STORY DEDUPLICATION AND MUTATION

Antoine Amend (Barclays)
Andrew Morgan (ByteSumo Ltd)

#EUstr9

Challenge

How do we study evolving geopolitics, as it happens, using global news?

It raises questions

How to use global news as a source?

How do we work at this scale?

What design patterns are needed?

What specialist techniques do we need?

How do we trace evolving storylines?

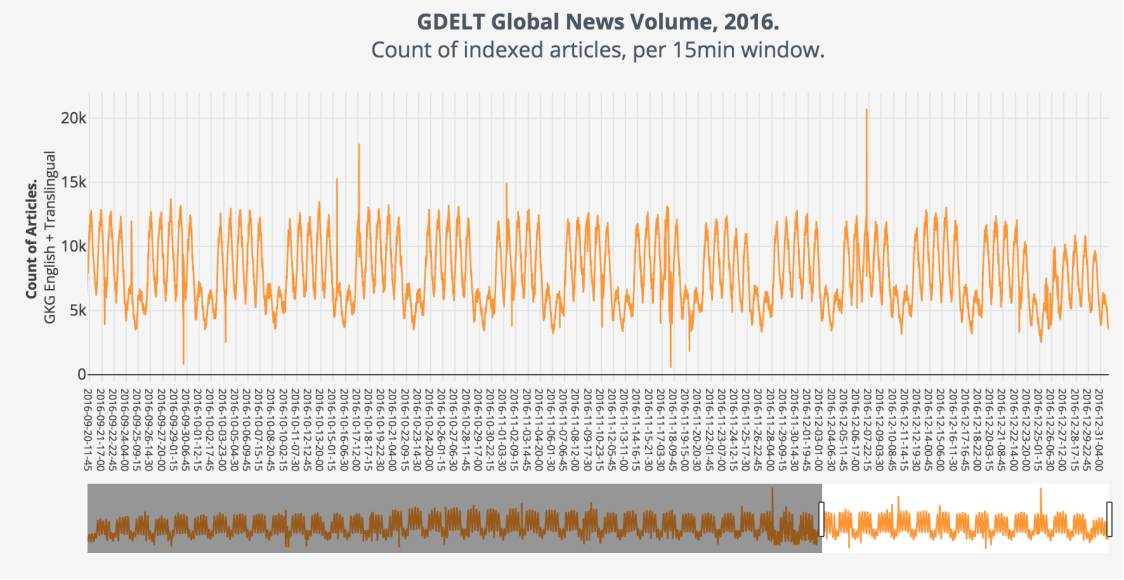
News Data: GDELT

- TLDR: It's big
- global news feed, published every 15min
- ~63k news publishing websites scraped
- ~60 Languages
- ~400k to ~800k articles / month

http://gdeltproject.org
http://data.gdeltproject.org/gdeltv2/masterfilelist.txt

The Data

GKG pipeline emits URLs, tags, sentiments. But not full text.
All languages ~15k articles each 15 minutes.



This chart highlights the global media machine's heartbeat, as news articles are published in all languages, globally.

It's a very regular pattern.

Notice clearly weekday vs weekend volumes. Spikes drive NFRs.

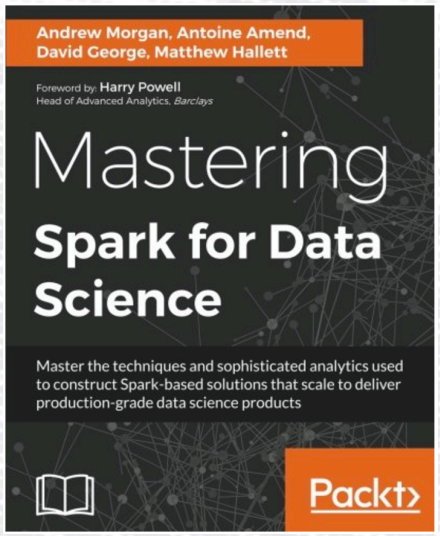
Who are we?

We are engineers doing big data science.

We wrote a text book to teach others about data science at scale.

For it, we wrote lots of new code, much not available elsewhere.

We open-sourced all our code.




