Language support and linguistics

in Lucene, Solr and ElasticSearch and the eco-system

June 3rd, 2013

Christian Moen
cm@atilika.com

Kultur Brauerei June 3-4 2013

Berlin buzzwords search. store. scale

atilika applied search innovation

About me

- MSc. in computer science, University of Oslo, Norway
- Worked with search at FAST (now Microsoft) for 10 years
  - 5 years in R&D building FAST Enterprise Search Platform in Oslo, Norway
  - 5 years in Services doing solution delivery, technical sales, etc. in Tokyo, Japan
- Founded アティリカ株式会社 in October, 2009
  - We help companies innovate using new technologies and good ideas
  - We do information retrieval, natural language processing and big data
  - We are based in Tokyo, but we have clients everywhere
  - We are a small company, but our customers are typically very big companies
- Newbie Lucene & Solr Committer
  - Mostly been working on Japanese language support (Kuromoji) so far
  - Working on Korean support from a code donation (LUCENE-4956)
- Please write me on cm@atilika.com or cm@apache.org

About this talk

- Basic searching and matching
- Challenges with natural language
- Basic measurements for search quality
- Linguistics in Apache Lucene
- Linguistics in ElasticSearch (quick intro)
- Linguistics in Apache Solr
- Linguistics in the NLP eco-system
- Summary and practical advice

Hands-on demos

Hands-on 1: Working with Apache Lucene analyzers

Hands-on 2: Multi-lingual search using ElasticSearch

Hands-on 3: Multi-lingual search with Apace Solr

Hands-on 4: Other text processing using OpenNLP

What is a search engine?

Documents

1

1	Sushi is very tasty in Japan
2	Visiting the Tsukiji fish market is very fun

Two documents (1 & 2) with English text

Text segmentation

1

1	Sushi is very tasty in Japan
2	Visiting the Tsukiji fish market is very fun

Two documents (1 & 2) with English text

2

1	Sushi	is	very	tasty	in	Japan		
2	Visiting	the	Tsukiji	fish	market	is	very	fun

Documents are turned into searchable terms (tokenization)

Text segmentation

1

1	Sushi is very tasty in Japan
2	Visiting the Tsukiji fish market is very fun

Two documents (1 & 2) with English text

2

1	Sushi	is	very	tasty	in	Japan		
2	Visiting	the	Tsukiji	fish	market	is	very	fun

Documents are turned into searchable terms (tokenization)

3

1	sushi	is	very	tasty	in	japan		
2	visiting	the	tsukiji	fish	market	is	very	fun

Terms/tokens are converted to lowercase form (normalization)

Document indexing

1	sushi	is	very	tasty	in	japan		
2	visiting	the	tsukiji	fish	market	is	very	fun

Tokenized documents with normalized tokens

Document indexing

1	sushi	is	very	tasty	in	japan		
2	visiting	the	tsukiji	fish	market	is	very	fun

Tokenized documents with normalized tokens

sushi	→	1
is	→	1 2
very	→	1 2
tasty	→	1
in	→	1
japan	→	1
visiting	→	2
the	→	2
tsukiji	→	2
fish	→	2
market	→	2
fun	→	2

Inverted index - tokens are mapped to the document ids that contain them

Searching

sushi	→	1	in	→	1	tsukiji	→	2
is	→	1 2	japan	→	1	fish	→	2
very	→	1 2	visiting	→	2	market	→	2
tasty	→	1	the	→	2	fun	→	2

Searching

query

very tasty sushi

sushi	→	1
is	→	1 2
very	→	1 2
tasty	→	1
in	→	1
japan	→	1
visiting	→	2
the	→	2
tsukiji	→	2
fish	→	2
market	→	2
fun	→	2

