Distributed Classification of Text Streams: Limitations, Challenges, and Solutions

Artem Trofimov (St. Petersburg State University / JetBrains Research),

Nikita Sokolov (ITMO University),

Mikhail Shavkunov (National Research University Higher School of Economics),

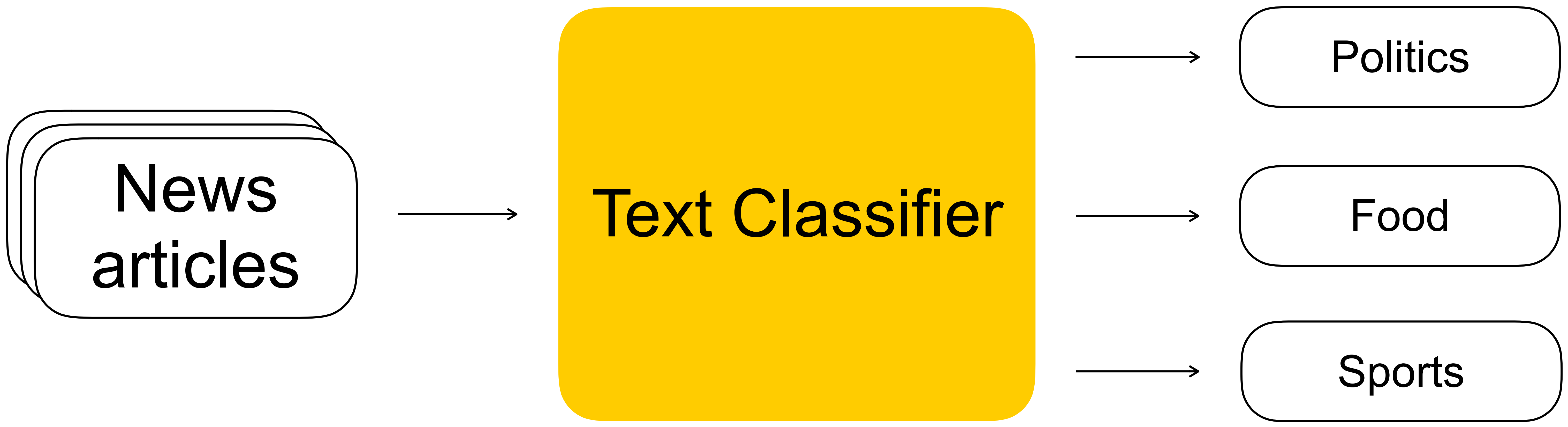
Igor Kuralenok (Yandex),

Boris Novikov (National Research University Higher School of Economics)

Classification of text streams: an example

News articles classification

- Multi-classification problem
- Automated news aggregation
- Event-based decisions



Classification of text streams: requirements

Production-ready solution

- Scalability
- Low latency
- Reproducibility
- Fault tolerance

Existing tools: python libraries, batch systems

```
>>> from sklearn.linear_model import SGDClassifier
>>> text_clf = Pipeline([
...     ('vect', CountVectorizer()),
...     ('tfidf', TfidfTransformer()),
...     ('clf', SGDClassifier(loss='hinge', penalty='l2',
...                           alpha=1e-3, random_state=42,
...                           max_iter=5, tol=None)),
... ])
```

```
>>> text_clf.fit(twenty_train.data, twenty_train.target)
Pipeline(...)
>>> predicted = text_clf.predict(docs_test)
>>> np.mean(predicted == twenty_test.target)
0.9101...
```