Antoine Amend


Andrew Morgan


Matthew Hallett


David George

Our Approach

Source near real-time URLs from GDELT

Scrape article text from source

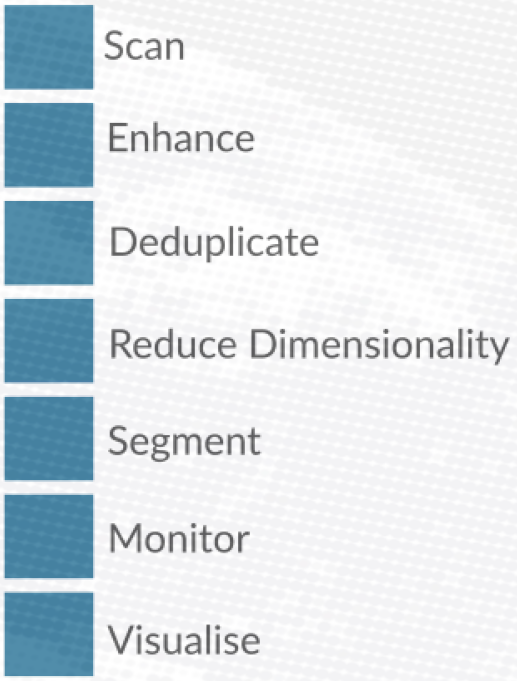
De-duplicate articles

Use "Random Indexing" to build stable features

Streaming K-Means to build top Stories

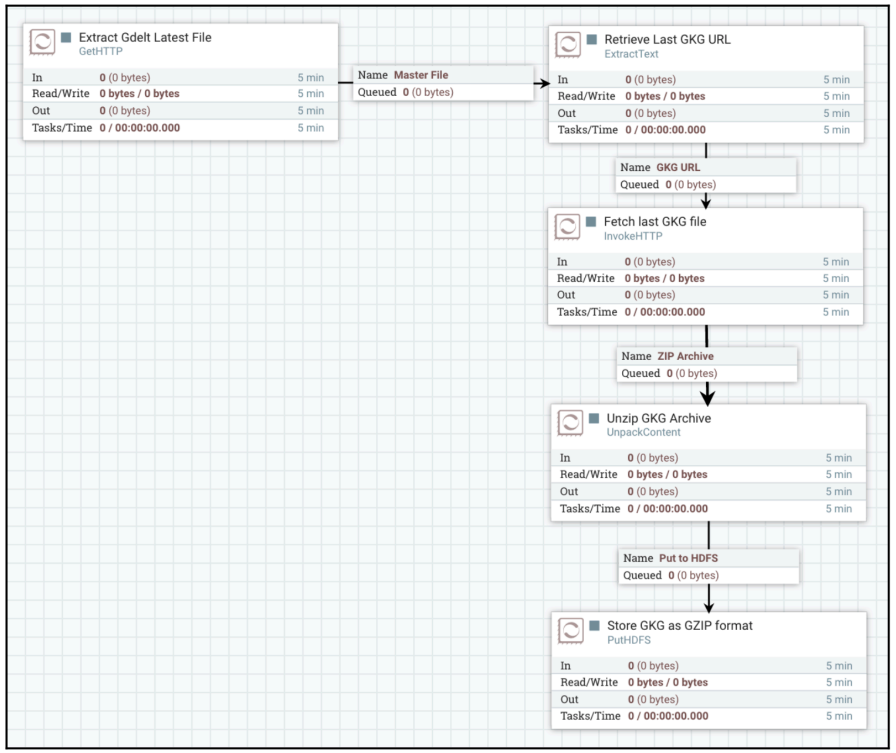
Track Story Mutation, to trace Evolving Geopolitics

