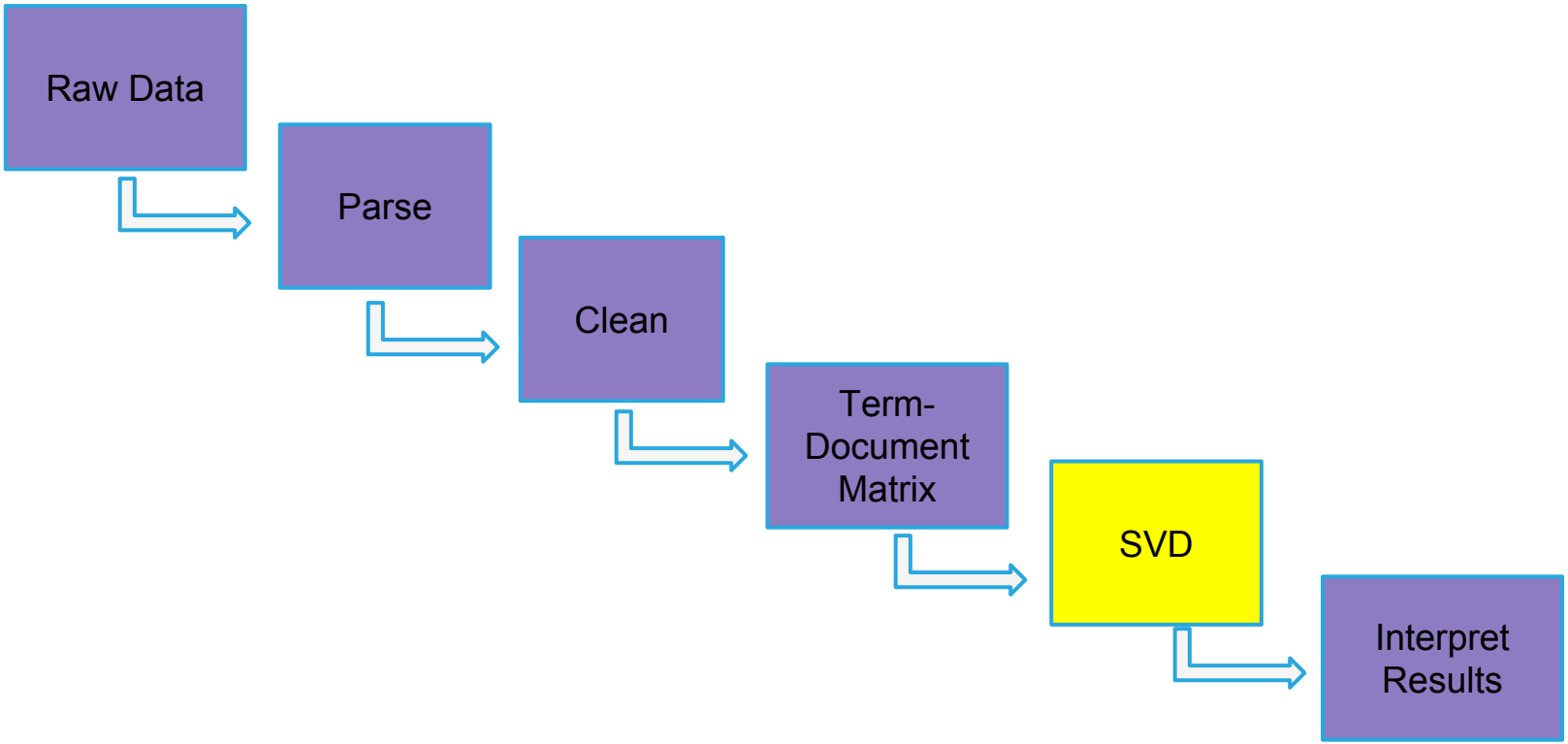
# tf-idf

- (Term Frequency) * (Inverse Document Frequency)
- tf(document, word) = # times word appears in document
- idf(word) = 1 / (# documents that contain word)

```
val rowVectors: RDD[Vector] = ...
```





# Singular Value Decomposition

- Factors matrix into the product of three matrices: U, S, and V
- m = # documents
- n = # terms
- U is m x n
- S is n x n
- V is n x n