hadoop

MAHOUT

Not scalable

High latency

Text classifier on top of distributed stream processing

Existing tools: distributed stream processing

Apache SAMOA
Scalable Advanced Massive Online Analysis

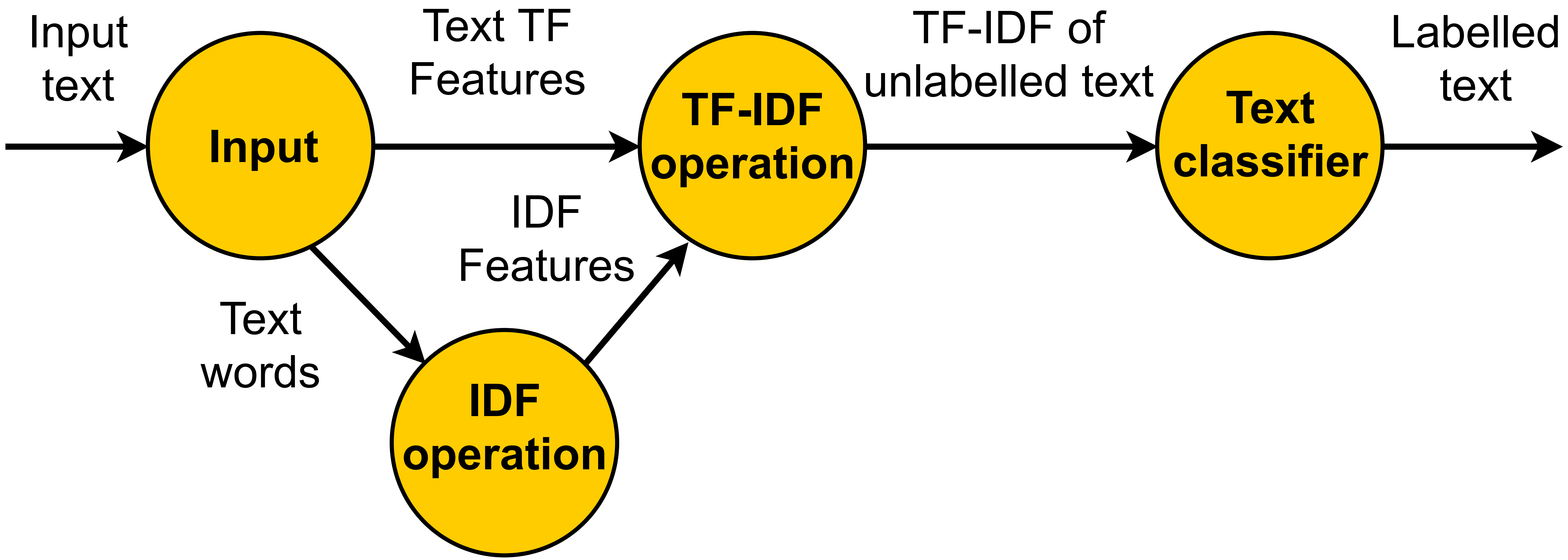
Spark Streaming

APACHE STORM™

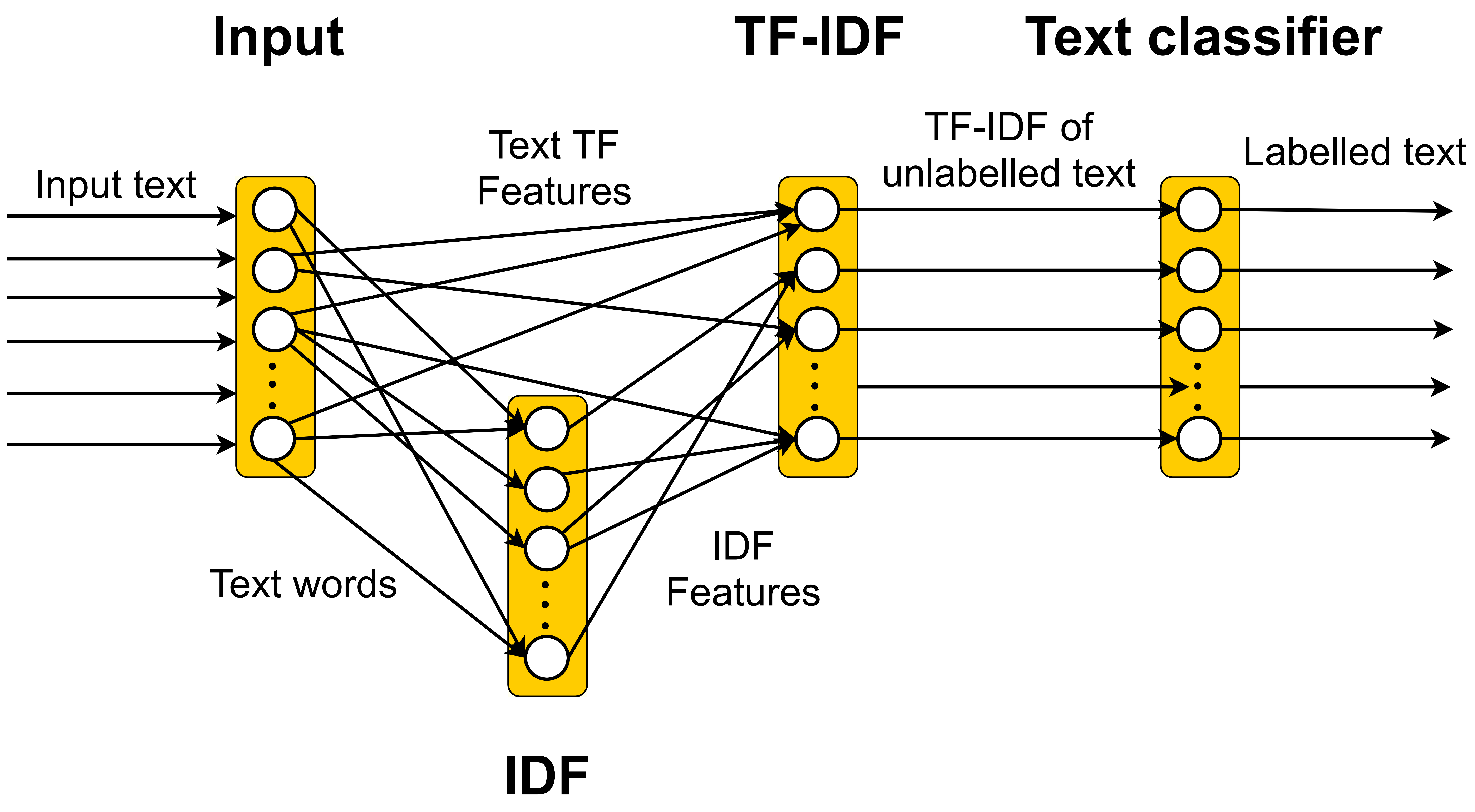
MLlib
The Machine Learning Library

Stream processing: building logical graph

- Bag-of-words model
- TF-IDF features



Stream processing: physical graph



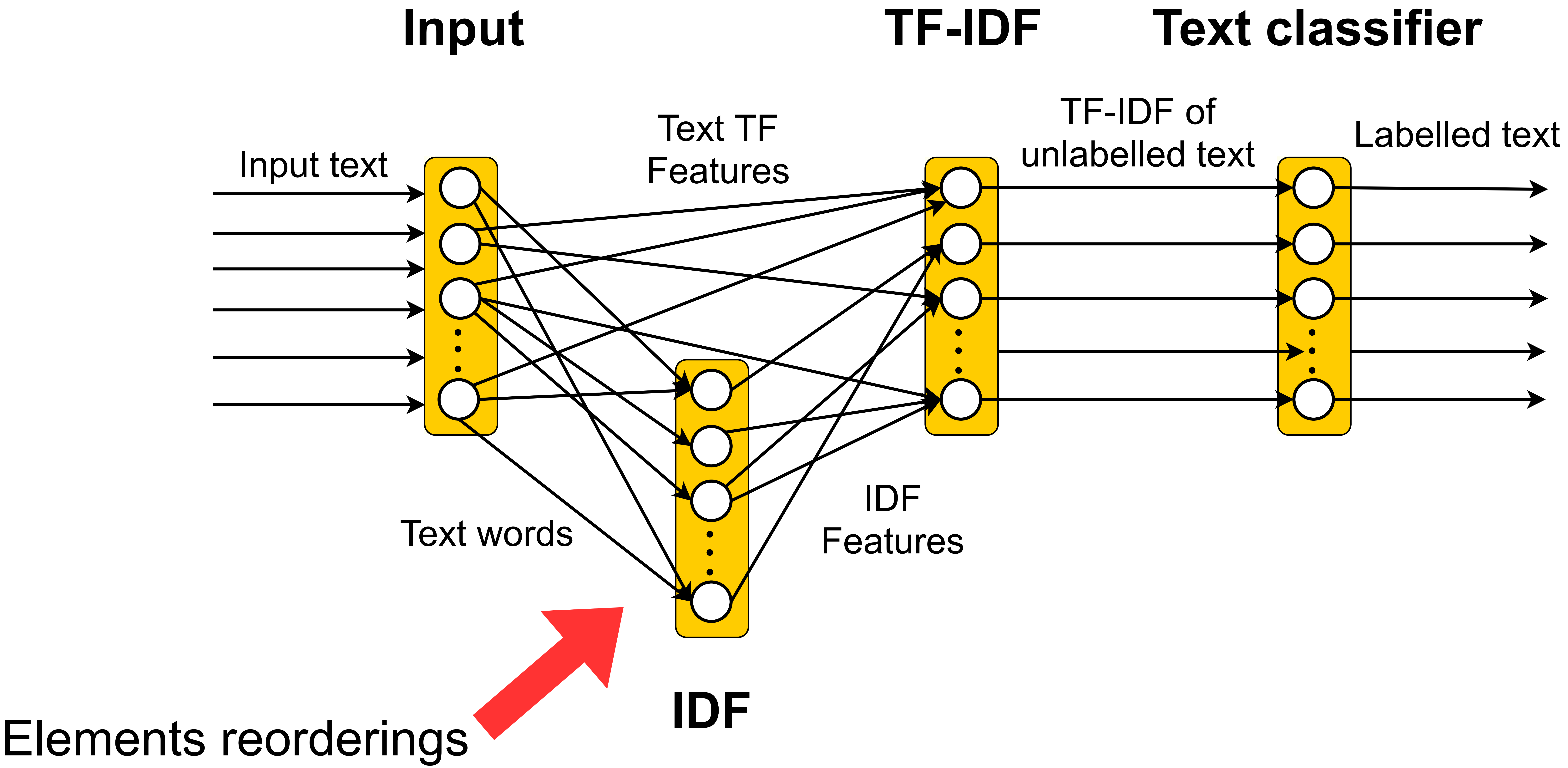
Text classification on top of distributed stream processing system

Perfect solution? Let's try!