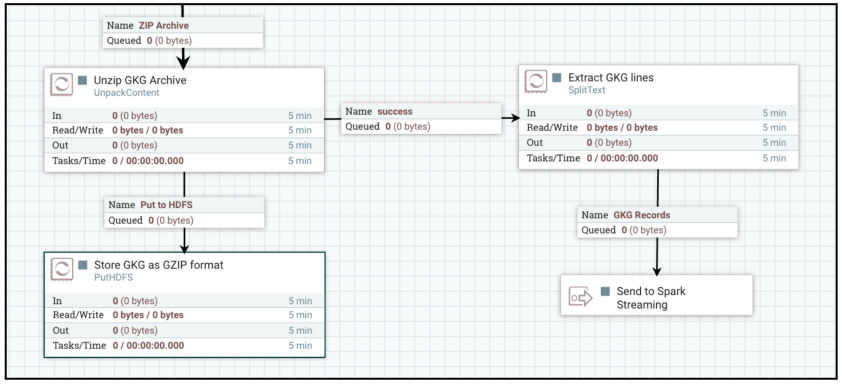
Interpret the analysis



Ingestion pipeline

NiFi + Kafka + Spark Streaming






```scala
def readFromNifi(ssc: StreamingContext): DStream[String] = {

  val nifiConf = new SiteToSiteClient.Builder()
    .url("http://localhost:8080/nifi")
    .portName("SEND_TO_SPARK")
    .buildConfig()

  ssc
    .receiverStream(new NiFiReceiver(nifiConf, StorageLevel.MEMORY_ONLY))
    .map(packet => new String(packet.getContent, StandardCharsets.UTF_8))
}
```

Indexing using Simhash

The secret sauce

shingles	hashcodes												
he	...	1	1	0	0	1	1	1	1	1	1	0	1
el	...	1	1	0	0	1	0	1	0	0	1	1	1
ll	...	1	1	0	1	1	0	0	0	0	0	0	0
lo	...	1	1	0	1	1	0	0	0	0	0	1	1
os	...	1	1	0	1	1	1	1	0	0	1	0	0
si	...	1	1	1	0	0	1	0	1	0	1	1	0
im	...	1	1	0	1	0	0	1	0	0	1	0	0
mh	...	1	1	0	1	1	0	0	1	1	0	1	1
ha	...	1	1	0	1	1	1	1	1	1	0	0	1
as	...	1	1	0	0	0	0	1	1	0	0	1	0
sh	...	1	1	1	0	0	1	0	1	0	1	0	1

text	simhash												
hello simhash	...	1	1	0	1	1	0	1	1	0	1	0	1
hello minhash	...	1	1	0	**0**	1	0	1	1	0	1	0	1
hello world		1	1	0	**0**	1	**1**	1	**0**	**1**	**0**	0	**0**

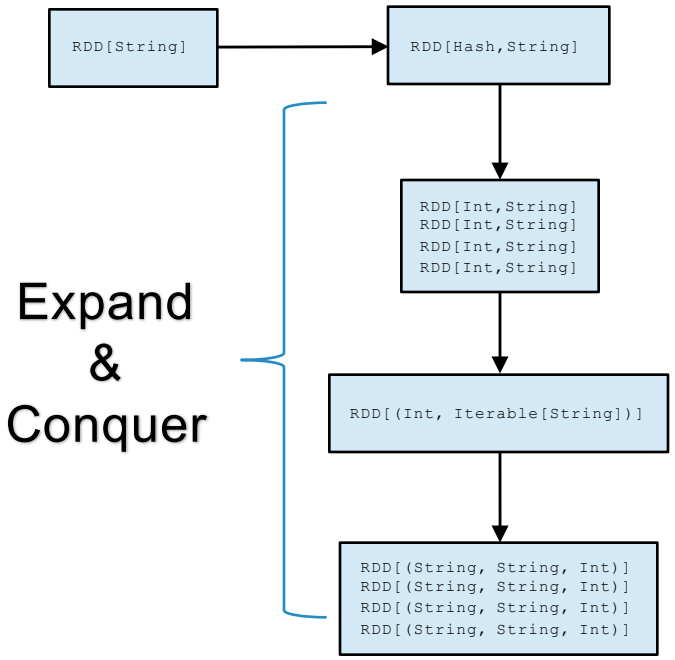
```
hello simhash ^ hello minhash = 1/32
hello simhash ^ hello world = 6/32
```