# Low Rank Approximation

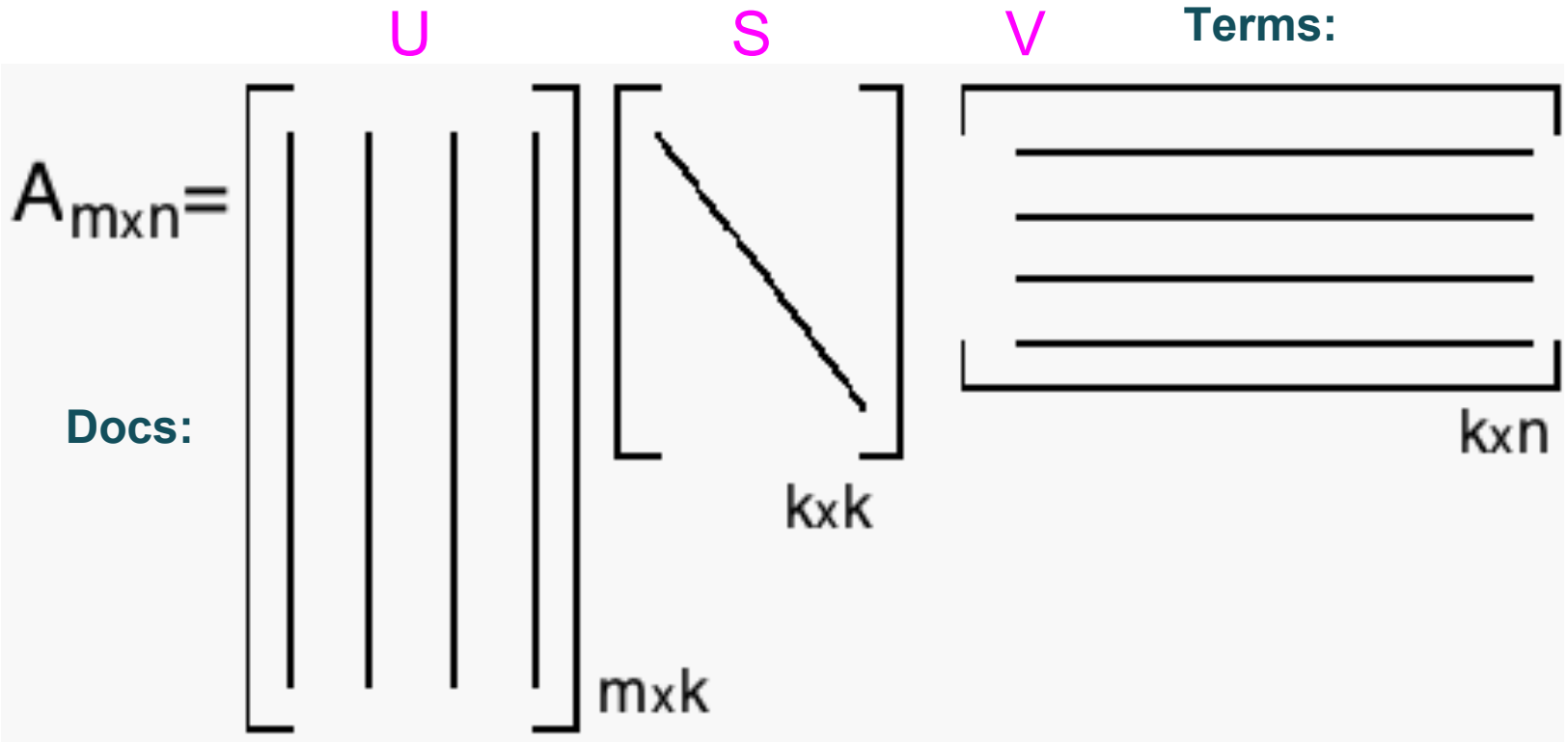
- Account for synonymy by condensing related terms.
- Account for polysemy by placing less weight on terms that have multiple meanings.
- Throw out noise.