- Apache Flink
- Multinomial logistic regression (less than 1% regret in comparison with SVM)
- lenta.ru dataset: 200.000 articles, 90 classes
- Amazon EC2 small instances

Reproducibility

Pitfall #1: races in dataflow



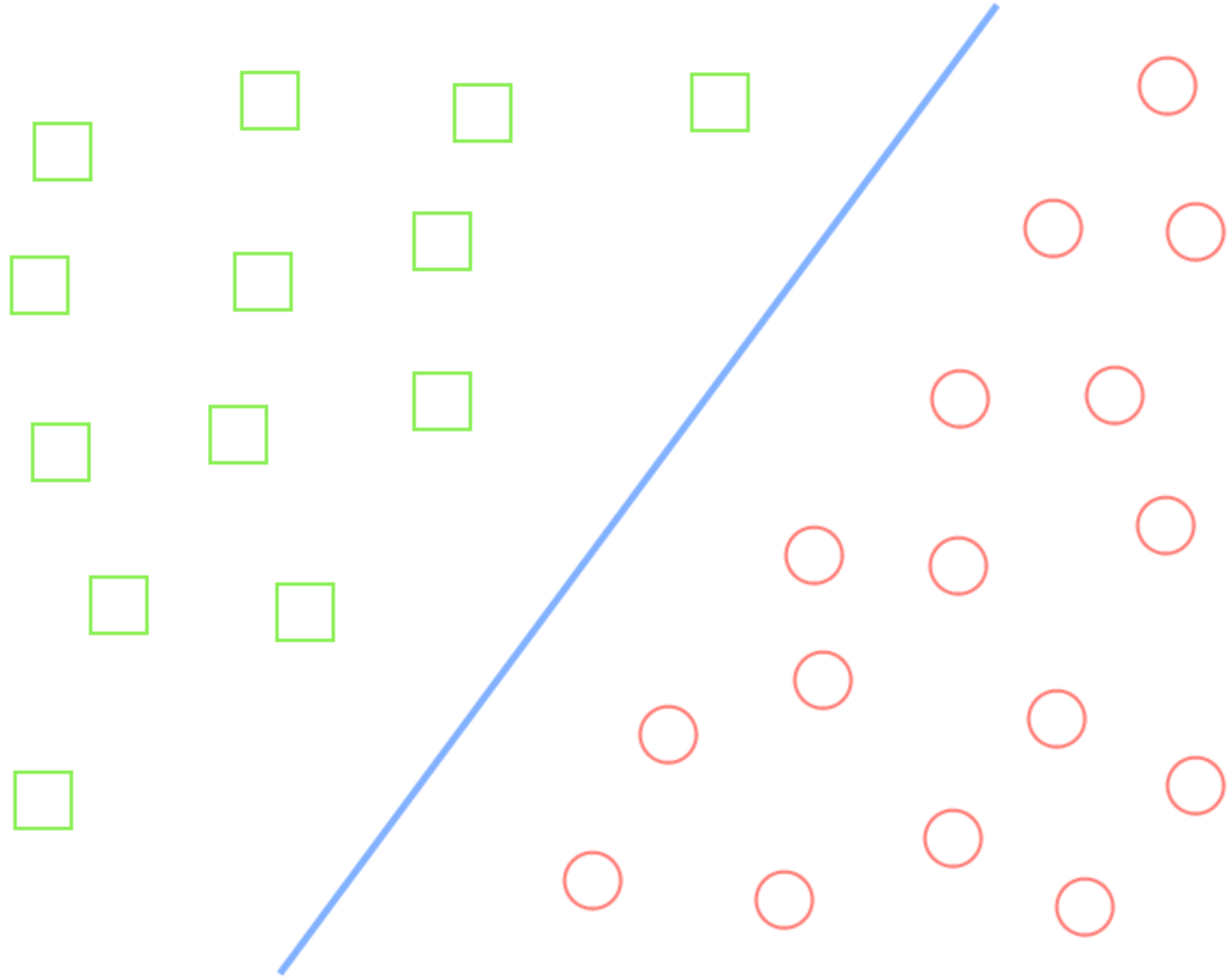
Races in dataflow: experiments

- 10 independent launches
- Results of individual points can be irreproducible

Cluster size	% of varied labels(mean±std)	Accuracy % (77.3)
2	0.9 ± 0.2	77.3 ± 0.2
4	1.7 ± 0.4	77.3 ± 0.2
8	1.9 ± 0.5	77.3 ± 0.2

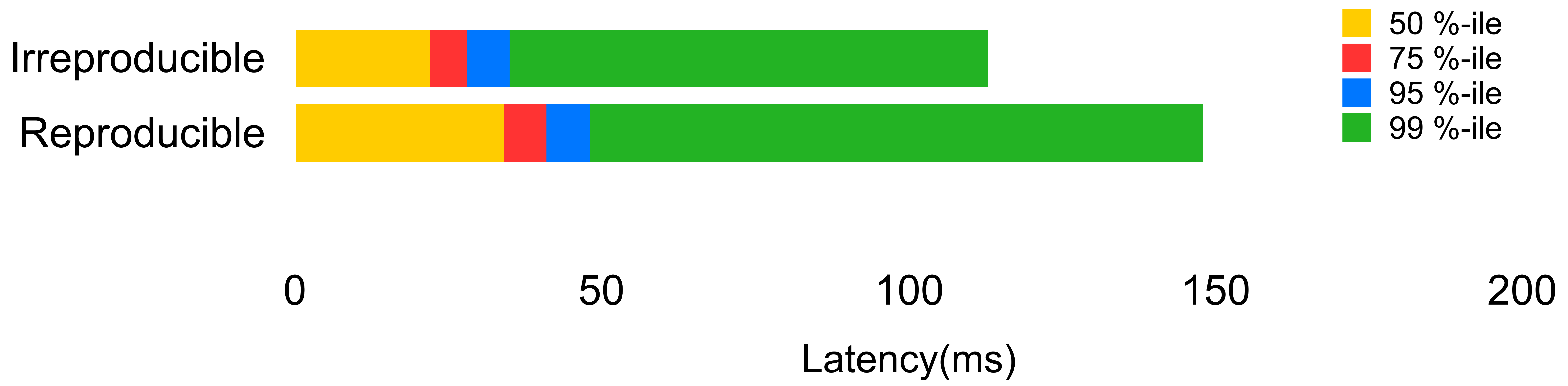
Races in dataflow: experiment explanation

- Differently labeled points are near discriminative surface
- Dependency from classifier accuracy requires further investigation