Fork me on GitHub

```scala
implicit class Simhash(content: String) {

  private def shingles(text: String) = {
    text.replaceAll("\\W", "")
      .toLowerCase()
      .toCharArray
      .sliding(2)
      .map(_.mkString("").hashCode())
      .toArray
  }

  private def isBitSet(i1: Int, bit: Int): Boolean = {
    ((i1 >> bit) & 1) == 1
  }

  def simhash: Int = {
    val aggHash = shingles(content).flatMap({ hash =>
      Range(0, 32).map({ bit =>
        (bit, if (isBitSet(hash, bit)) 1 else -1)
      })
    }).groupBy(_._1).mapValues(_.map(_._2).sum > 0).toArray
    buildSimhash(0, aggHash)
  }

  @tailrec
  private def buildSimhash(simhash: Int, aggBit: Array[(Int, Boolean)]): Int = {
    if (aggBit.isEmpty) return simhash
    val (bit, isSet) = aggBit.head
    val newSimhash = if (isSet) {
      simhash | (1 << bit)
    } else {
      simhash
    }
    buildSimhash(newSimhash, aggBit.tail)
  }

}
```

Deduplication Graph

Detecting near duplicates



Note 7 fiasco leaves Samsung's smartphone brand in question

By ASSOCIATED PRESS
PUBLISHED: 20:02, 12 October 2016 | **UPDATED:** 20:02, 12 October 2016

Share

SEOUL, South Korea (AP) — The fiasco of Samsung's fire-prone Galaxy Note 7 smartphones — and its stumbling response to the problem — has left consumers from Shanghai to New York reconsidering how they feel about the South Korean tech giant and its products.

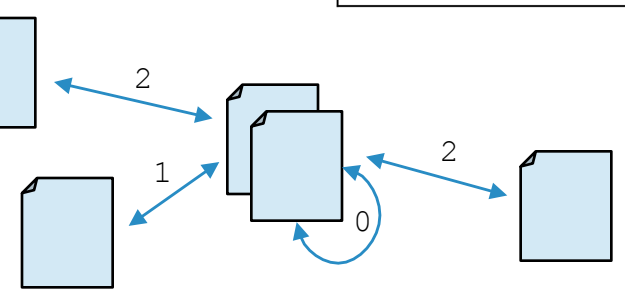
Samsung Electronics said this week that it would stop making the Note 7 for good, after first recalling some devices and then recalling their replacements , too. Now,

Note 7 Fiasco Leaves Samsung's Smartphone Brand in Question

A visitor passes by an advertisement of the Samsung Electronics Galaxy Note 7 smartphone at its shop in Seoul, South Korea, Oct. 11, 2016.

SEOUL, SOUTH KOREA — The fiasco of Samsung's fire-prone Galaxy Note 7 smartphones – and its stumbling response to the problem – has left consumers from Shanghai to New York reconsidering how they feel about the South Korean tech giant and its products.

Related



Dimensionality reduction

Hashing + Random Indexing = stable streaming

- TF-IDF is commonly used as a feature vector
- Oxford dictionary contains 170K words
- We need dimensionality reduction!
  - **PCA**
    - Computationally intensive
    - Not fit for streaming, where future words are unknown
  - **Hashing TF**
    - Fixed number of features (2^{20} by default), fit for streaming
    - Collisions leads to dramatic overestimates
  - **Random Indexing**
    - Preserved distance measure (as per Johnson-Lindenstrauss Lemma)
    - Combined with Hashing TF, guarantees fixed number of dimensions embedded in lower vector space

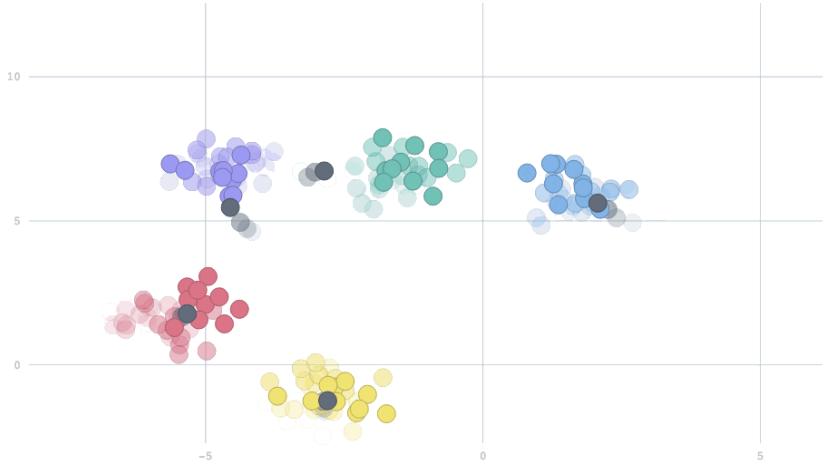
	Random contexts								
story	1	0	0	0	0	1	1	0	-1
deduplication	0	1	0	0	1	1	1	-1	0
mutation	1	0	0	0	1	1	-1	0	1
spark	0	0	0	0	0	0	0	1	0
streaming	-1	1	1	0	1	1	0	0	-1