Searching

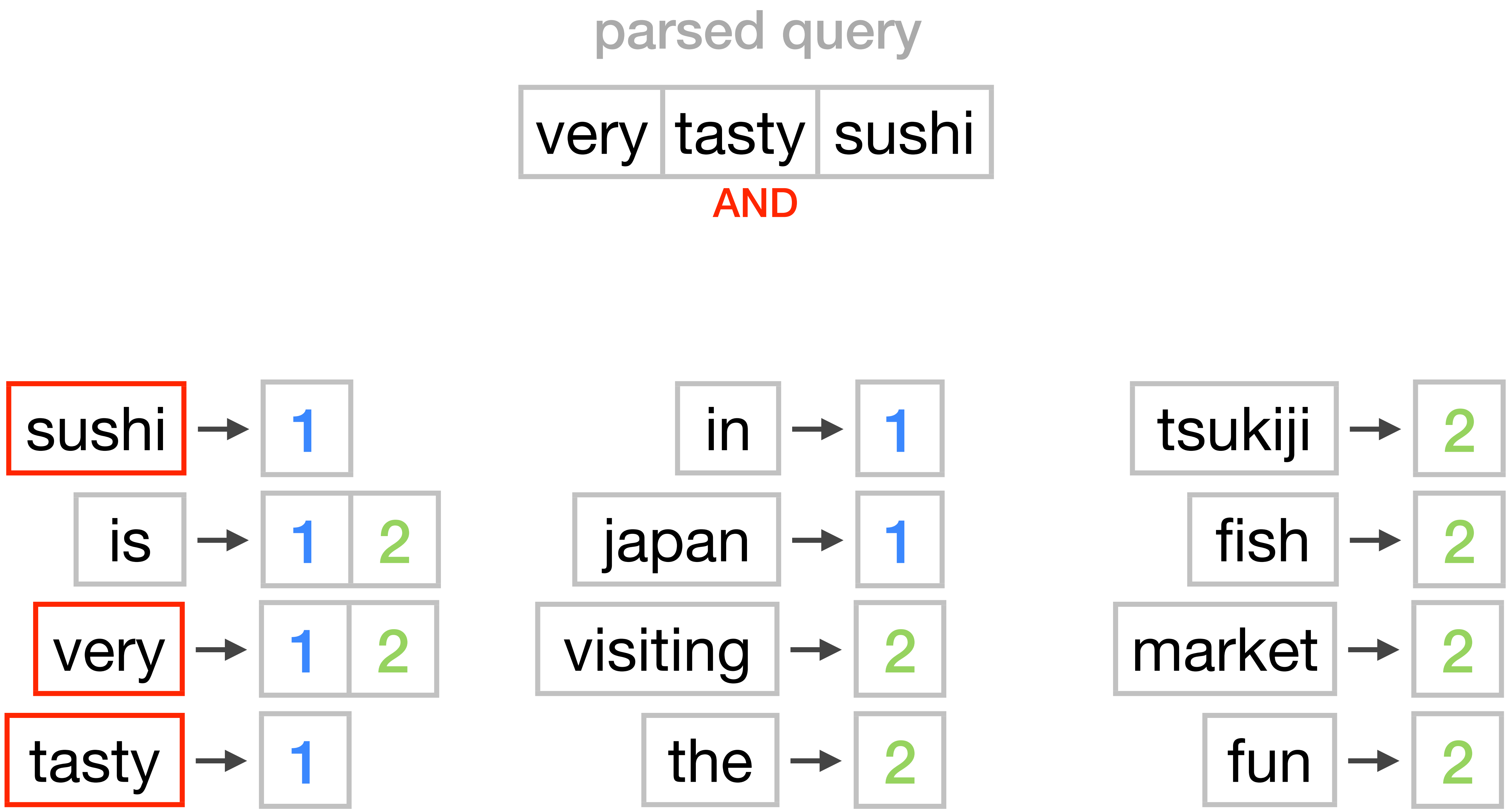
parsed query

very	tasty	sushi

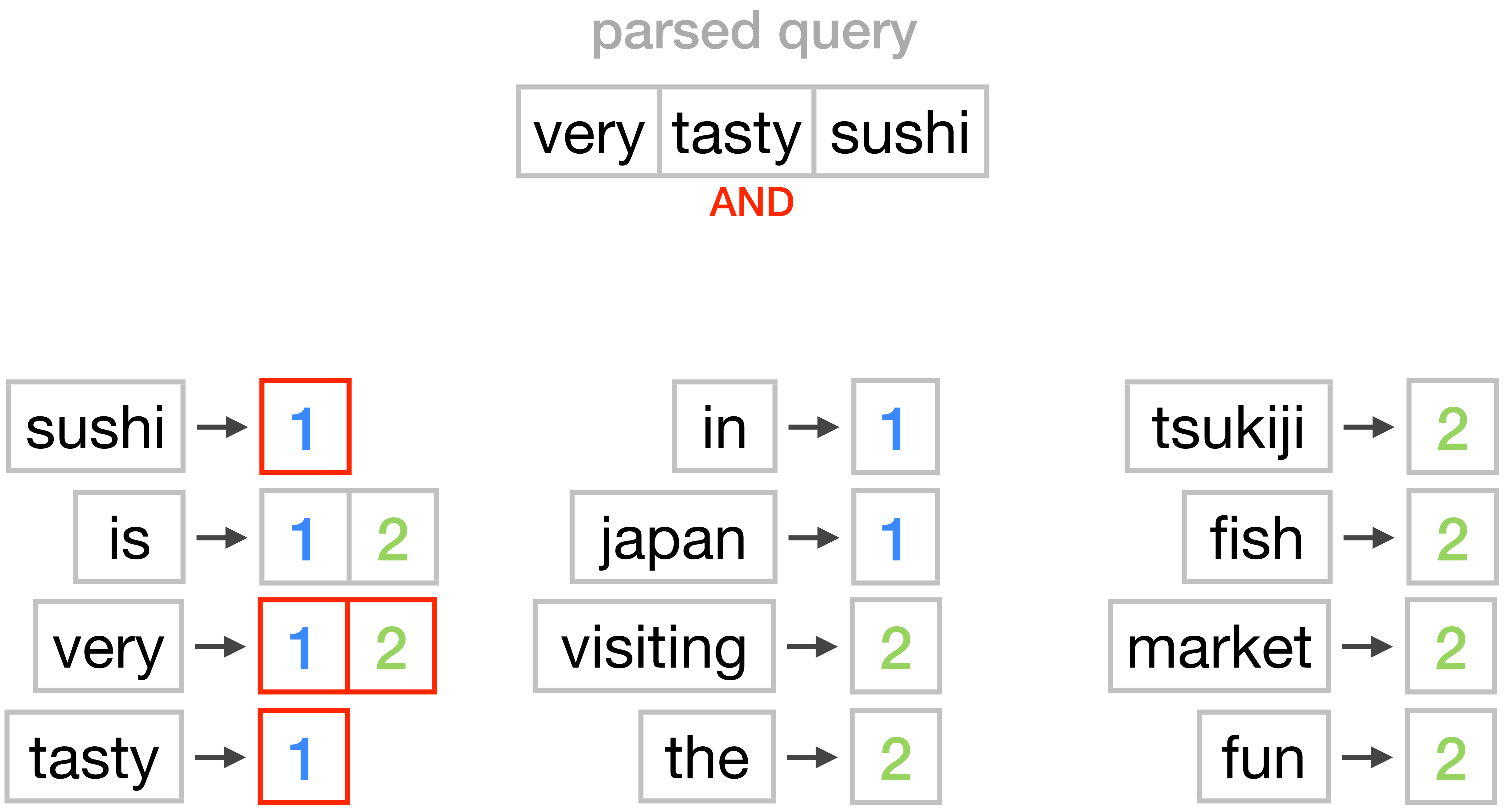
AND

sushi	→	1
is	→	1 2
very	→	1 2
tasty	→	1
in	→	1
japan	→	1
visiting	→	2
the	→	2
tsukiji	→	2
fish	→	2
market	→	2
fun	→	2

Searching

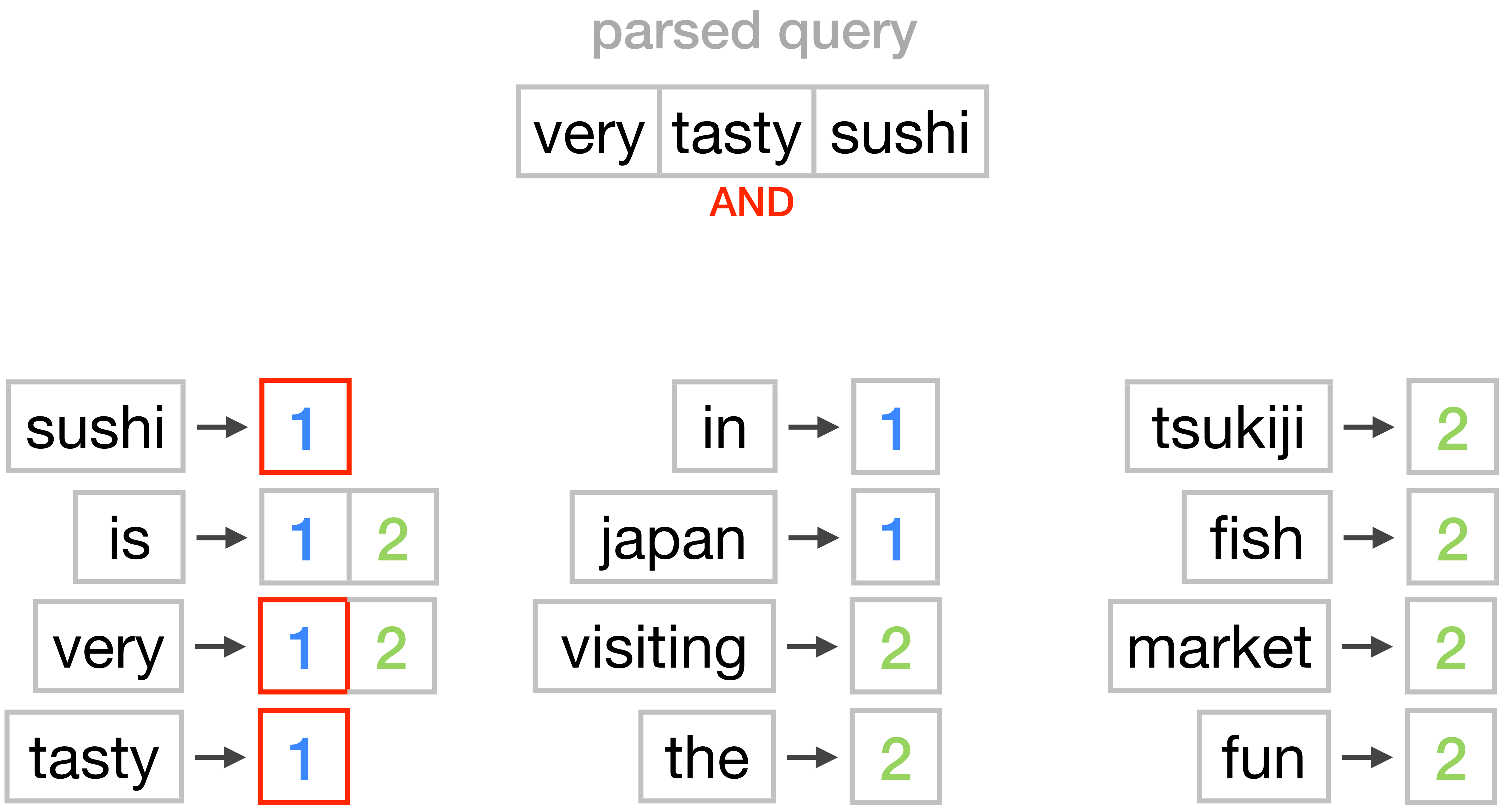

parsed query
very
tasty
sushi
AND
sushi → 1
is → 1 2
very → 1 2
tasty → 1
in → 1
japan → 1
visiting → 2
the → 2
tsukiji → 2
fish → 2
market → 2
fun → 2