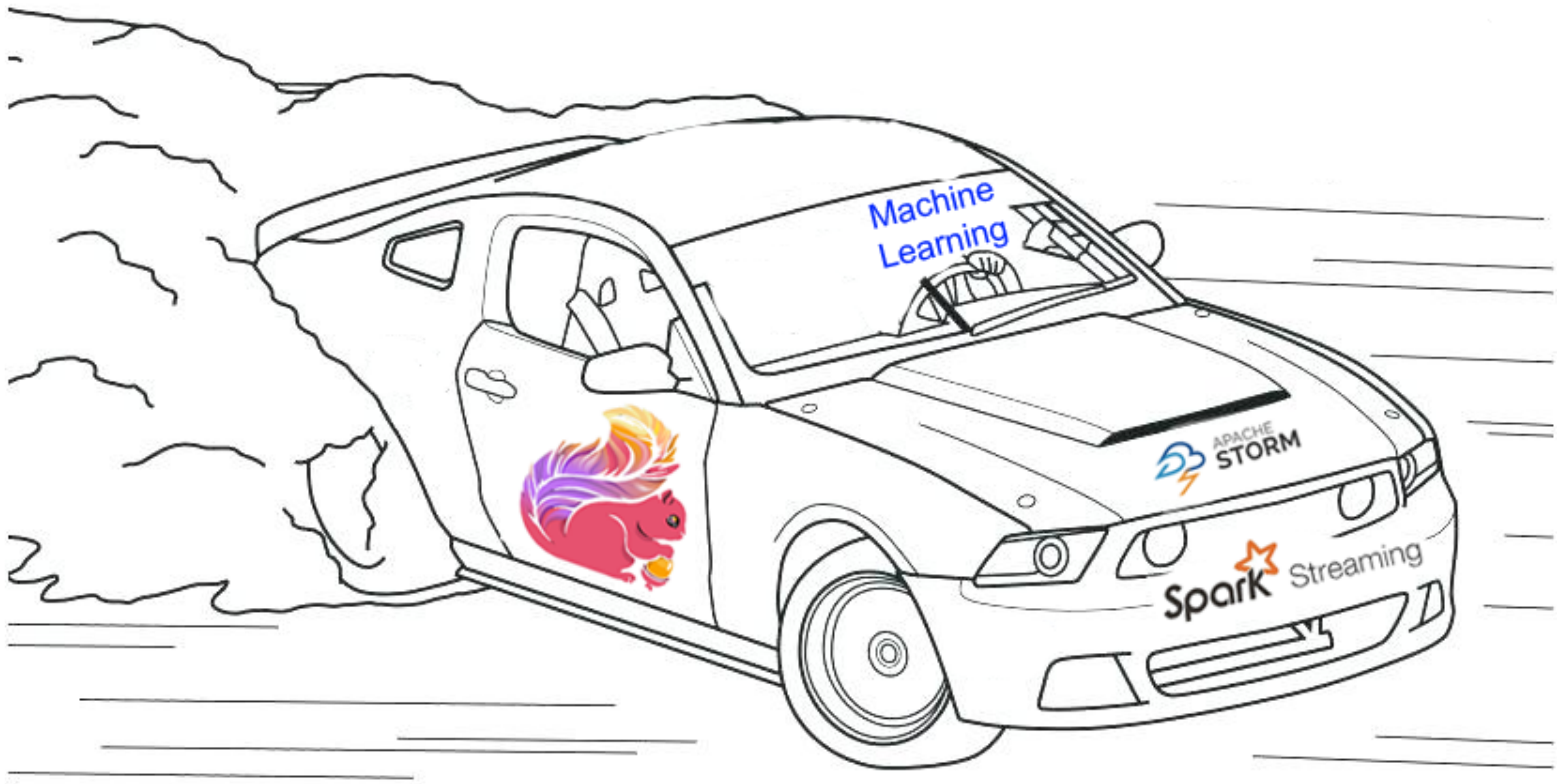
Races in dataflow: straightforward resolution

- Set order on elements
- Buffer until punctuation (watermark) arrives
- Sort according to the order



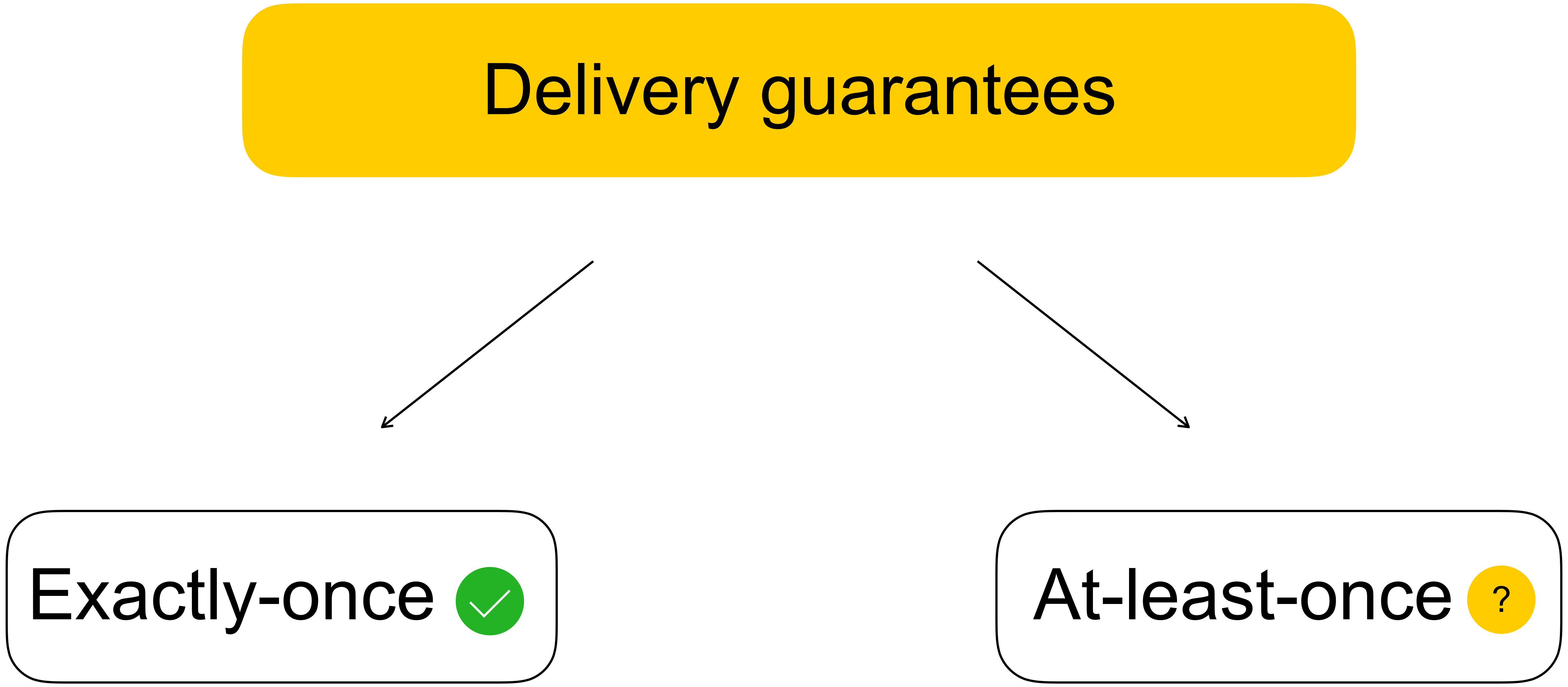
Races in dataflow: overview

- Can affect the result of individual documents, but not accuracy
- Should be considered if reproducibility in terms of individual elements matters
- Straightforward solution has slight latency overhead