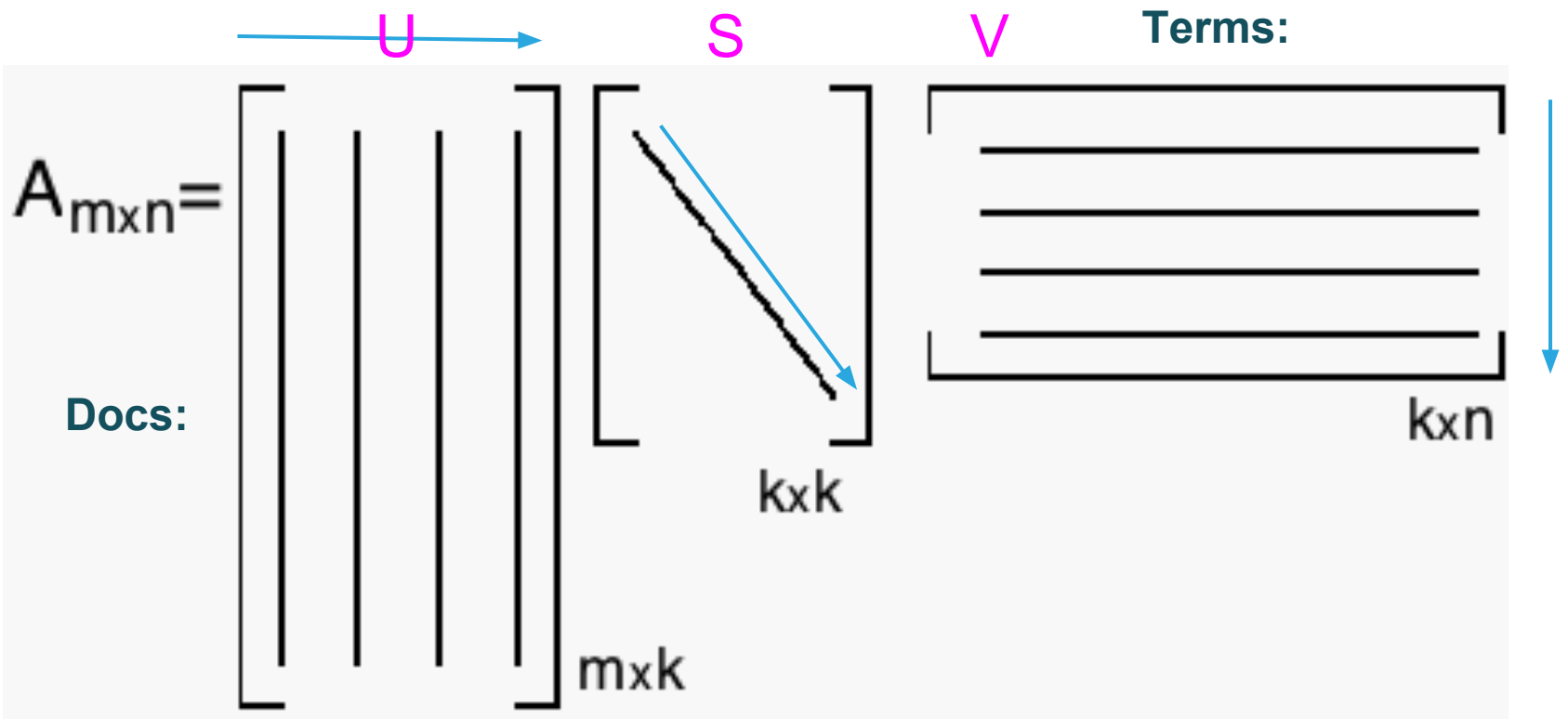
SVD can find the rank-k approximation that has the lowest Frobenius distance from the original matrix.

# Singular Value Decomposition

- Factors matrix into the product of three matrices: U, S, and V
- m = # documents
- n = # terms
- **k = # concepts**
- U is m x n
- S is **k** x **k**
- V is **k** x n