Searching

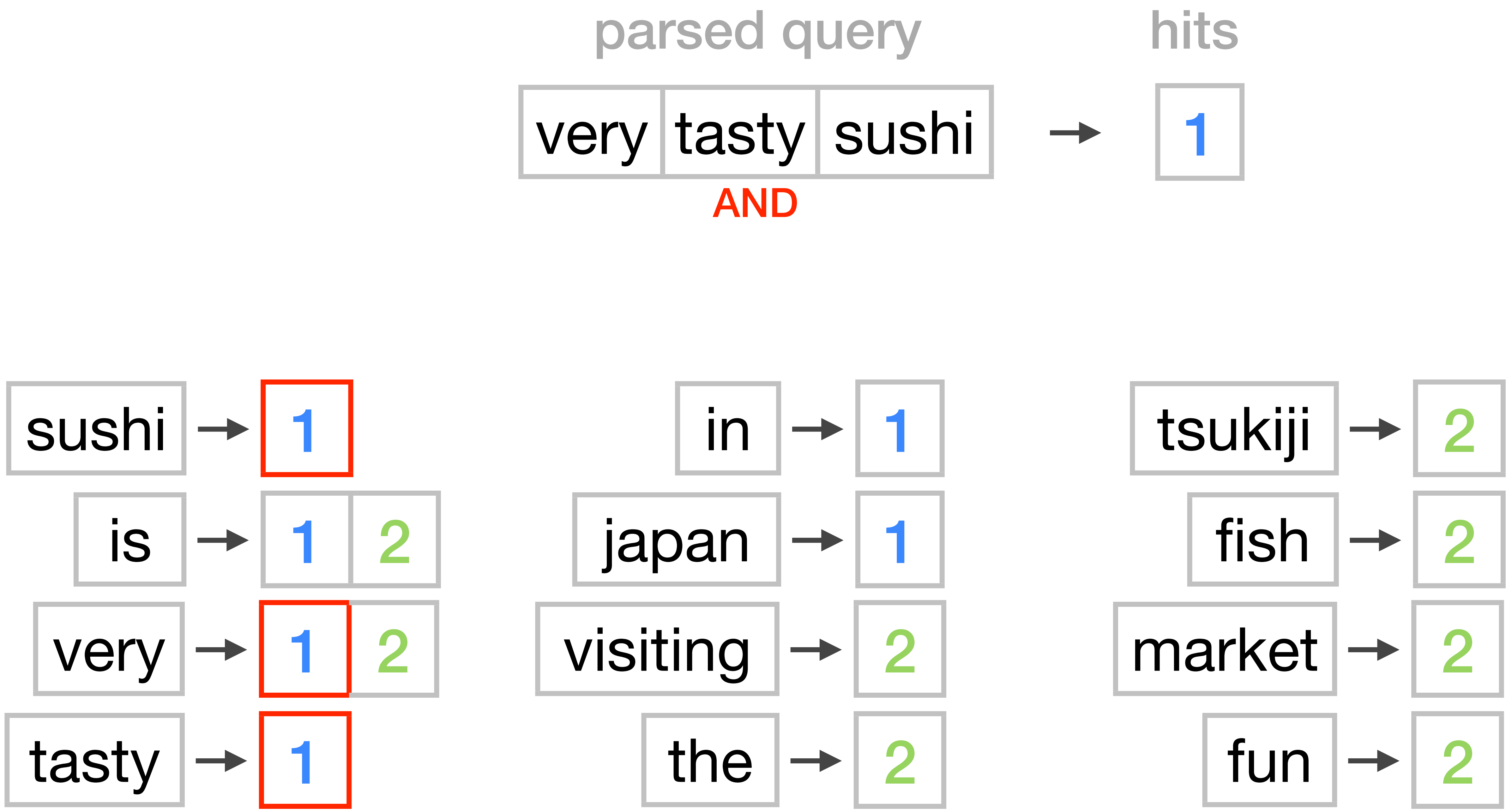


parsed query
very
tasty
sushi
AND
sushi → 1
is → 1 2
very → 1 2
tasty → 1
in → 1
japan → 1
visiting → 2
the → 2
tsukiji → 2
fish → 2
market → 2
fun → 2

Searching


parsed query
very
tasty
sushi
AND
sushi → 1
is → 1 2
very → 1 2
tasty → 1
in → 1
japan → 1
visiting → 2
the → 2
tsukiji → 2
fish → 2
market → 2
fun → 2

Searching


parsed query
hits
very
tasty
sushi
1
AND
sushi
1
is
1
2
very
1
2
tasty
1
in
1
japan
1
visiting
2
the
2
tsukiji
2
fish
2
market
2
fun
2

Searching

sushi	→	1
is	→	1, 2
very	→	1, 2
tasty	→	1
in	→	1
japan	→	1
visiting	→	2
the	→	2
tsukiji	→	2
fish	→	2
market	→	2
fun	→	2

Searching

query

visit fun market

sushi	→	1
is	→	1 2
very	→	1 2
tasty	→	1
in	→	1
japan	→	1
visiting	→	2
the	→	2
tsukiji	→	2
fish	→	2
market	→	2
fun	→	2