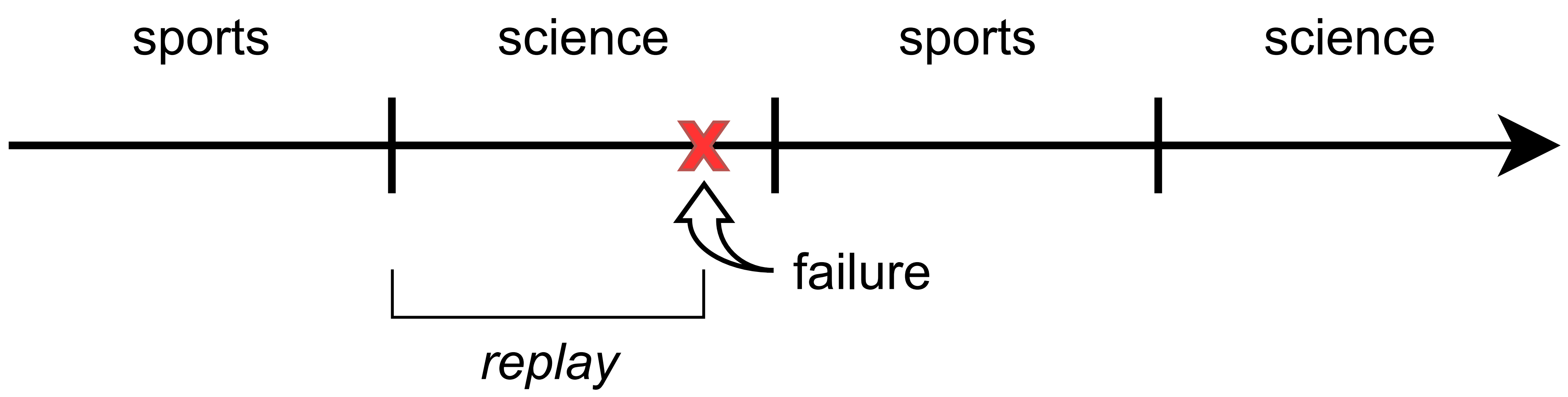
Fault tolerance

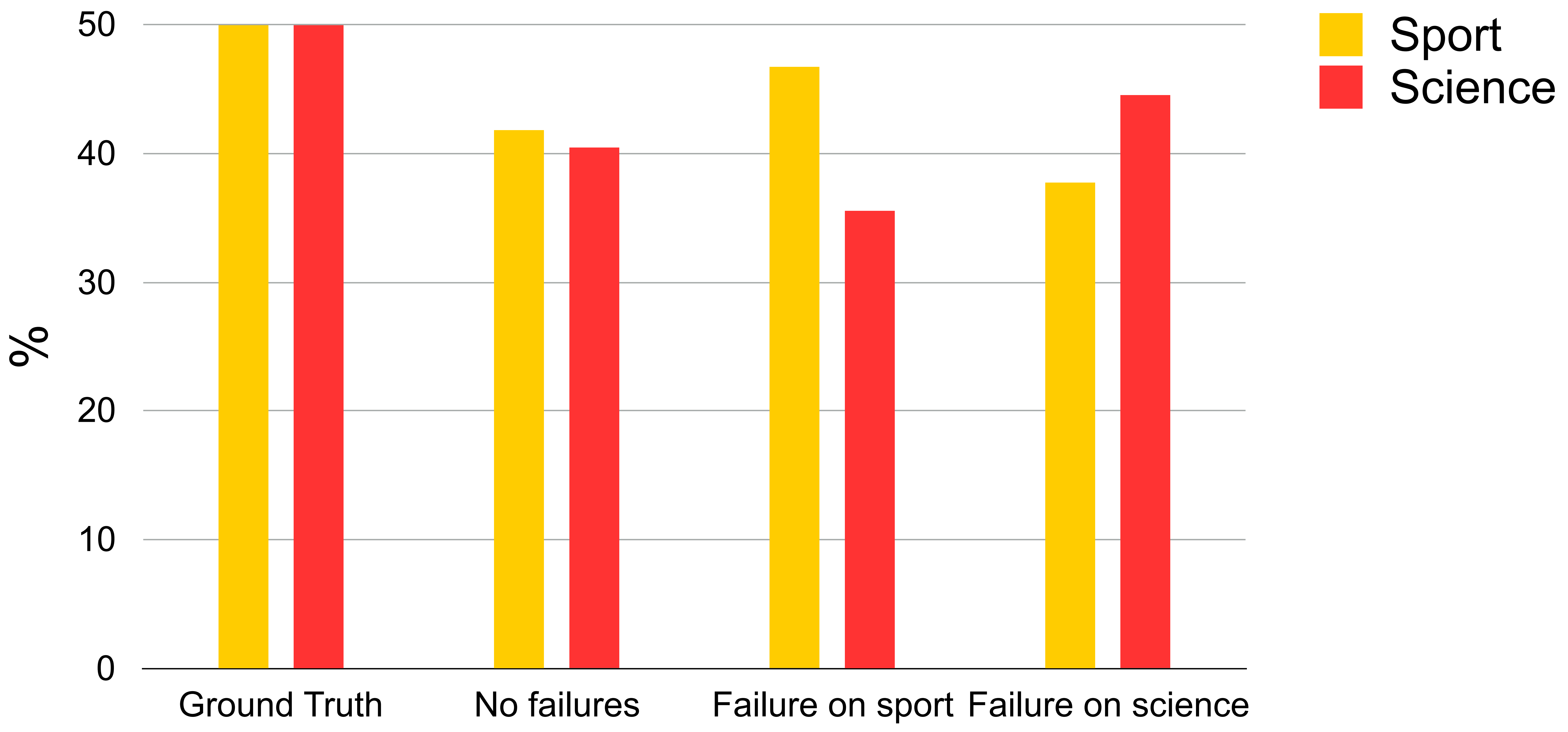
Pitfall #2: the choice of delivery guarantee



At-least-once: biased distribution

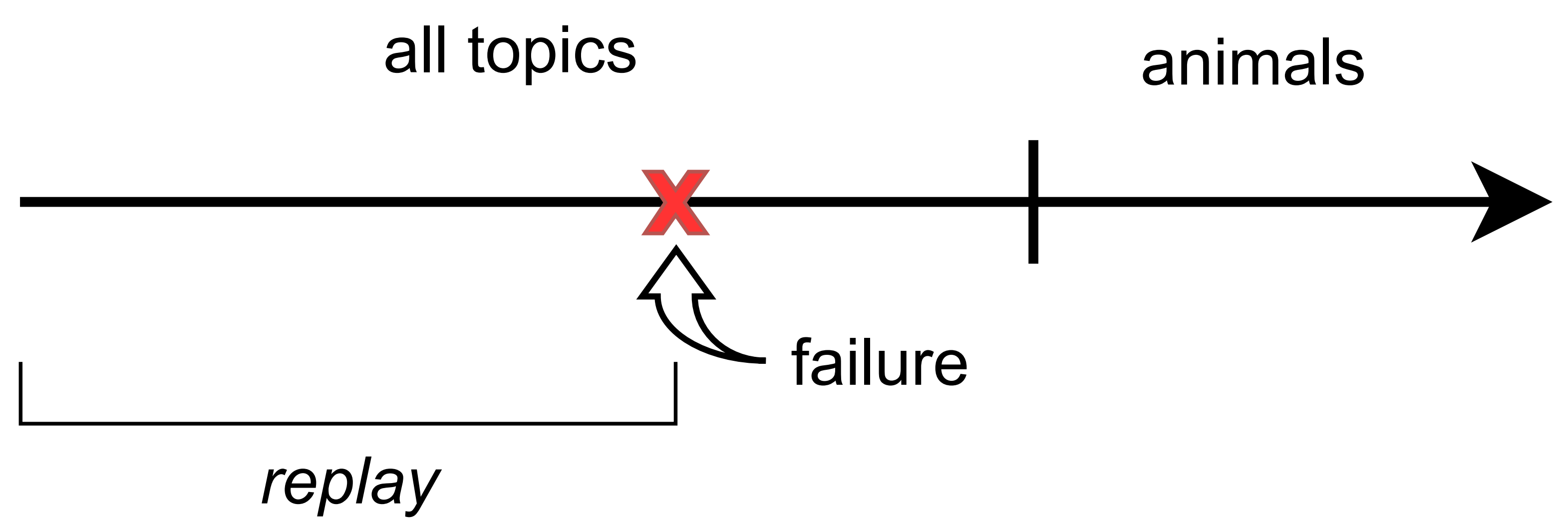
- 4000 articles (window)
- Sport and science topics
- 2 Amazon EC2 small instances

