

意味的類似度に基づいたソースコード とドキュメントの紐付け

オープンソースソフトウェア工学研究室
学籍番号：60230113
氏名：松尾 和輝

ドキュメント（ユーザマニュアル）

ソフトウェアに関する情報をユーザに提供する文書

- － 開発者がソフトウェアの機能説明， 使用法などを記述

Latent Dirichlet allocation (LDA)

LDA is implemented as an Estimator that supports both `EMLDA0optimizer` and `OnlineLDA0optimizer`, and generates a `LDAModel` as the base model. Expert users may cast a `LDAModel` generated by `EMLDA0optimizer` to a `DistributedLDAModel` if needed.

Examples

Python [Scala](#) [Java](#) [R](#)

Refer to the [Python API docs](#) for more details.

```
from pyspark.ml.clustering import LDA

# Loads data.
dataset = spark.read.format("libsvm").load("data/mllib/sample_lda_libsvm_data.txt")

# Trains a LDA model.
lda = LDA(k=10, maxIter=10)
model = lda.fit(dataset)
```

ドキュメントの例¹

ソフトウェアの更新に伴いドキュメントも更新される

1. <https://spark.apache.org/docs/latest/ml-clustering.html>

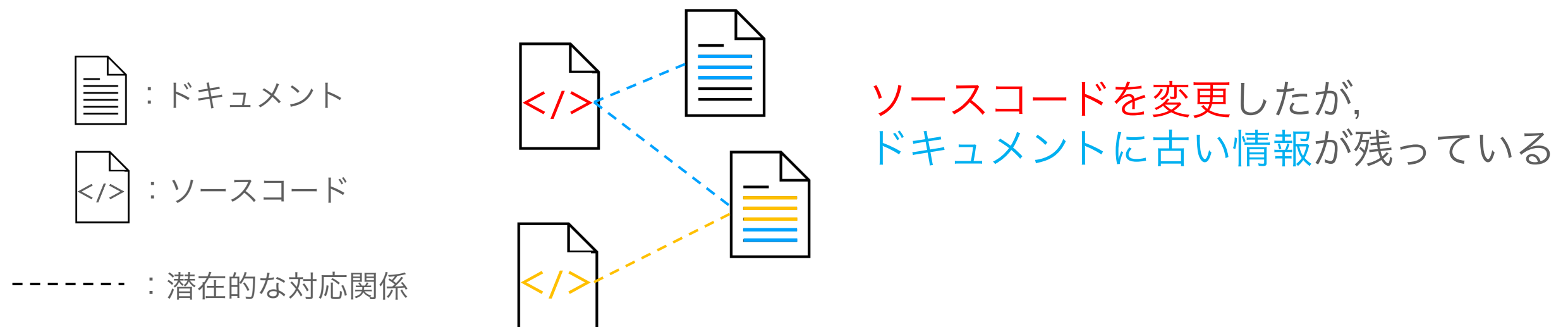
コードとドキュメントの不整合

ドキュメントの更新が遅れるor忘れられることがある

- コードとドキュメントが明示的に対応付けられておらず、開発者が更新すべきドキュメントを見逃してしまうため

問題

- ドキュメント参照者に間違った情報を与えてしまう^[1]
- ソフトウェアの品質低下^[2]



[1] Mira Kajko-Mattsson. 2005. A survey of documentation practice within corrective maintenance. Empirical Software Engineering 10, 1 (2005), 31–55.

[2] David Lorge Parnas. 1994. Software aging. In Proceedings of International Conference on Software Engineering. 279–287.

先行研究[?]

目的 | ソースコードとドキュメントの明示的な紐付け

- 空間的観点 | コードエレメントを用いた単純マッチング
- 時間的観点 | 更新タイミングの時間的近接性

コードエレメント

- 変数名, 関数名などのコードに関係する単語

Environment Variables	
Environment Variable	Meaning
SPARK_DAEMON_MEMORY	Memory to allocate to the history server (default: 1g).
SPARK_DAEMON_JAVA_OPTS	JVM options for the history server (default: none).
SPARK_DAEMON_CLASSPATH	Classpath for the history server (default: none).
SPARK_PUBLIC_DNS	The public address for the history server. If this is not set, links to application history may use the internal address of the server, resulting in broken links (default: none).
SPARK_HISTORY_OPTS	spark.history.* configuration options for the history server (default: none).

Applying compaction on rolling event log files

A long-running application (e.g. streaming) can bring a huge single event log file which may cost a lot to maintain and also requires a bunch of resource to replay per each update in Spark History Server.

Enabling spark.eventLog.rolling.enabled and spark.eventLog.rolling.maxFileSize would let you have rolling event log files instead of single huge event log file which may help some scenarios on its own, but it still doesn't help you reducing the overall size of logs.