Searching

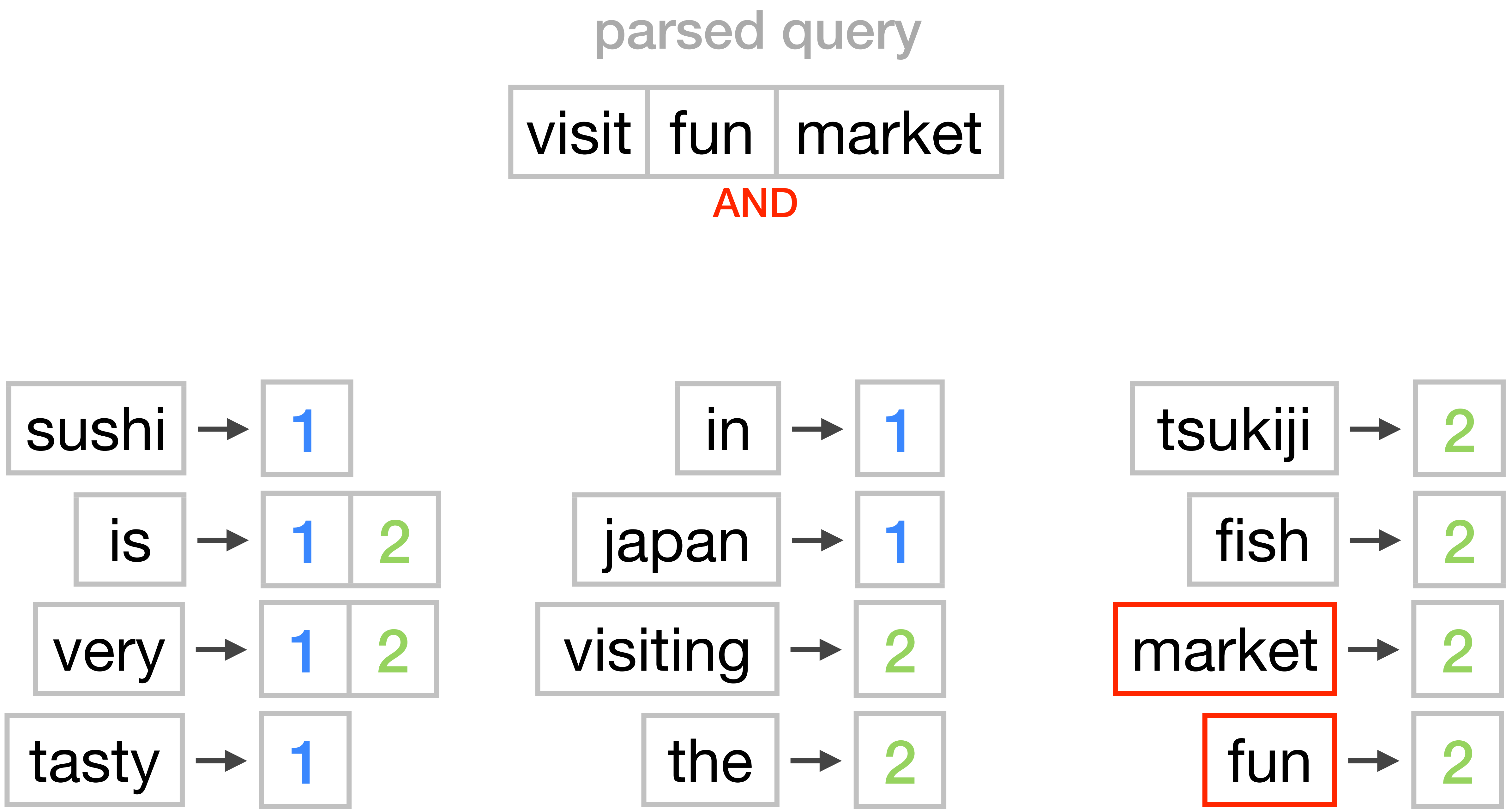
parsed query

visit	fun	market

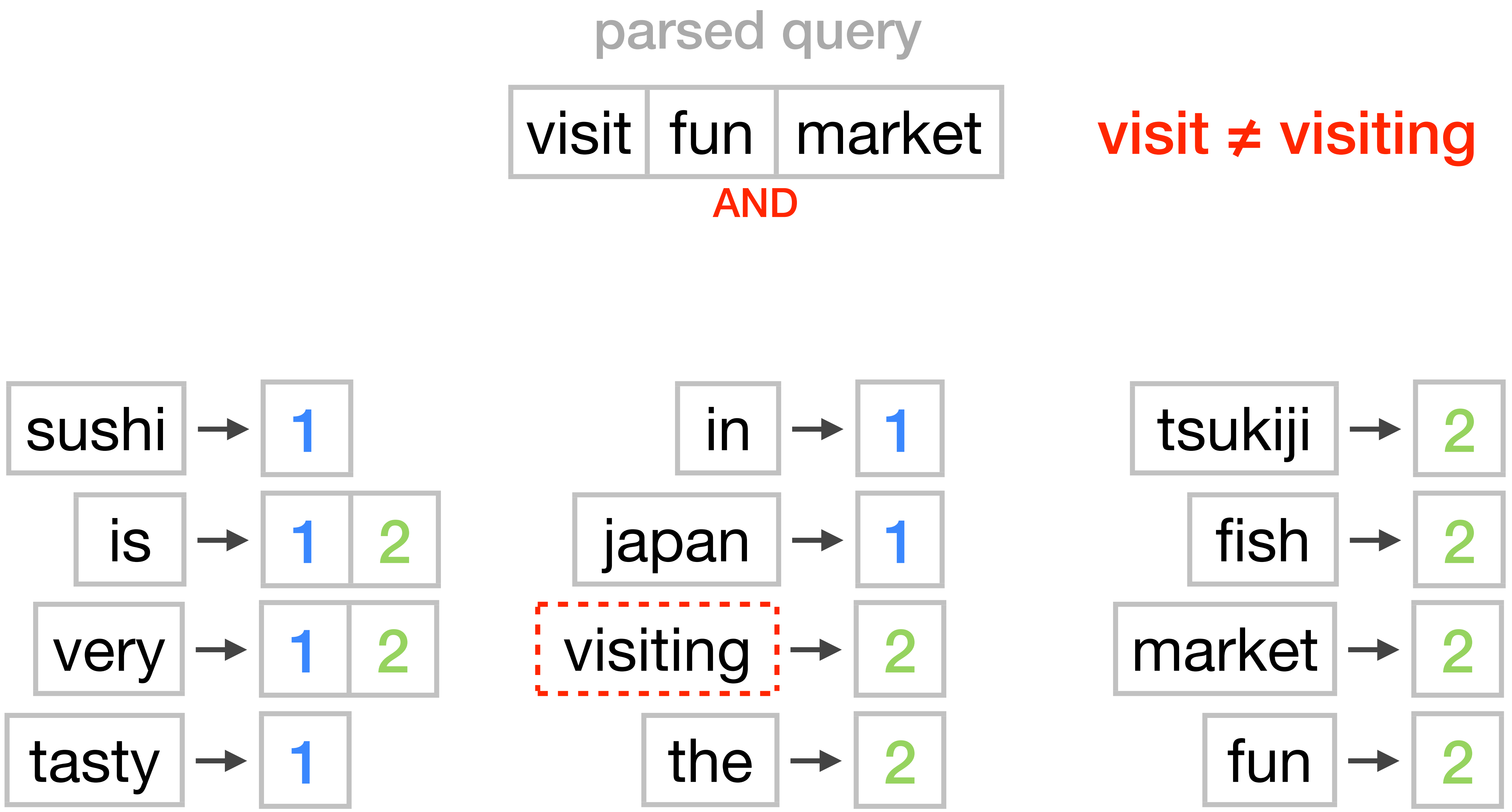
AND

sushi	→	1
is	→	1 2
very	→	1 2
tasty	→	1
in	→	1
japan	→	1
visiting	→	2
the	→	2
tsukiji	→	2
fish	→	2
market	→	2
fun	→	2

Searching

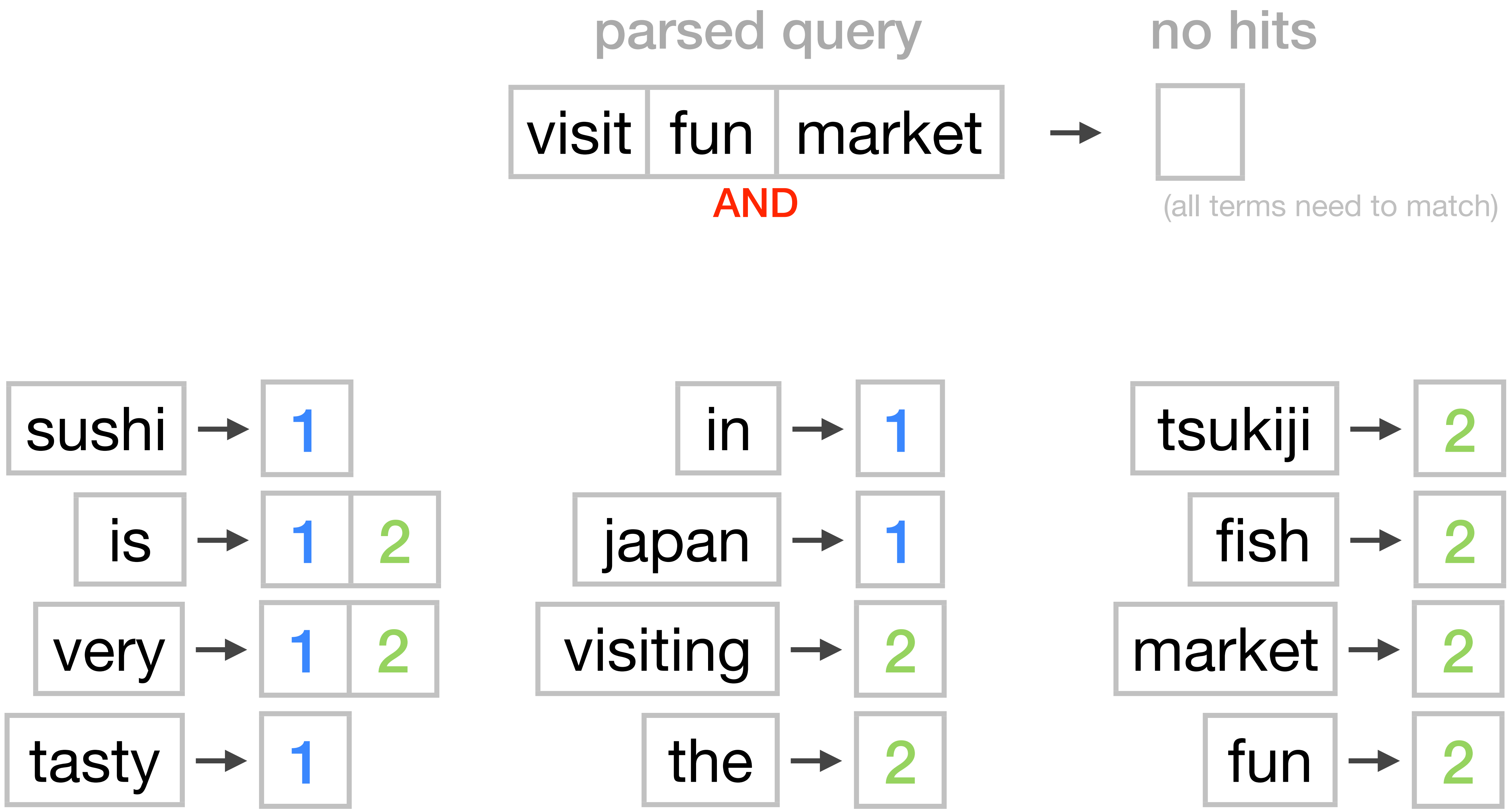

parsed query
visit
fun
market
AND
sushi → 1
is → 1 2
very → 1 2
tasty → 1
in → 1
japan → 1
visiting → 2
the → 2
tsukiji → 2
fish → 2
market → 2
fun → 2

Searching



parsed query
visit
fun
market
AND
visit ≠ visiting
sushi → 1
is → 1 2
very → 1 2
tasty → 1
in → 1
japan → 1
visiting → 2
the → 2
tsukiji → 2
fish → 2
market → 2
fun → 2

Searching


parsed query
visit fun market
AND
no hits
(all terms need to match)
sushi → 1
is → 1 2
very → 1 2
tasty → 1
in → 1
japan → 1
visiting → 2
the → 2
tsukiji → 2
fish → 2
market → 2
fun → 2

What’s the problem?