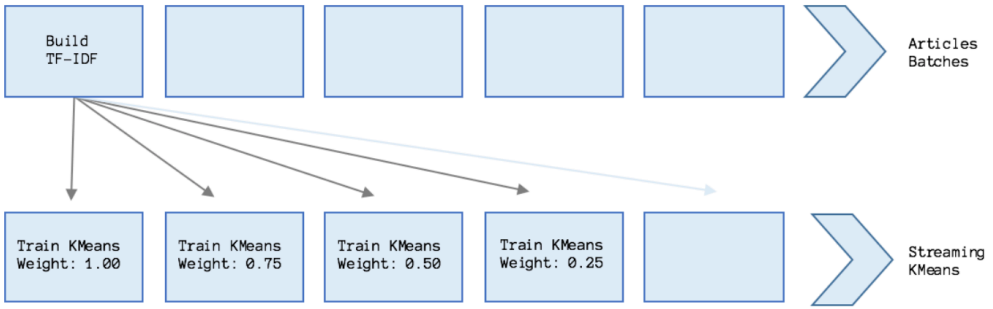
	Aggregated context								
TF	1	2	1	0	3	4	1	0	-1

https://github.com/derrickburns/generalized-kmeans-clustering
https://www.sics.se/~mange/papers/RI_intro.pdf

Streaming Kmeans

Remember the past, drift towards the future



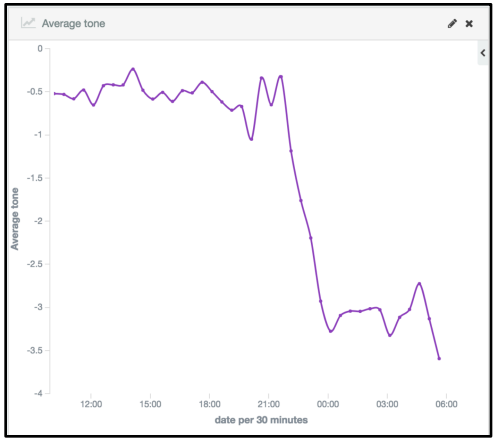


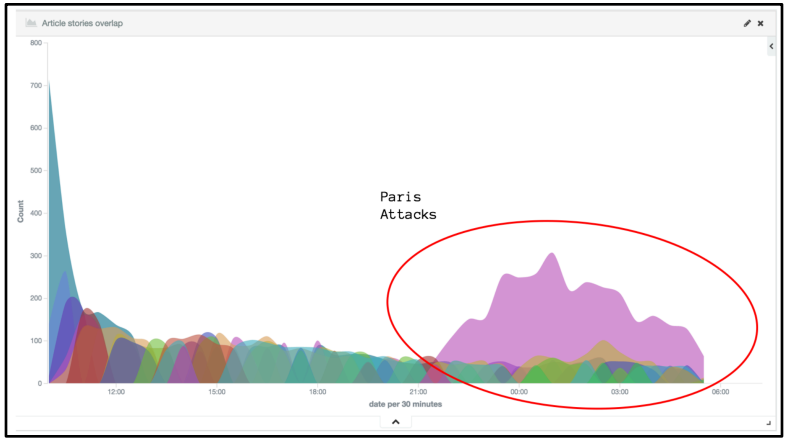
- Defining k = 10 (number of topics at a given time)
- Defining forgetfulness parameter = **1h**
  - **Infinite decay:** we remember all the past, our code is more defensive towards fake news
  - **Zero decay:** our clusters are re-defined at every batch, we can't track stories over time
- Cluster centers adjust to new articles, allowing stories to drift over time