cloudera





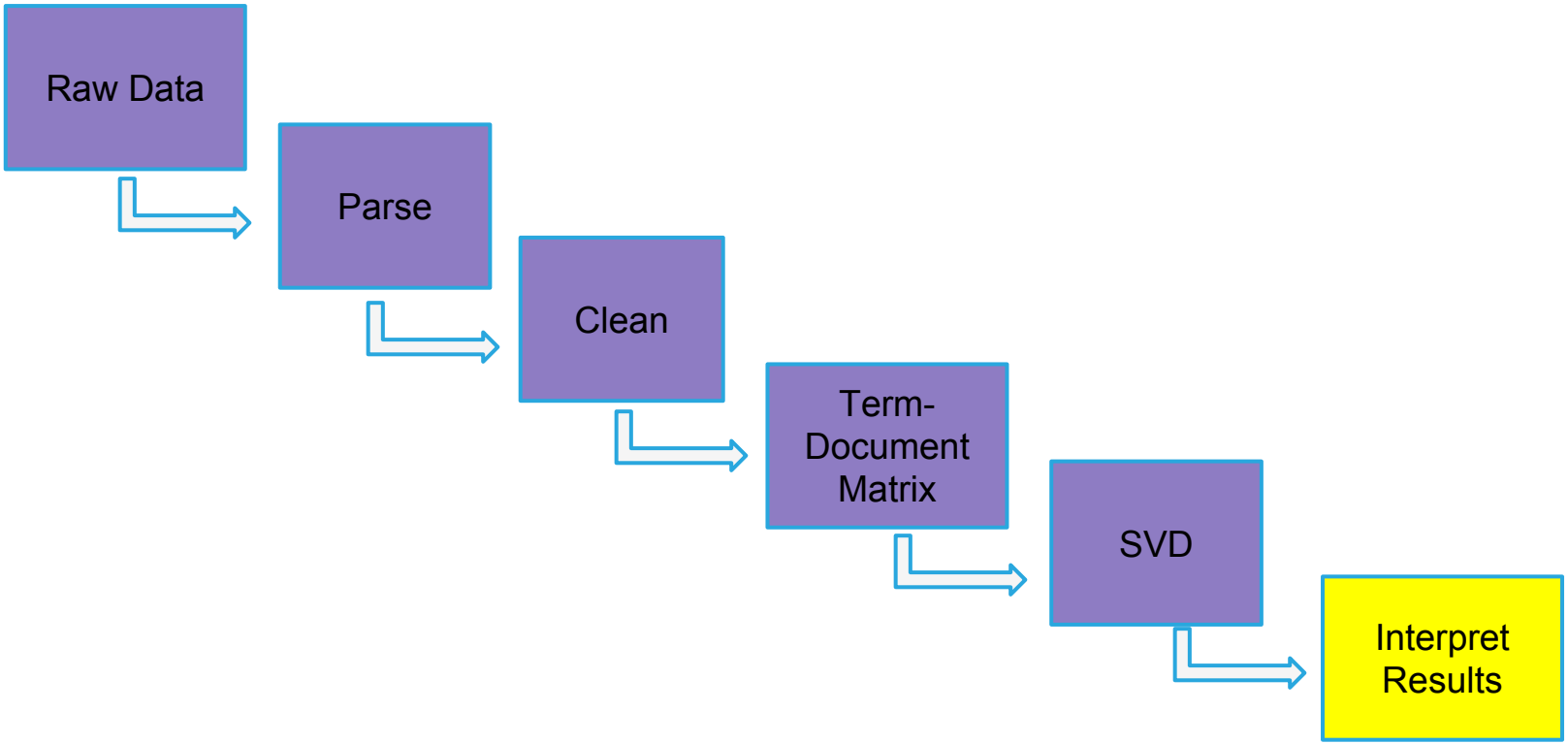


```
rowVectors.cache()
val mat = new RowMatrix(rowVectors)
val k = 1000
val svd = mat.computeSVD(k, computeU=true)
```