- Search engines are <u>not</u> magical answering machines

- They match terms in queries against terms in documents, and order matches by rank

Key takeaways

- Text processing affects search quality in big way because it affects matching

Garbage in ⇒ Garbage out

- The “magic” of a search engine is often provided by high quality text processing

Natural language and search

English Deutsch Français العربيةى 日本語

English

Pale ale is a beer made through warm fermentation using pale malt and is one of the world's major beer styles.

English

Pale ale is a beer made through warm fermentation using pale malt and is one of the world's major beer styles.

? How do we want to index world's?

English

Pale ale is a beer made through warm fermentation using pale malt and is one of the world's major beer styles.

- ? How do we want to index world's?
- ? Should a search for style match styles? And should ferment match fermentation?

German

Das Oktoberfest ist das größte Volksfest der Welt und es findet in der bayerischen Landeshauptstadt München.

German

Das Oktoberfest ist das größte Volksfest der Welt und es findet in der bayerischen Landeshauptstadt München.

The Oktoberfest is the world’s largest festival and it takes place in the Bavarian capital Munich.

German

Das Oktoberfest ist das größte Volksfest der Welt und es findet in der bayerischen Landeshauptstadt München.

The Oktoberfest is the world's largest festival and it takes place in the Bavarian capital Munich.

? How do we want to search ü, ö and ß?

German

Das Oktoberfest ist das größte Volksfest der Welt und es findet in der bayerischen Landeshauptstadt München.

The Oktoberfest is the world's largest festival and it takes place in the Bavarian capital Munich.

? How do we want to search ü, ö and ß?

? Do we want a search for **hauptstadt** to match **Landeshauptstadt**?

French

Le champagne est un vin pétillant français protégé par une appellation d'origine contrôlée.

French

Le champagne est un vin pétillant français protégé par une appellation d'origine contrôlée.

Champagne is a French sparkling wine with a protected designation of origin.

French

Le champagne est un vin pétillant français protégé par une appellation d'origine contrôlée.

Champagne is a French sparkling wine with a protected designation of origin.

? How do we want to search é, ç and ô?

French

Le champagne est un vin pétillant français protégé par une appellation d'origine contrôlée.

Champagne is a French sparkling wine with a protected designation of origin.

? How do we want to search é, ç and ô?

? Do we want a search for aoc to match appellation d'origine contrôlée?

Arabic

تُعْتَبَر القَهْوَة العَرَبِيَّه الأصِـــــــيلْـَة رَمْزًا مِن رُمُوْز الكَرَم عِنْد العَرَبْ فِى العَالمَ العَرَبِي.

Arabic

Reads from right to left

تُعْتَبَر القَهْوَة العَرَبِيَّه الأصِــــــيْلَة رَمْزًا مِن رُمُوْز الكَرَم عِنْد العَرَبْ فِى العَالَمَ العَرَبِي.

Arabic

تُعْتَبَر القَهْوَة العَرَبِيَّه الأصِـــــيلْـَة رَمْزًا مِن رُمُوْز الكَرَم عِنْد العَرَبْ فِى العَالمَ اَلعَرَبِـي.

Original Arabian coffee is considered a symbol of generosity among the Arabs in the Arab world.

Arabic

تُعْتَبَر القَهْوَة العَرَبِيَّه الأصِـــــــيلْةَ رَمْزاً مِن رُمُوْز الكَرَم عِنْد العَرَبْ فِى العَالَمَ العَرَبِي.

Original Arabian coffee is considered a symbol of generosity among the Arabs in the Arab world.

? How do we want to search الأصِـــــــيلْةَ?

Arabic

تُعْتَبَر القَهْوَة العَرَبِيَّه الأصِـــــــيلْـَة رَمْزًا مِن رُمُوْز الكَرَم عِنْد العَرَبْ فِى العَالَمَ اَلعَرَبِي.

Original Arabian coffee is considered a symbol of generosity among the Arabs in the Arab world.

? How do we want to search الأصِـــــــيلْـَة?

? Do we want to normalize diacritics?

Arabic

Diacritics normalized (removed)

تعتبر القهوة العربية الأصــــــــيلة رمزا من رموز الكرم عند العرب فى العالم العربي.

Original Arabian coffee is considered a symbol of generosity among the Arabs in the Arab world.

- ? How do we want to search الأصِــــــيلَْة?
- ? Do we want to normalize diacritics?

Arabic

تُعْتَبَر القَهْوَة العَرَبِيَّه الأَصِـــــــيلْةَ رَمْزًا مِن رُمُوْز الكَرَم عِنْد العَرَبْ فِى العَالَمَ العَرَبِي.

Original Arabian coffee is considered a symbol of generosity among the Arabs in the Arab world.

? How do we want to search الأصِـــــــيلْةَ?

? Do we want to normalize diacritics?

? Do we want to correct the common spelling mistake for فِى and ه?

Japanese

ＪＲ新宿駅の近くにビールを飲みに行こうか？

Japanese

ＪＲ新宿駅の近くにビールを飲みに行こうか？

Shall we go for a beer near JR Shinjuku station?

Japanese

ＪＲ新宿駅の近くにビールを飲みに行こうか？

Shall we go for a beer near JR Shinjuku station?

? What are the words in this sentence?
Which tokens do we index?

Japanese

ＪＲ新宿駅の近くにビールを飲みに行こうか？

Shall we go for a beer near JR Shinjuku station?

? What are the words in this sentence? Which tokens do we index?

! Words are *implicit* in Japanese - there is no white space that separates them

Japanese

ＪＲ新宿駅の近くにビールを飲みに行こうか？

Shall we go for a beer near JR Shinjuku station?

? What are the words in this sentence? Which tokens do we index?

! Words are *implicit* in Japanese - there is no white space that separates them

? But how do we find the tokens?

Japanese

ＪＲ	新宿	駅	の	近く	に	ビール	を	飲み	に	行こ	う	か	？

Shall we go for a beer near JR Shinjuku station?

? What are the words in this sentence? Which tokens do we index?

! Words are *implicit* in Japanese - there is no white space that separates them

? But how do we find the tokens?

Japanese

ＪＲ新宿駅の近くにビールを飲みに行こうか？

Shall we go for a beer near JR Shinjuku station?

? Do we want 飲む (to drink) to match 飲み?

Japanese

ＪＲ新宿駅の近くにビールを飲みに行こうか？

Shall we go for a beer near JR Shinjuku station?

? Do we want 飲む (to drink) to match 飲み?

? Do we want ﾋﾞｰﾙ to match ビール?
Does half-width match full-width?

Japanese

ＪＲ新宿駅の近くにビールを飲みに行こうか？

Shall we go for a beer near JR Shinjuku station?

- **?** Do we want 飲む (to drink) to match 飲み?
- **?** Do we want ﾋﾞｰﾙ to match ビール?
  Does half-width match full-width?
- **?** Do we want 🍺 (emoji) to match?

Common traits

- Segmenting source text into tokens
  - Dealing with non-space separated languages
  - Handling punctuation in space separated languages
  - Segmenting compounds into their parts
- Apply relevant linguistic normalizations
  - Character normalization
  - Morphological (or grammatical) normalizations
  - Spelling variations
  - Synonyms and stopwords

Key take-aways

- Natural language is very complex
  - Each language is different with its own set of complexities
  - We have had a high level look at languages

English German French Arabic Japanese

  - But there is also...