Story drift

Tracking stories over time

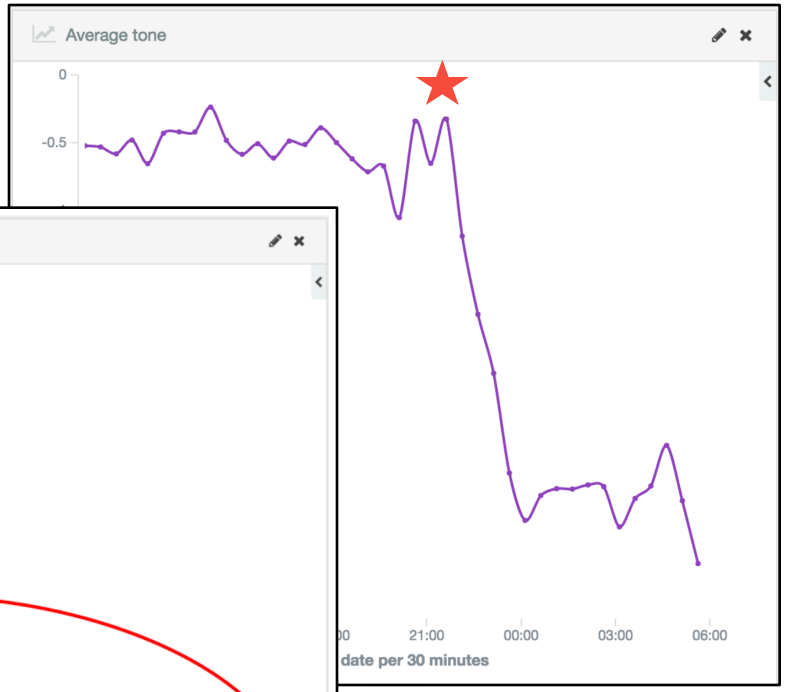
- Index each article + simhash + predicted cluster
- Detected Paris attack articles on Nov. 15' at 9:30pm
- Story burst from the [TERROR / VIOLENCE] topic
- We track GDELT metadata over time
- We ensure homogeneous clusters (RI embedding)