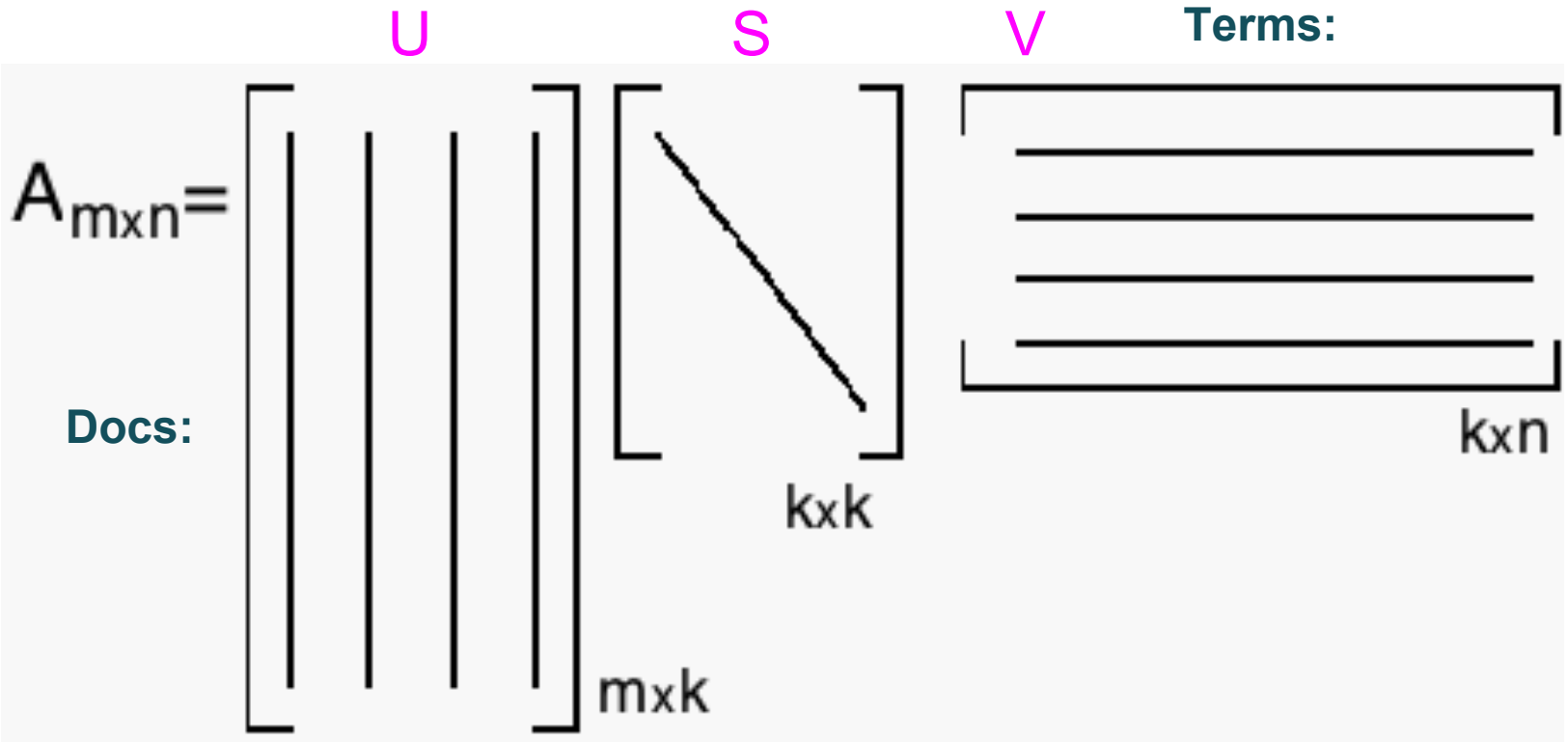
cloudera

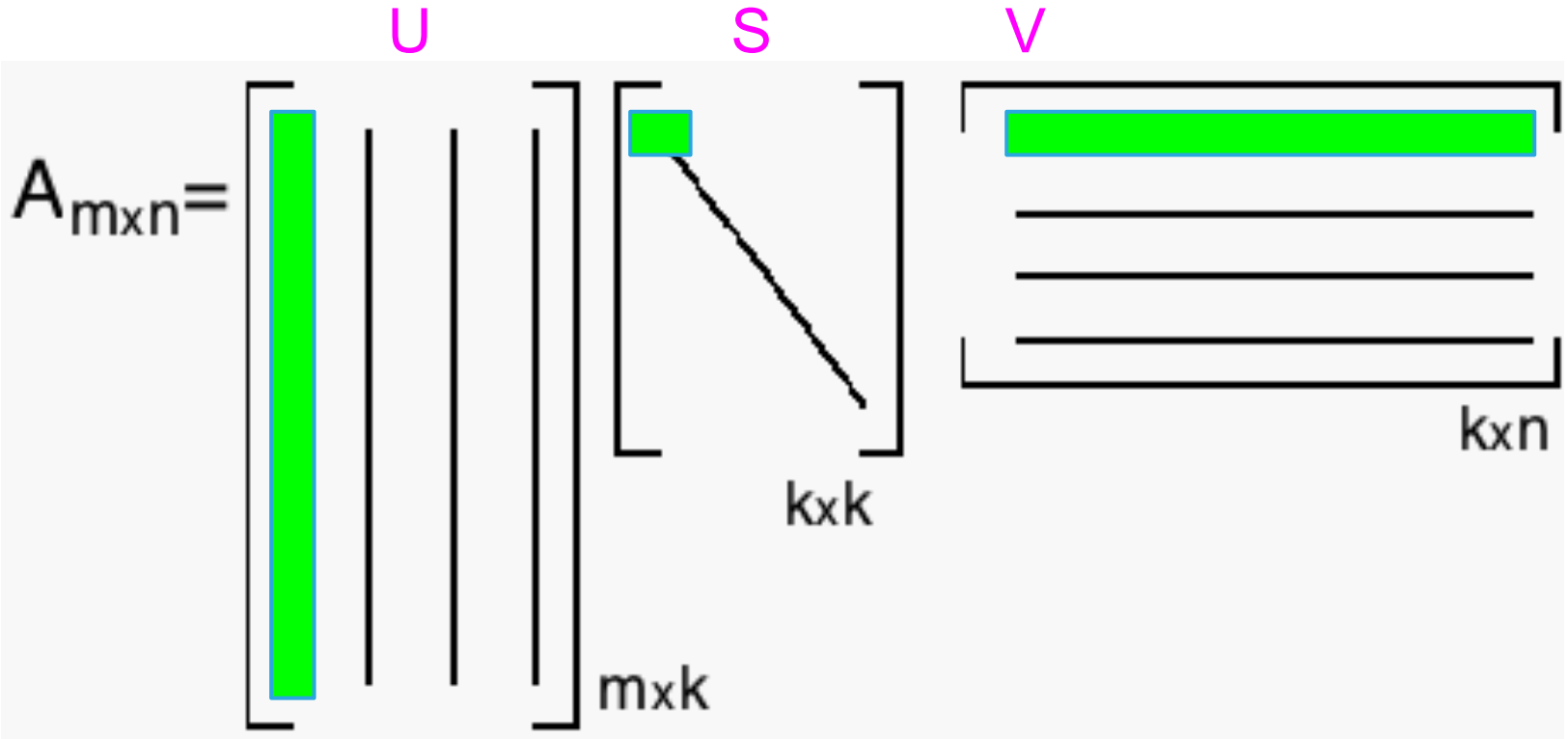
 

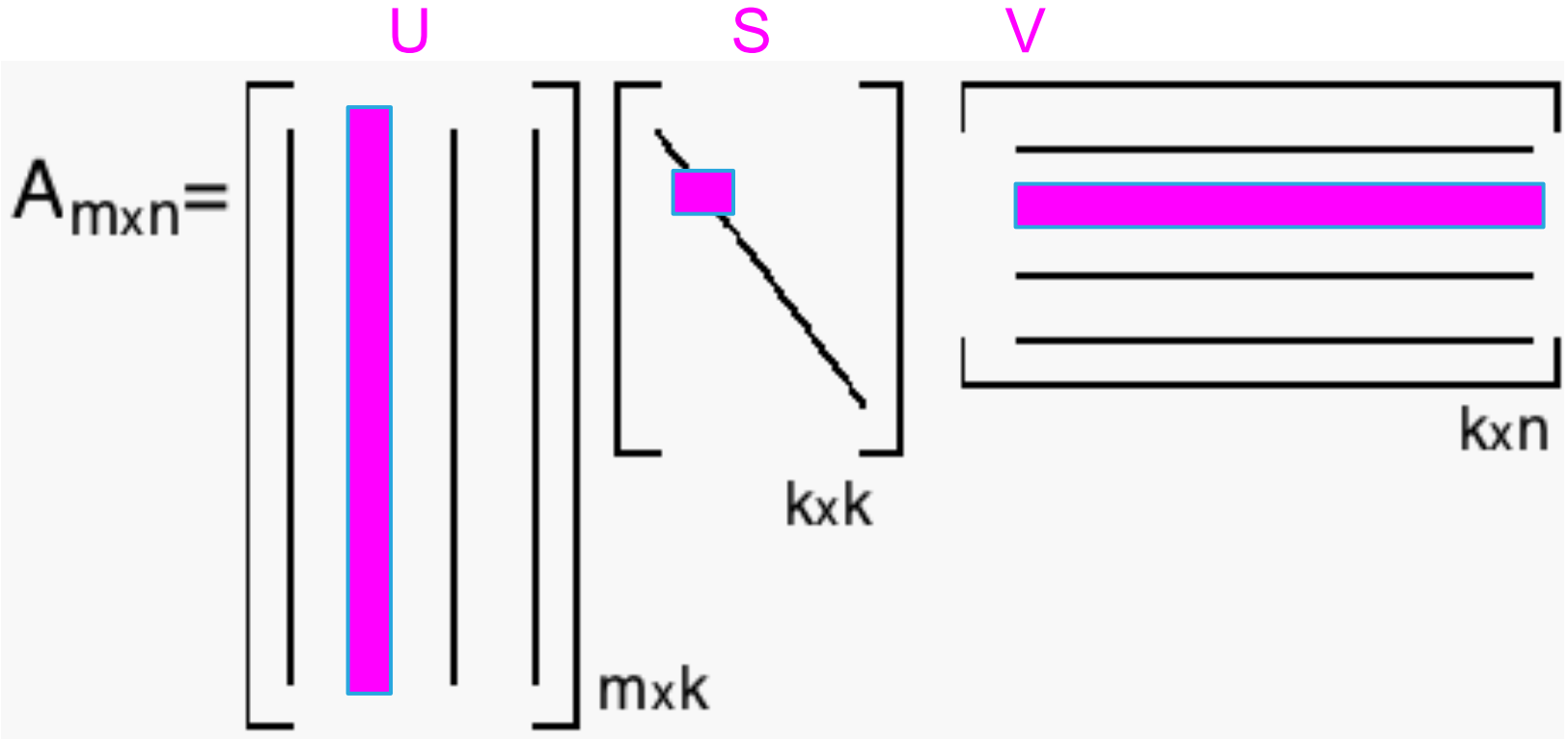


# What are the top “concepts”?

I.e. what dimensions in term-space and document-space explain most of the variance of the data?







```
def topTermsInConcept(concept: Int, numTerms: Int)
  : Seq[(String, Double)] = {
  val v = svd.V.toBreezeMatrix
  val termWeights = v(::, k).toArray.zipWithIndex
  val sorted = termWeights.sortBy(-_._1)
  sorted.take(numTerms)
}
```

```
def topDocsInConcept(concept: Int, numDocs: Int)
  : Seq[Seq[(String, Double)]] = {
  val u = svd.U
  val docWeights =
    u.rows.map(_.toArray(concept)).zipWithUniqueId()
  docWeights.top(numDocs)
}
```

# Concept 1

**Terms**: department, commune, communes, insee, france, see, also, southwestern, oise, marne, moselle, manche, eure, aisne, isère

**Docs**: Communes in France, Saint-Mard, Meurthe-et-Moselle, Saint-Firmin, Meurthe-et-Moselle, Saint-Clément, Meurthe-et-Moselle, Saint-Sardos, Lot-et-Garonne, Saint-Urcisse, Lot-et-Garonne, Saint-Sernin, Lot-et-Garonne, Saint-Robert, Lot-et-Garonne, Saint-Léon, Lot-et-Garonne, Saint-Astier, Lot-et-Garonne