Spanish Greek Hebrew Russian Thai Korean Chinese ... and many more

- Search needs per-language processing
  - Many considerations to be made (often application-specific)

Basic search quality measurements

Precision

Fraction of retrieved documents that are relevant

$$\text{precision} = \frac{|\{\text{ relevant docs }\} \cap \{\text{ retrieved docs }\}|}{|\{\text{ retrieved docs }\}|}$$

Recall

Fraction of relevant documents that are retrieved

$$\text{recall} = \frac{|\,\{\text{ relevant docs }\} \cap \{\text{ retrieved docs }\}\,|}{|\,\{\text{ relevant docs }\}\,|}$$

Precision vs. Recall

? Should I optimize for precision or recall?

Precision vs. Recall

? Should I optimize for precision or recall?

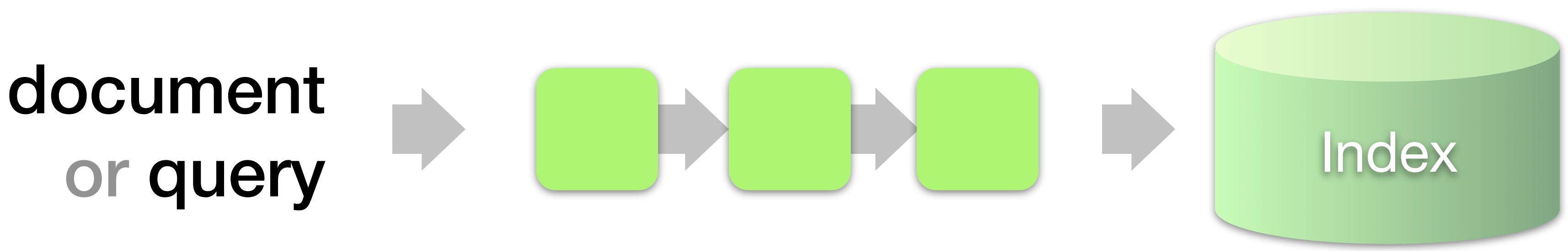
! That depends on your application

Precision vs. Recall

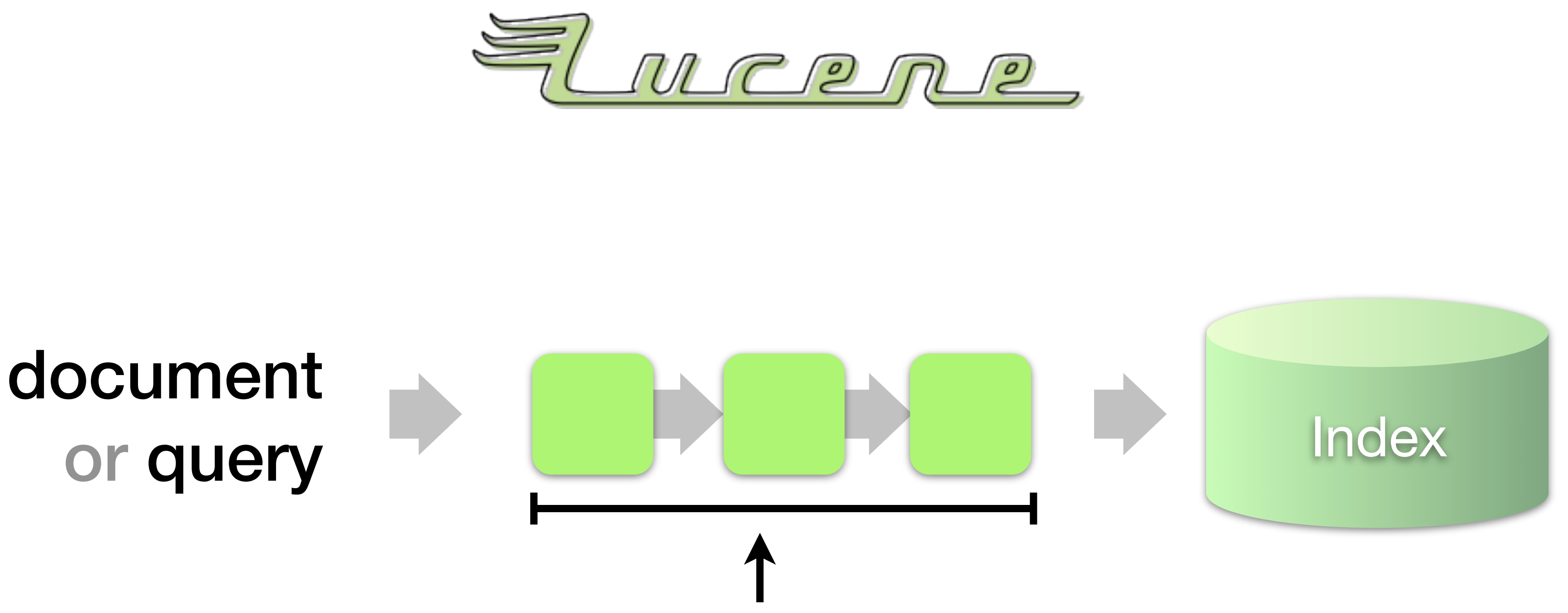
- ? Should I optimize for precision or recall?
- ! That depends on your application
- ! A lot of tuning work is in practice often about improving recall without hurting precision

Linguistics in Lucene

Simplified architecture



Simplified architecture



Lucene analysis chain / Analyzer

1. Analyzes queries or documents in a pipelined fashion before indexing or searching
2. Analysis itself is done by an **analyzer** on a per field basis
3. Key plug-in point for linguistics in Lucene

Analyzers

- ? What does an Analyzer do?

Analyzers

? What does an Analyzer do?

! Analyzers take text as its input and turns it into a stream of tokens

Analyzers

? What does an Analyzer do?

! Analyzers take text as its input and turns it into a stream of tokens

! Tokens are produced by a Tokenizer

Analyzers

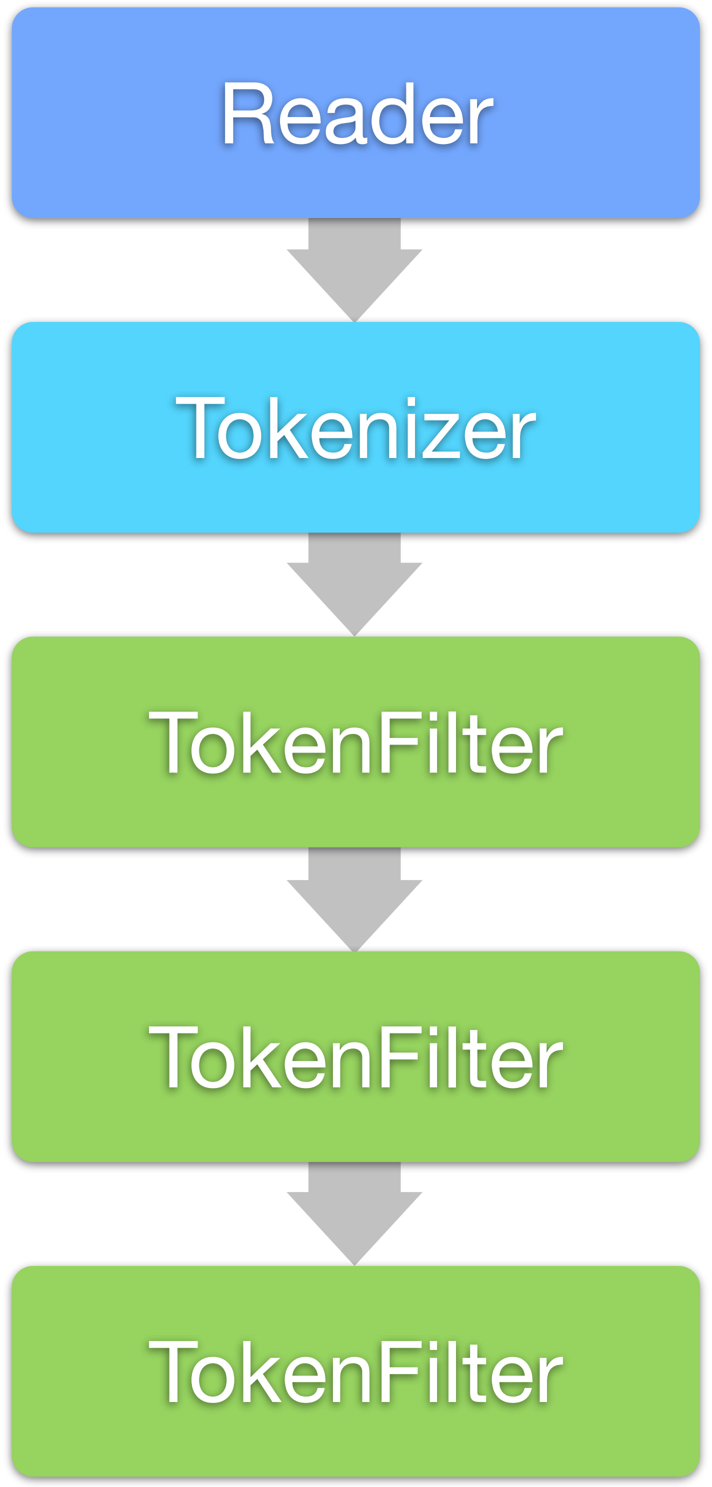
? What does an Analyzer do?