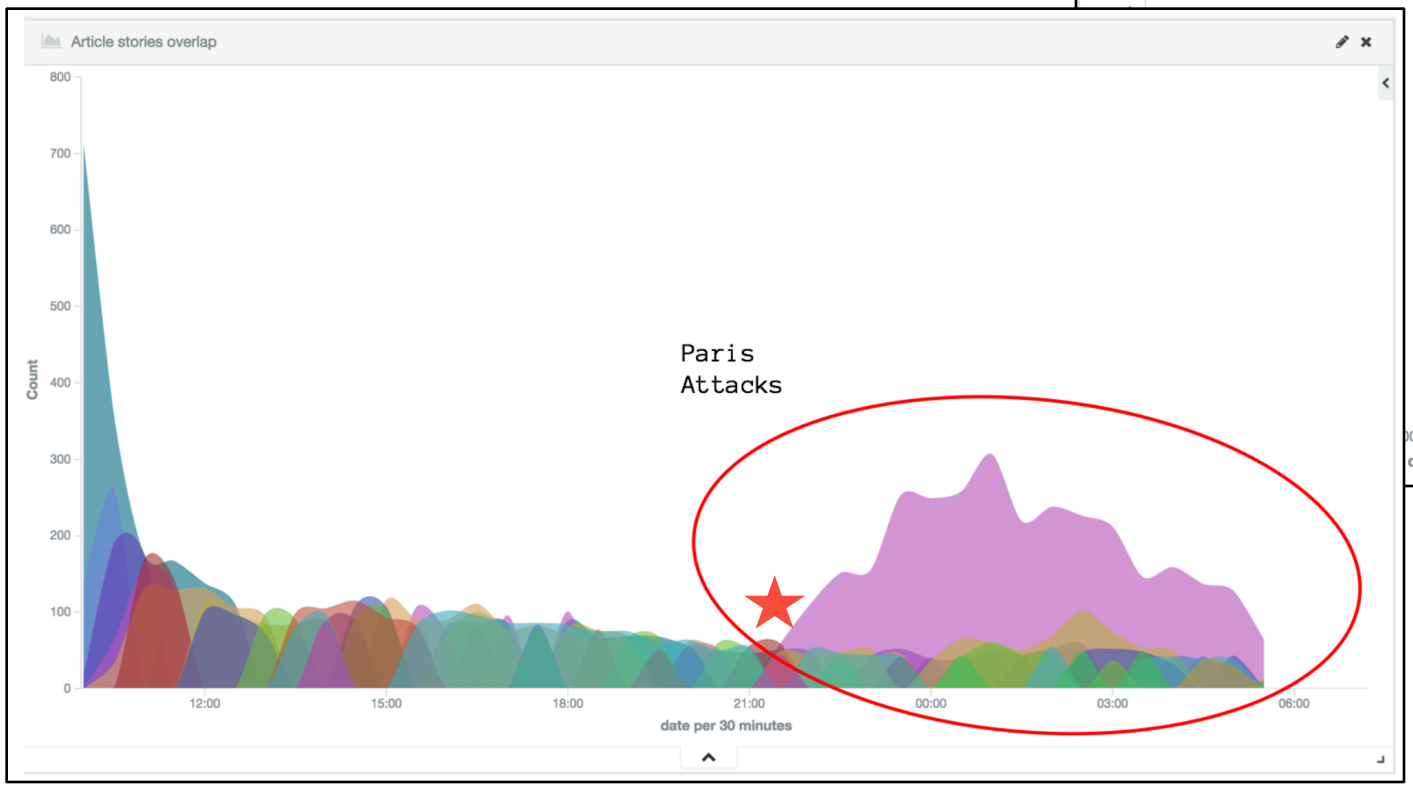




Story drift:

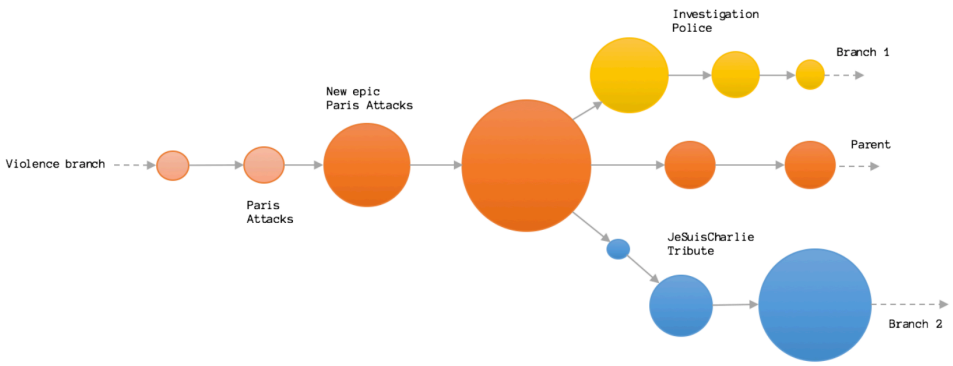
Tracking the Paris attacks

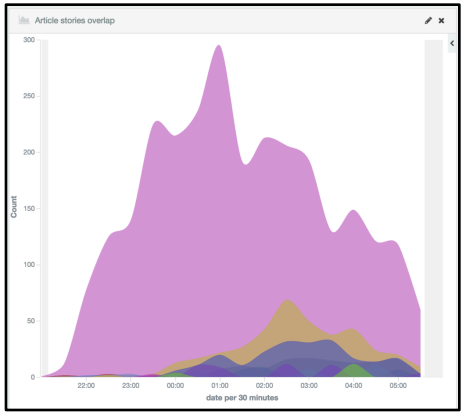




Recycling dying clusters
Evolving geopolitics

- Searching for “paris attack” shows additional clusters after 1h
  - Spark streaming started to recycle dying clusters
  - All eyes are now on breaking news, ignoring the least popular topics
- Topic modelling (LDA) exhibits different “flavors” of a same event
  - Main cluster stays focused on the fact
  - Second most important is around social media
  - Third one is about politics and tributes paid