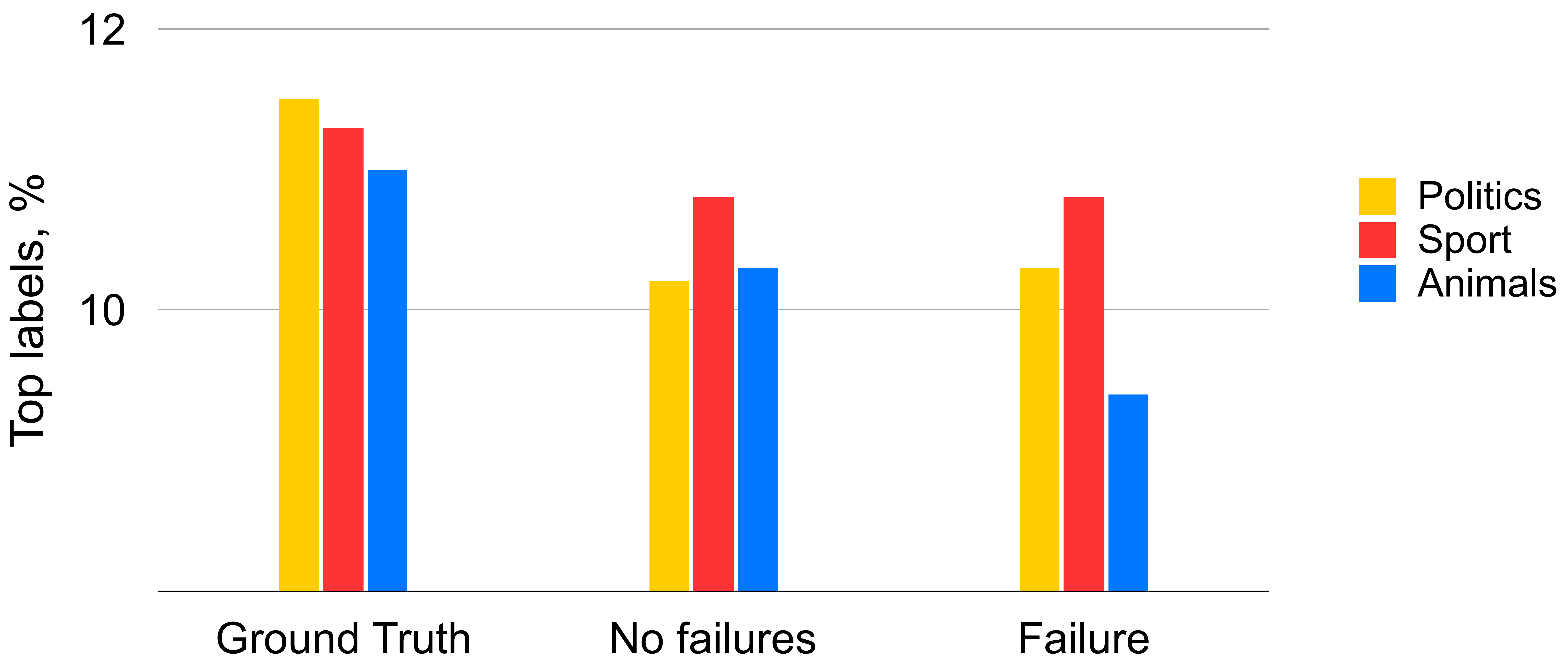




At-least-once: biased threshold

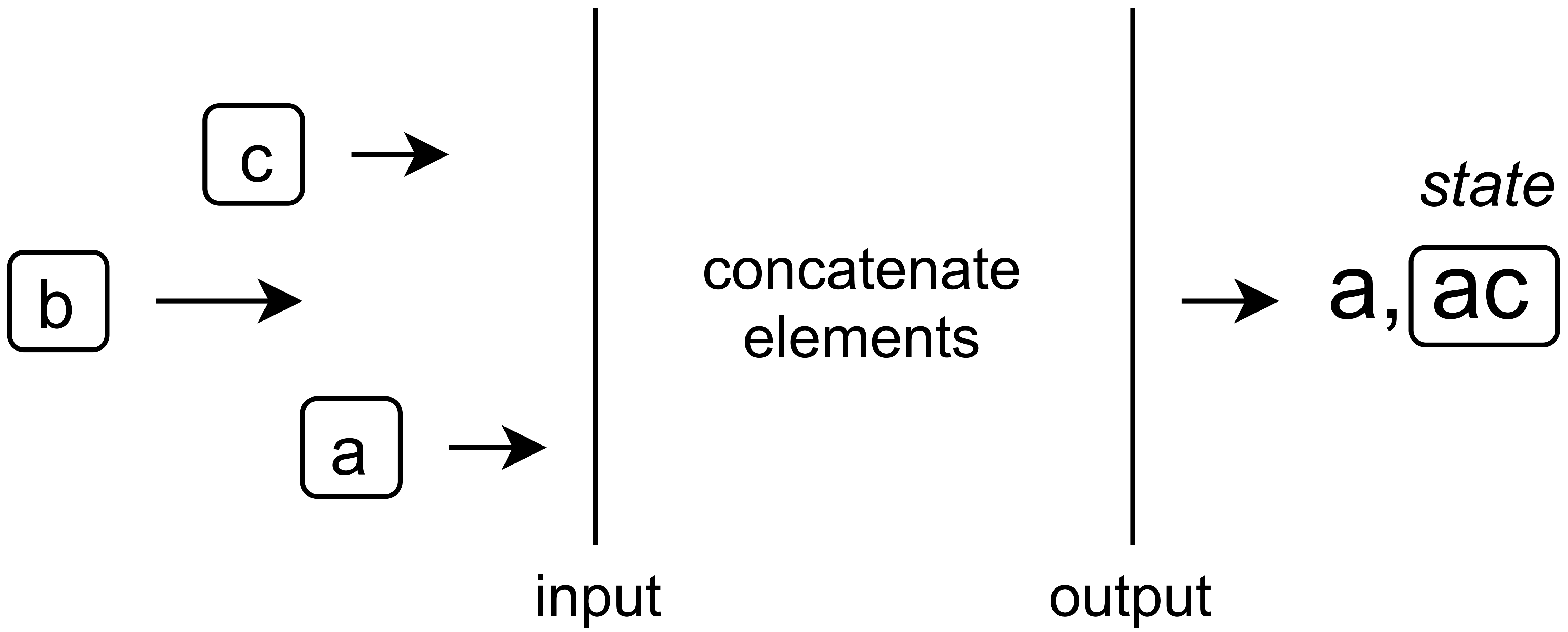
- 5000 articles (window)
- Looking for “popular” topics
- 2 Amazon EC2 small instances





