Blocking strategies to accelerate record matching for Big Data integration

Ivan Kadochnikov - kadivas@jinr.ru
Vladimir Papoyan - vlpapoyan@jinr.ru

NEC 2019

Record matching

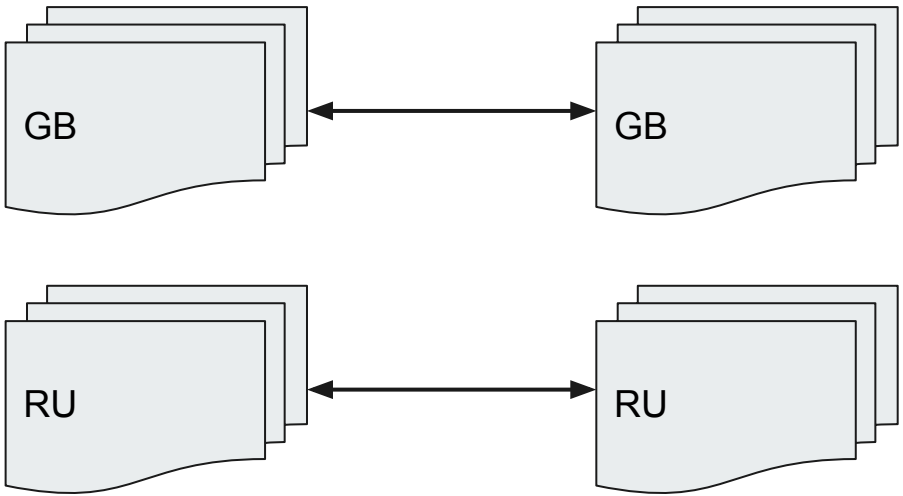
id	Name	Address	….
123456789	Example LTD	1 Sample road ...	….

number	Legal name	Country	….
98765-432	Example, limited	UK	….

String distance (edit distance)

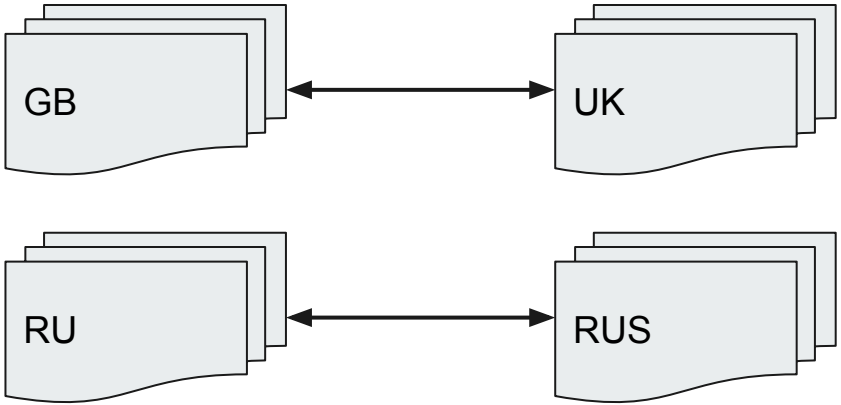
Levenshtein(
‘Example limited’,
‘Exams unlimited’
) = 5

Attribute-based blocking



Attribute-based blocking

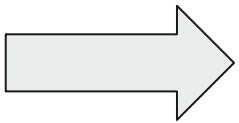
- Needs robust categorical attributes shared between datasets

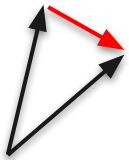


Nearest-neighbour search with LSH

- Fast approximation of string distance

“Example”
“Exemples”