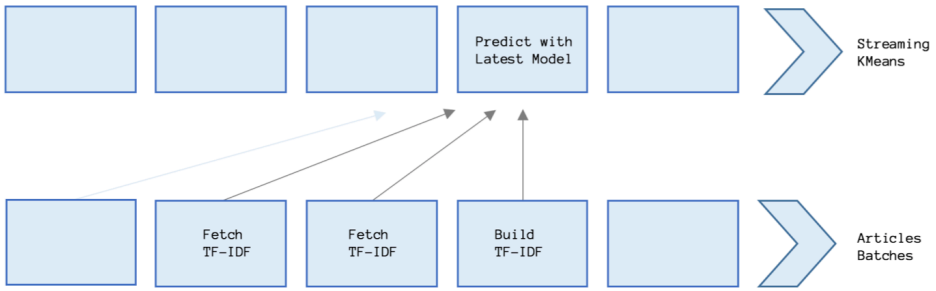


Connecting stories

Branching / Merging model

*If we observe many articles at a time **t** that belonged to a cluster **S** and now belong to a cluster **S'** at a time **t+dt**, then **S** migrated to **S'** in **dt** time.*

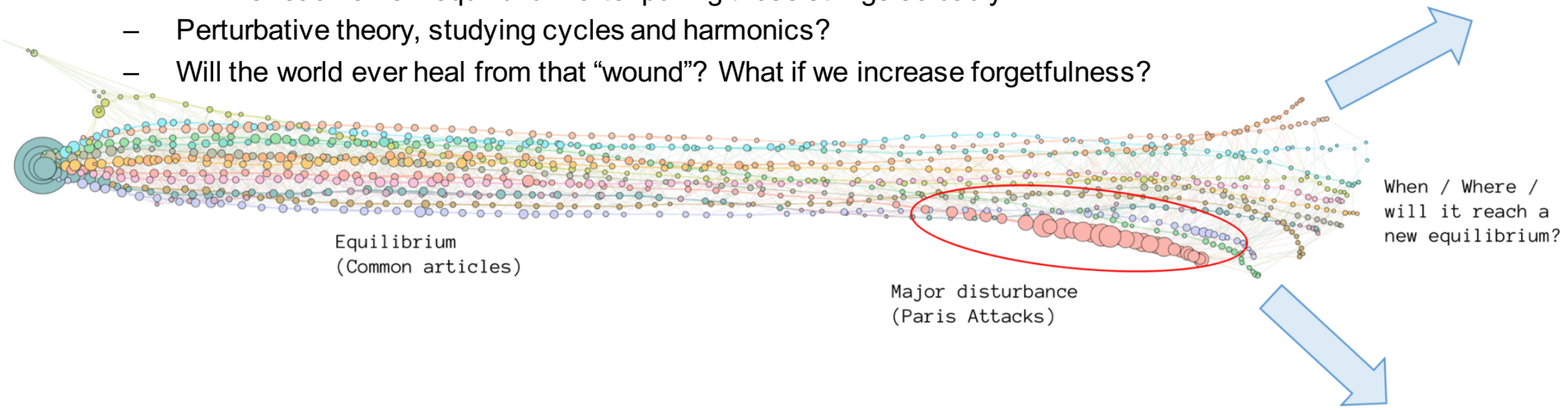
- We store our feature vectors alongside the articles
- We predict outdated vectors against an up-to-date model
- We save these connections as Edges and explore branching as **a weighted directed graph**



Watching the world unfold

A fragile equilibrium

- Before the attack, the world fitted into 10 distinct topics
  - The attack recycled dying clusters and created new flavors of a same event
  - The extra coverage on Paris pushed other topics upwards, resulting in this scatter shape
- The Paris attack perturbed an equilibrium, reshaped the vision of the world
  - Will we reach a new equilibrium after pulling those strings so badly?
  - Perturbative theory, studying cycles and harmonics?
  - Will the world ever heal from that “wound”? What if we increase forgetfulness?

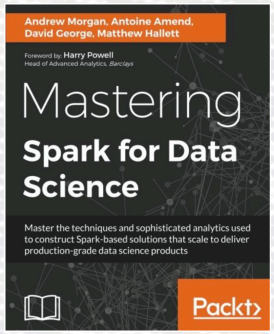


Your Challenge

Thanks!

All the code needed to try this at home is here:

https://github.com/PacktPublishing/Mastering-Spark-for-Data-Science



Antoine Amend

Matthew Hallett

Andrew Morgan

David George