正規表現により抽出

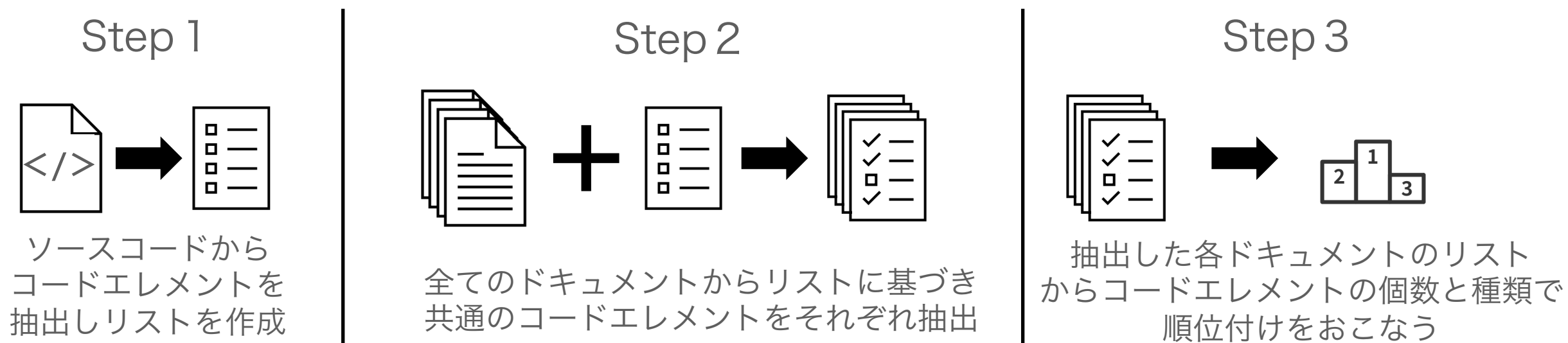
コードエレメント

SPARK_DAEMON_MEMORY
SPARK_DAEMON_JAVA_OPTS
SPARK_DAEMON_CLASSPATH
SPARK_PUBLIC_DNS
SPARK_HISTORY_OPTS
⋮

先行研究 | アプローチ

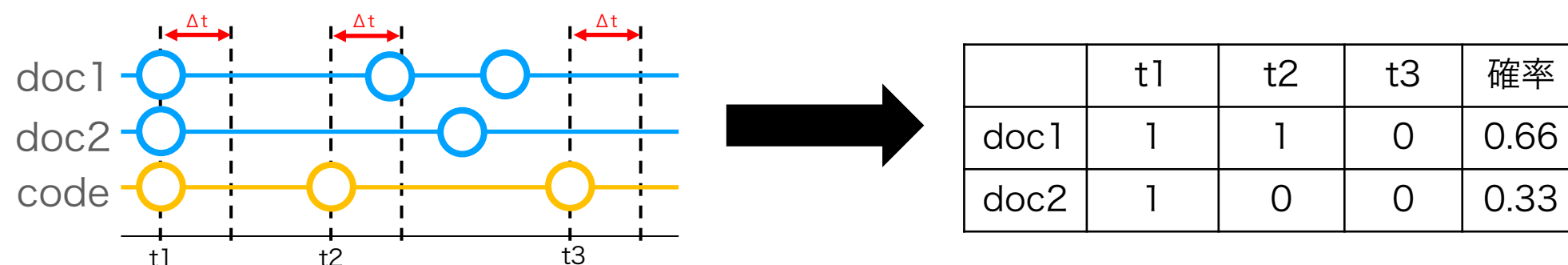
空間的観点

- ソースコードとドキュメントで共通するコード要素の個数 or 種類に基づき順位づけをおこなう



時間的観点

- ソースコードの更新から一定期間内にドキュメントが更新される確率に基づき順位付けをおこなう



先行研究の課題

- 共通するコードエレメントが無いと紐付けできない
 - 先行研究のデータセットの約30%は共通するコードエレメントを持たない

原因

- 同じ意味を持つ, 類似する文章で記述されている
- ドキュメントとソースコード（コメント）を意味的類似度に基づいて紐付けできるのでは

本研究の目的とアプローチ

- 目的
 - ソースコードとドキュメントの紐付け手法の構築
- アプローチ
 - ソースコードをコードブロックとコードコメントに分割
 - ドキュメントを見出しごとに分割
 - それぞれベクトル化し，類似度を計算
 - 類似する部分が一つ以上あれば紐付け

今後の予定

- 背景, 課題, アプローチをしっかりと固める
- 正解集合の作成
 - ソースコードとドキュメントを明示的に紐づける