Overhead on exactly once: causes

- Latency depends on snapshotting period in non-deterministic systems



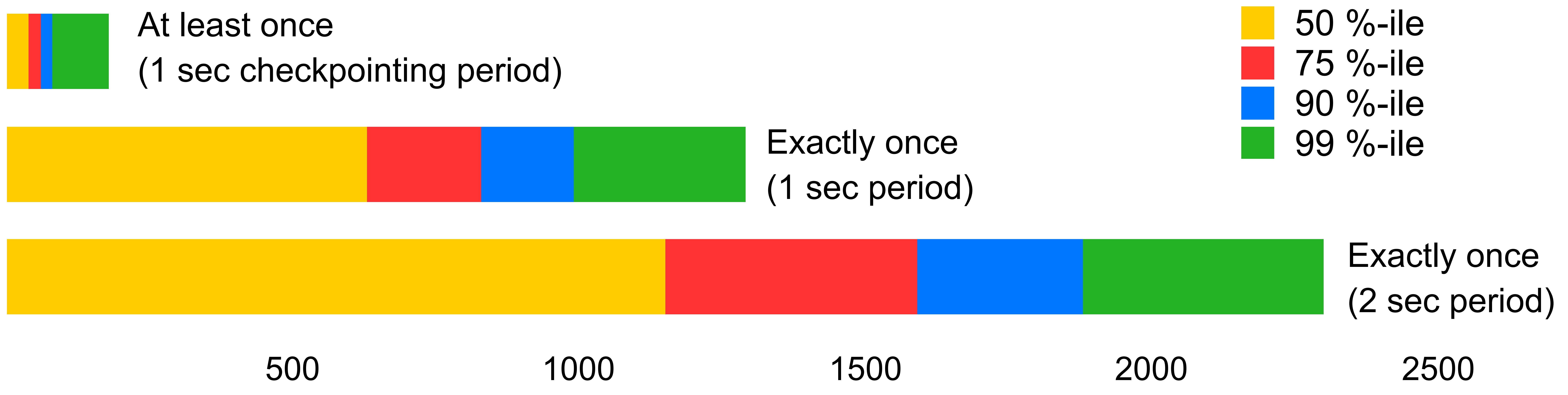
failure ⟶ restore state

restored state: $\boxed{\text{ca}}$

output: a, ac, bca

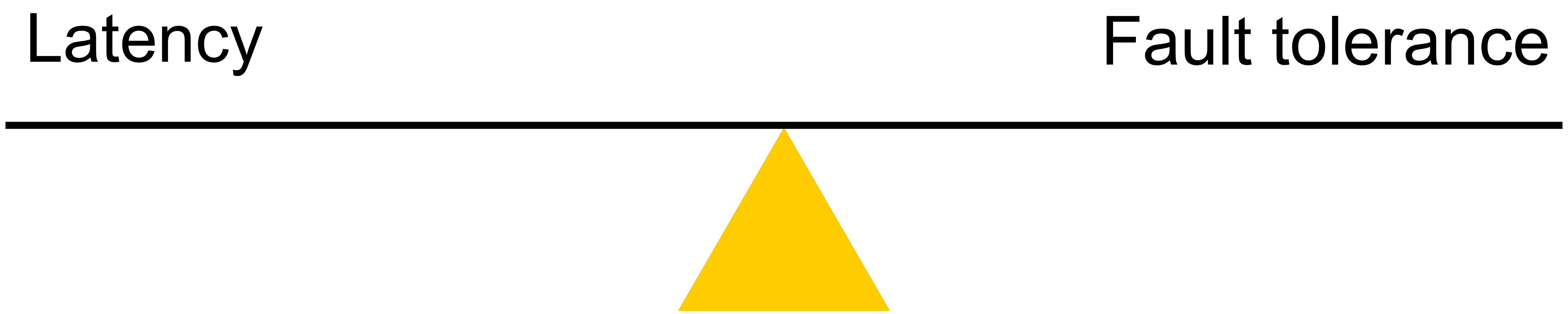
Overhead on exactly once: experiment

- Apache Flink
- 1 and 2 seconds period between snapshots
- 2 Amazon EC2 small instances



Fault tolerance: overview

- Failures within at-least-once guarantee can significantly influence results
- Overhead on exactly-once is high in state-of-the-art stream processing systems



Promising directions

System	Exactly-once	Determinism	Latency
Storm	–	–	low (< 500 ms)
Heron	–	–	low
Samza	–	–	low
Apache Spark	+	+	high
Flink	+	–	high*
MillWheel	+	+	NA
FlameStream	+	+	low

* - with enabled exactly-once

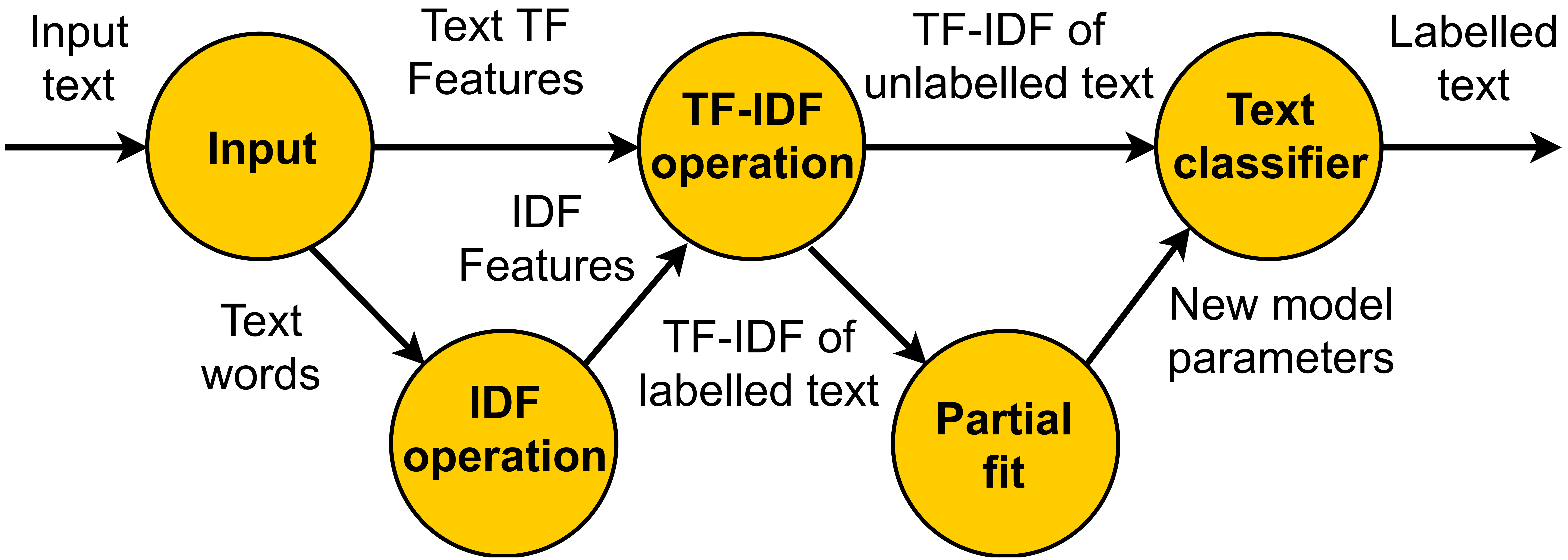
› Zacheilas, N., Kalogeraki, V., Nikolakopoulos, Y., Gulisano, V., Papatriantafilou, M., & Tsigas, P. (2017, June). Maximizing determinism in stream processing under latency constraints. In Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems (pp. 112-123). ACM.