! Analyzers take text as its input and turns it into a stream of tokens

! Tokens are produced by a Tokenizer

! Tokens can be processed further by a chain of TokenFilters downstream

Analyzer high-level concepts



Reader

- Stream to be analyzed is provided by a Reader (from java.io)
- Can have chain of associated CharFilters (not discussed)

Tokenizer

- Segments text provider by reader into tokens
- Most interesting things happen in incrementToken() method

TokenFilter

- Updates, mutates or enriches tokens
- Most interesting things happen in incrementToken() method

TokenFilter

…

TokenFilter

…

Lucene processing example

Le champagne est protégé par une appellation d'origine contrôlée.

FrenchAnalyzer

Le champagne est protégé par une appellation d'origine contrôlée.

FrenchAnalyzer

Le champagne est protégé par une appellation d'origine contrôlée.

StandardTokenizer

FrenchAnalyzer

Le champagne est protégé par une appellation d'origine contrôlée.

StandardTokenizer

Le	champagne	est	protégé	par	une	appellation	d'origine	contrôlée

FrenchAnalyzer

Le champagne est protégé par une appellation d'origine contrôlée.

$\downarrow$ StandardTokenizer $\downarrow$

Le	champagne	est	protégé	par	une	appellation	d'origine	contrôlée

$\downarrow$ ElisionFilter $\downarrow$

FrenchAnalyzer

Le champagne est protégé par une appellation d'origine contrôlée.

StandardTokenizer

Le	champagne	est	protégé	par	une	appellation	d'origine	contrôlée

ElisionFilter

Le	champagne	est	protégé	par	une	appellation	origine	contrôlée

FrenchAnalyzer

Le champagne est protégé par une appellation d'origine contrôlée.

StandardTokenizer

↓

Le	champagne	est	protégé	par	une	appellation	d'origine	contrôlée

ElisionFilter

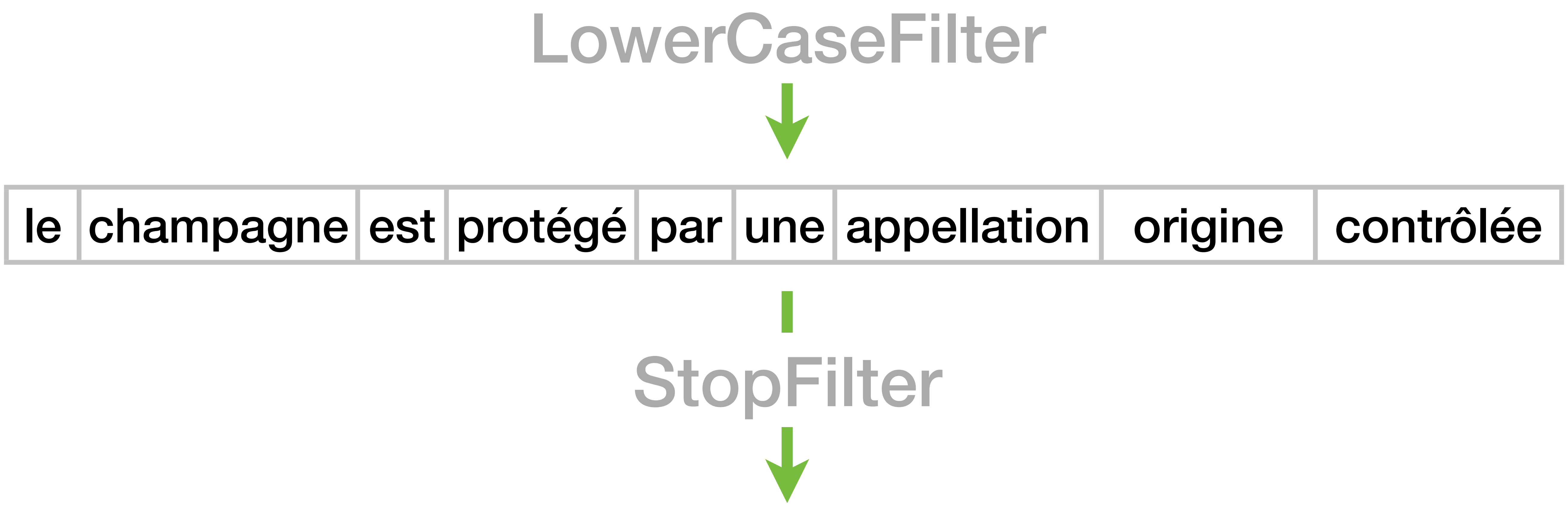
↓

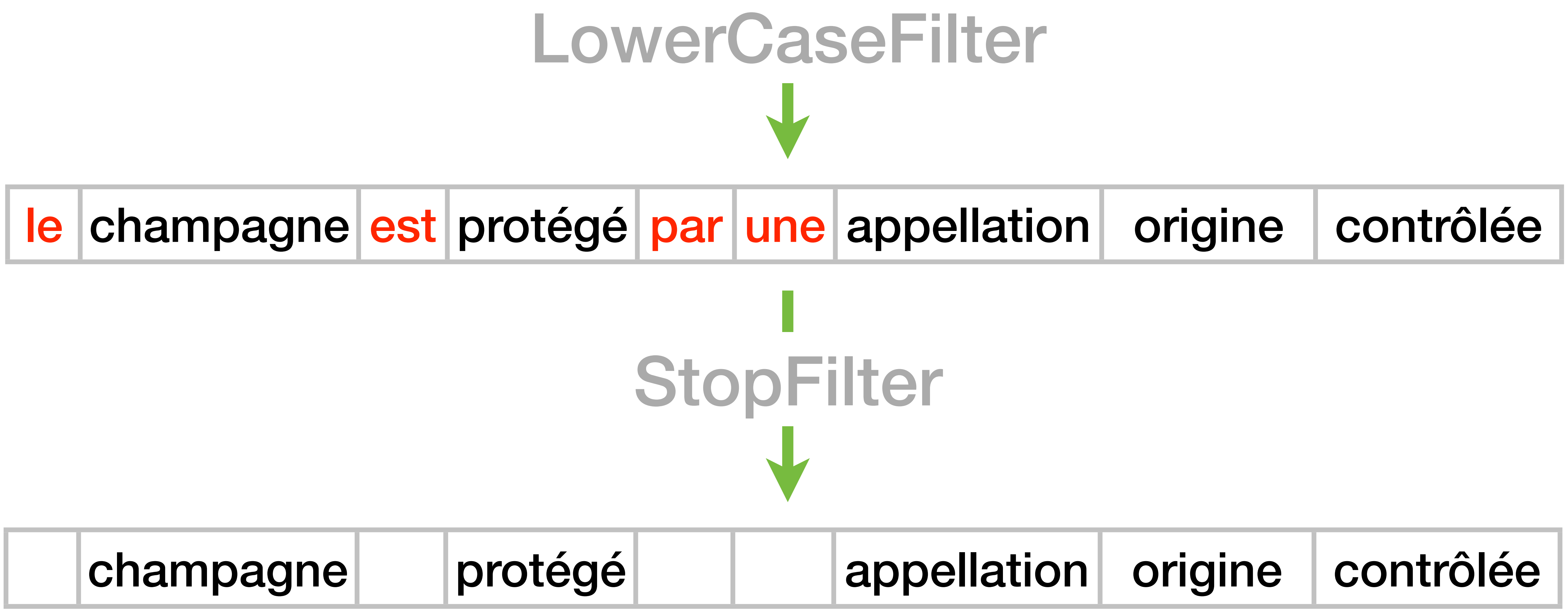
Le	champagne	est	protégé	par	une	appellation	origine	contrôlée