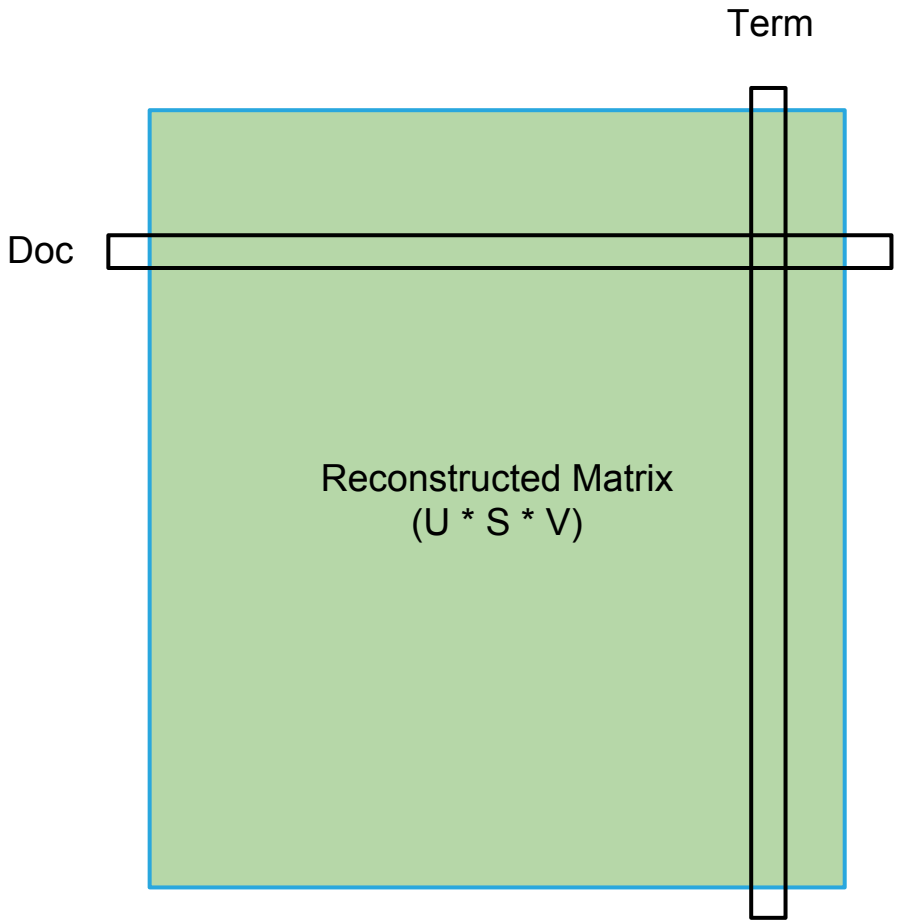
# Concept 2

**Terms**: genus, species, moth, family, lepidoptera, beetle, bulbophyllum, snail, database, natural, find, geometridae, reference, museum, noctuidae

**Docs**: Chelonia (genus), Palea (genus), Argiope (genus), Sphingini, Cribrilinidae, Tahla (genus), Gigartinales, Parapodia (genus), Alpina (moth), Arycanda (moth)

# Querying

- Given a set of terms, find the closest documents in the latent space



```
def topTermsForTerm(
    normalizedVS: BDenseMatrix[Double],
    termId: Int): Seq[(Double, Int)] = {
  val rowVec = new BDenseVector[Double](row(normalizedVS, termId).toArray)
  val termScores = (normalizedVS * rowVec).toArray.zipWithIndex
  termScores.sortBy(-_._1).take(10)
}

val VS = multiplyByDiagonalMatrix(svd.V, svd.s)
val normalizedVS = rowsNormalized(VS)
topTermsForTerm(normalizedVS, id, termIds)
```

# printRelevantTerms("radiohead")

| Term | Similarity |
| --- | --- |
| radiohead | 0.9999999999999993 |
| lyrically | 0.8837403315233519 |
| catchy | 0.8780717902060333 |
| riff | 0.861326571452104 |
| lyricsthe | 0.8460798060853993 |
| lyric | 0.8434937575368959 |
| upbeat | 0.8410212279939793 |

# printRelevantTerms("algorithm")

| Term | Similarity |
|---|---|
| algorithm | 1.000000000000002 |
| heuristic | 0.8773199836391916 |
| compute | 0.8561015487853708 |
| constraint | 0.8370707630657652 |
| optimization | 0.8331940333186296 |
| complexity | 0.823738607119692 |
| algorithmic | 0.8227315888559854 |

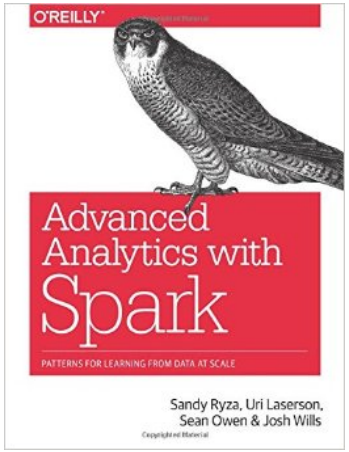
(algorithm,1.000000000000002), (heuristic,0.8773199836391916),
(compute,0.8561015487853708), (constraint,0.8370707630657652),
(optimization,0.8331940333186296), (complexity,0.823738607119692),
(algorithmic,0.8227315888559854), (iterative,0.822364922633442),
(recursive,0.8176921180556759), (minimization,0.8160188481409465)

```
def topDocsForTerm(US: RowMatrix, V: Matrix, termId: Int)
  : Seq[(Double, Long)] = {
  val rowArr = row(V, termId).toArray
  val rowVec = Matrices.dense(termRowArr.length, 1, termRowArr)
  val docScores = US.multiply(termRowVec)
  val allDocWeights = docScores.rows.map(_.toArray(0)).
    zipWithUniqueId()
  allDocWeights.top(10)
}
```

# printRelevantDocs("fir")

| Document | Similarity |
|---|---|
| Silver tree | 0.006292909647173194 |
| See the forest for the trees | 0.004785047583508223 |
| Eucalyptus tree | 0.004592837783089319 |
| Sequoia tree | 0.004497446632469554 |
| Willow tree | 0.004429936059594164 |
| Coniferous tree | 0.004381572286629475 |
| Tulip Tree | 0.004374705020233878 |

# More detail?

- https://github.com/sryza/aas/tree/master/ch06-lsa
- https://spark.apache.org/docs/latest/mllib-dimensionality-reduction.html
- 

cloudera

# Thank you

@sandysifting