› Kuralenok, I. E., Trofimov, A., Marshalkin, N., & Novikov, B. (2018, September). Deterministic Model for Distributed Speculative Stream Processing. In European Conference on Advances in Databases and Information Systems (pp. 233-246). Springer, Cham.

On-the-fly model updates

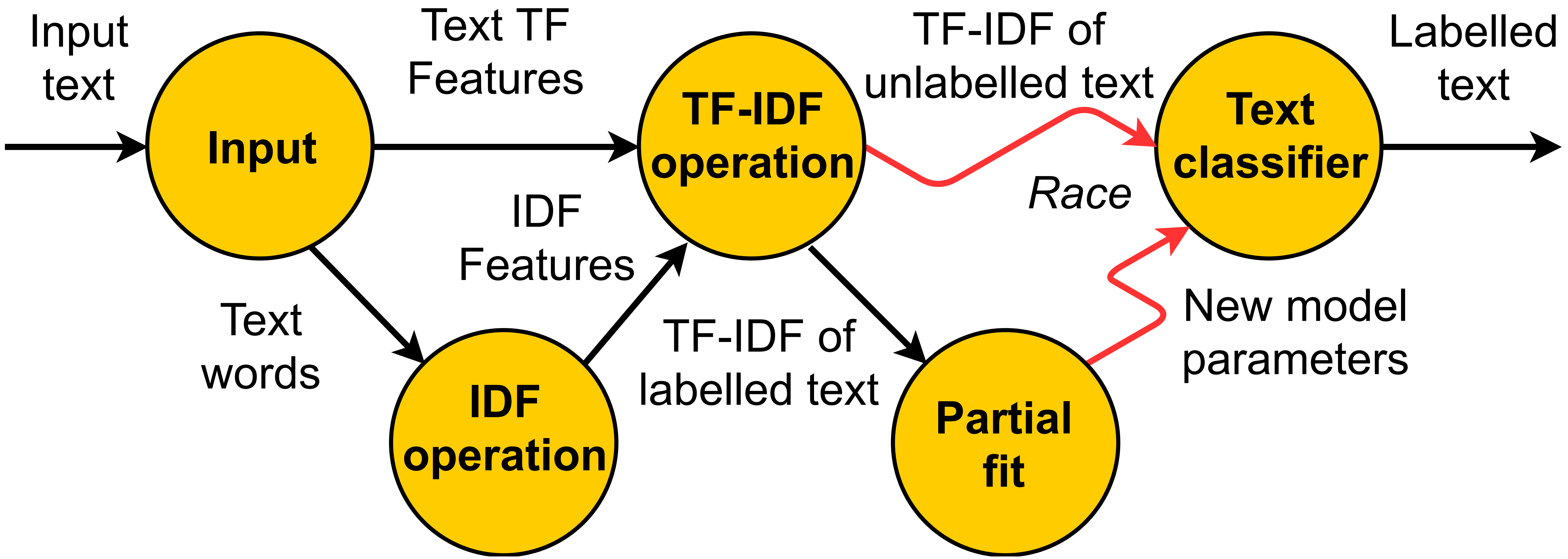
Pitfall #3: on-the-fly model updates

- Data is rapidly changing
- Two types of elements: pre-labeled and raw
- Raw elements should update ML model



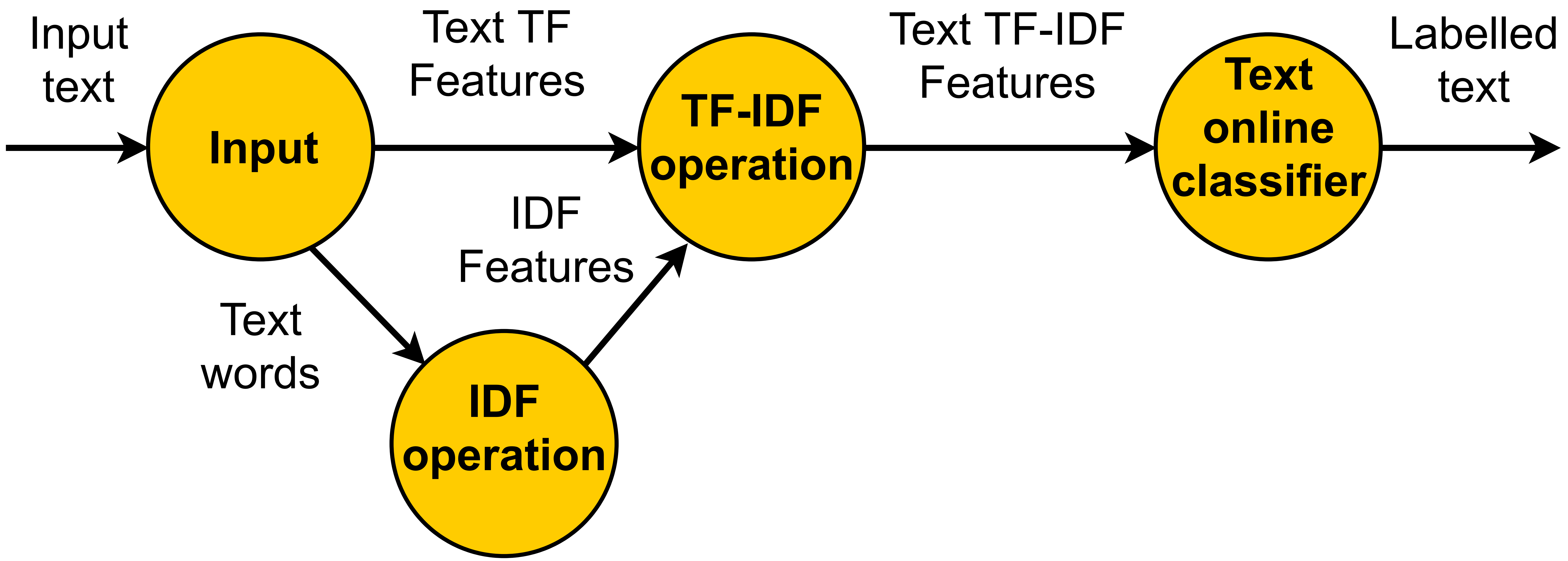
On-the-fly model updates: reproducibility

- Training process may be time-consuming
- Consecutive training and prediction -> latency spikes
- Online learning!



On-the-fly model updates: future work

- Straightforward solution leads to latency spikes or irreproducibile results
- Only online learning algorithms are suitable



Conclusion

Conclusion

- Moving to distributed streaming environment is complex
- Migration of even simple pipeline can be a difficult problem
- There are no automatic tools for migration or for finding issues
- There are several approaches which can potentially fix the problems: deterministic stream processing and online learning algorithms