TF/IDF to transform string into vectors

- Term frequency × inverse document frequency
- Document = short string (name)
- Term = n-gram of tokens
- Tokens = individual characters
- Sparse 2^{15}dimensional vector result

$$IDF(t, D) = \log \frac{|D| + 1}{DF(t, D) + 1},$$

Location-sensitive hashing

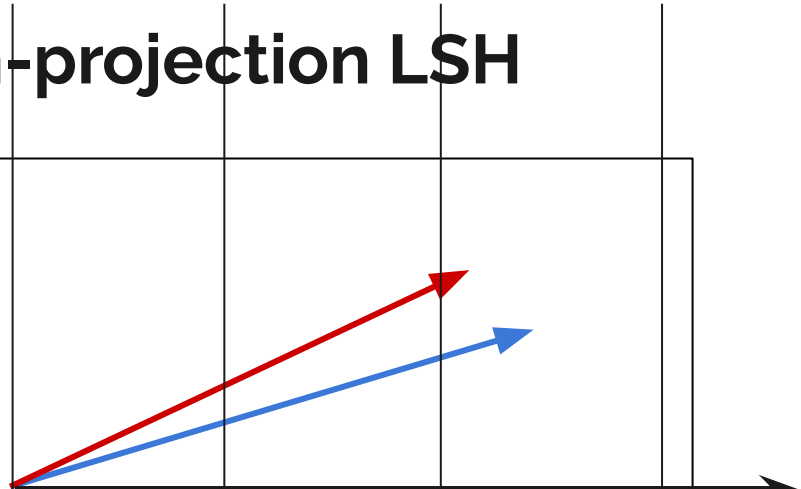
- Project many-dimensional vector into hash buckets
- For a specific metric, “close” vectors are likely to have the same hash
- “Far” vectors are not likely to have the same hash
- OR-amplification
- AND-amplification

NNS in vector space with LSH

- Project full latent vector space into hash buckets
- By definition, close vectors probably fall in the same bucket
- Compute pairwise distance in full vector space inside each bucket

Euclidean distance random-projection LSH

- Random projection direction
- Project every vector onto d



Spark-ml implementation

- Part of Spark
- Spark DataFrame API
- Euclidean or Jaccard distance (no cosine distance hash)
- OR-amplification of NNS only
- Unstable when buckets contain too many records

ScANNS implementation

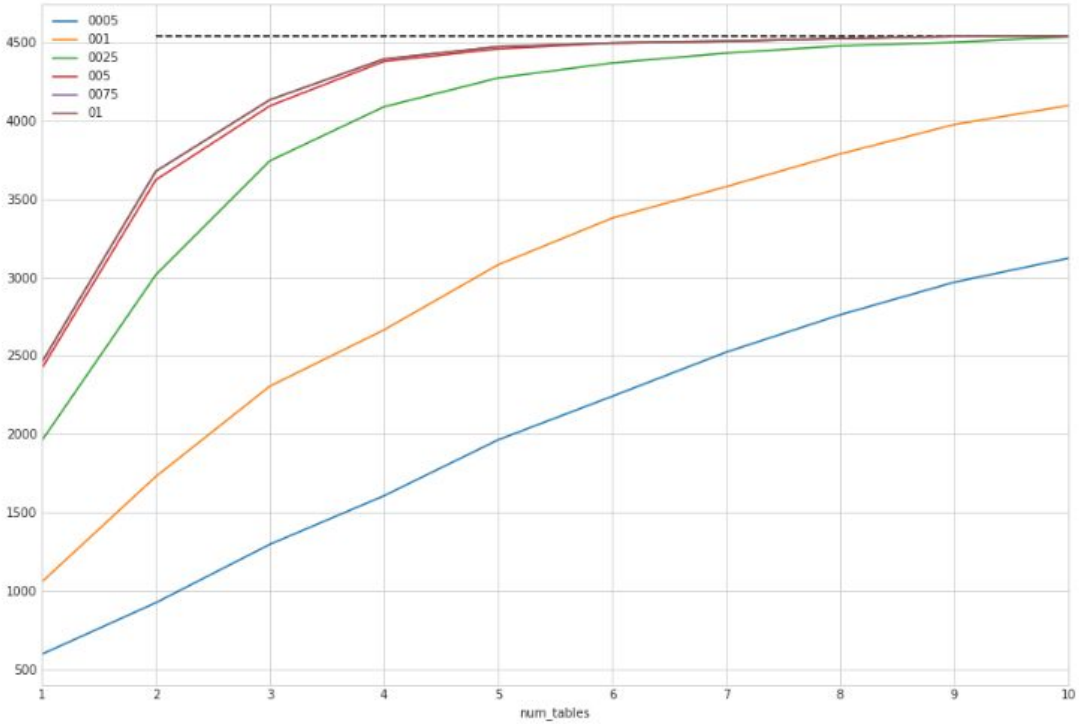
Scalable Approximate Nearest Neighbor Search:

- Spark library using RDD Spark API
- Open-sourced in Feb 2017, needs a fix to work
- Developed at LinkedIn
- Euclidean, Jaccard and cosine distance hashes.
- Both And- and Or-amplification of hashes
- Drops records from overfull buckets

Linking CompaniesHouse and GLEIF datasets

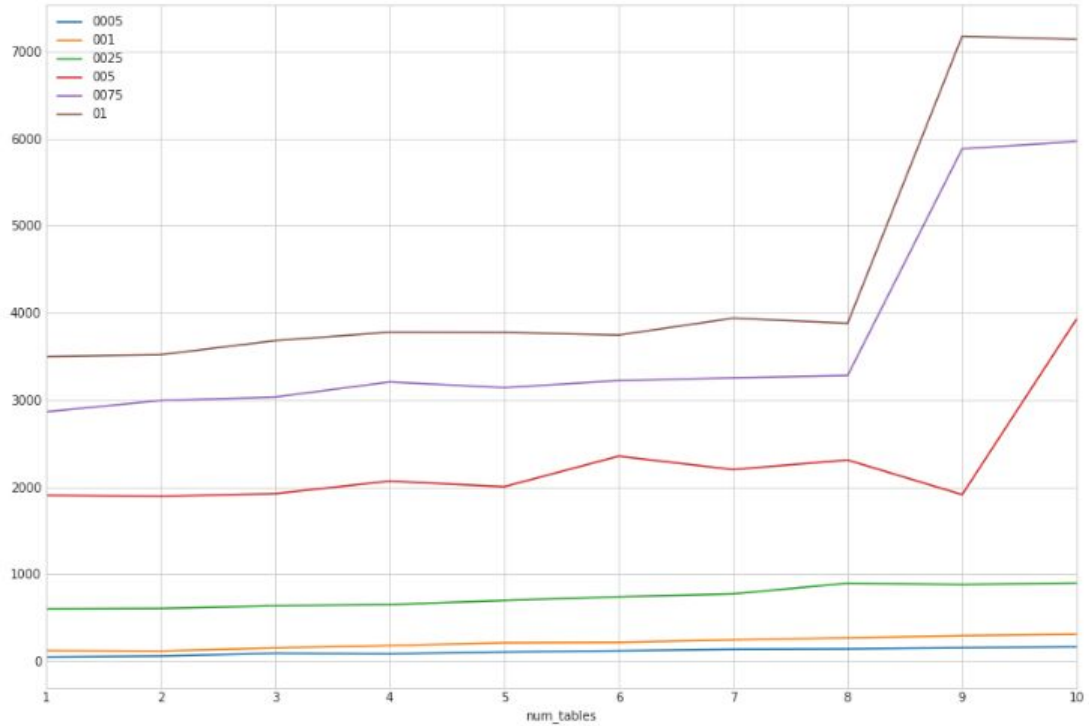
- 4206355 records in CompaniesHouse, 1208110 records in GLEIF
- Record matching on company name
- Euclidean distance < 11.0, character triples
- One Spark node on 16 cores and 18GB of memory
- 0.01 sample from CH, 0.1 sample from GLEIF for the graphs

OR-amplification (matches vs n_hashes)



Computing time vs n_hashes

t, seconds



Results

- Implemented LSH-based blocking method for record matching in Spark
- Demonstrated effectiveness of using and-amplification
- Implemented blocking with Scanns LSH
- To do:
  - Use larger cluster
  - Test on a labeled record-matching dataset
  - Measure Scanns performance

Thank you!

- This work is supported by the Russian Science Foundation under grant 19-71-30008