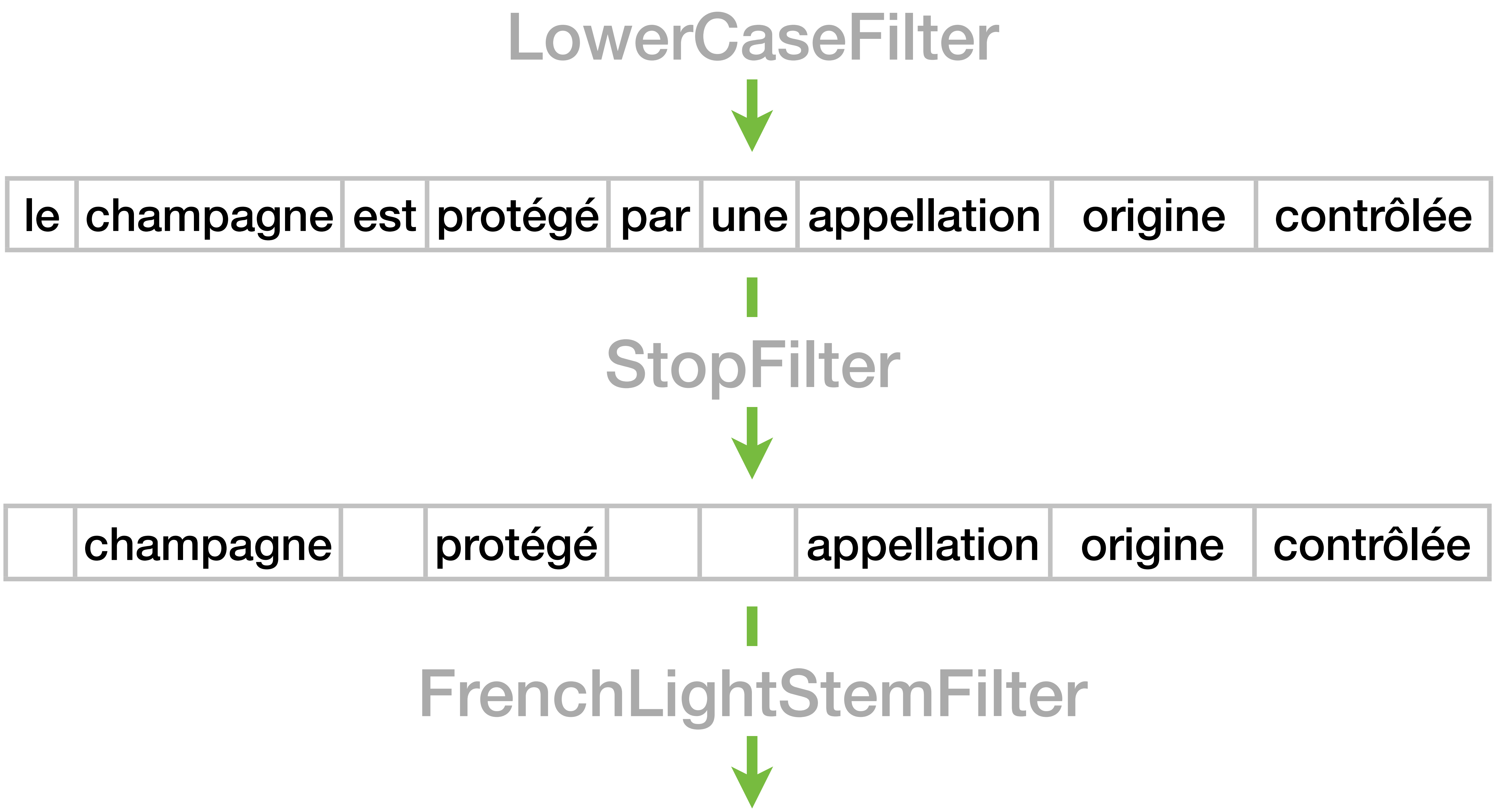
LowerCaseFilter

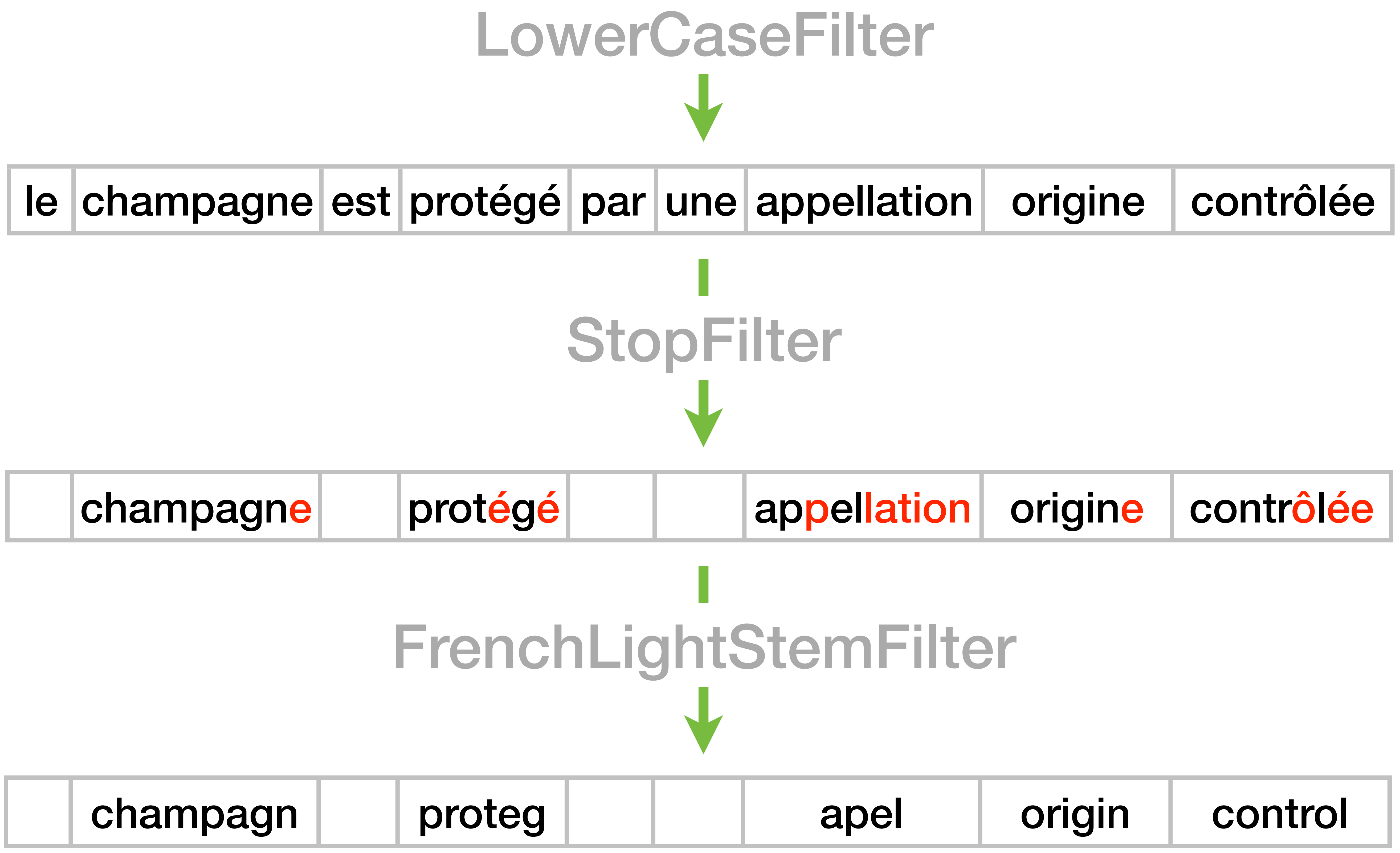
LowerCaseFilter

le	champagne	est	protégé	par	une	appellation	origine	contrôlée


LowerCaseFilter
le
champagne
est
protégé
par
une
appellation
origine
contrôlée
StopFilter

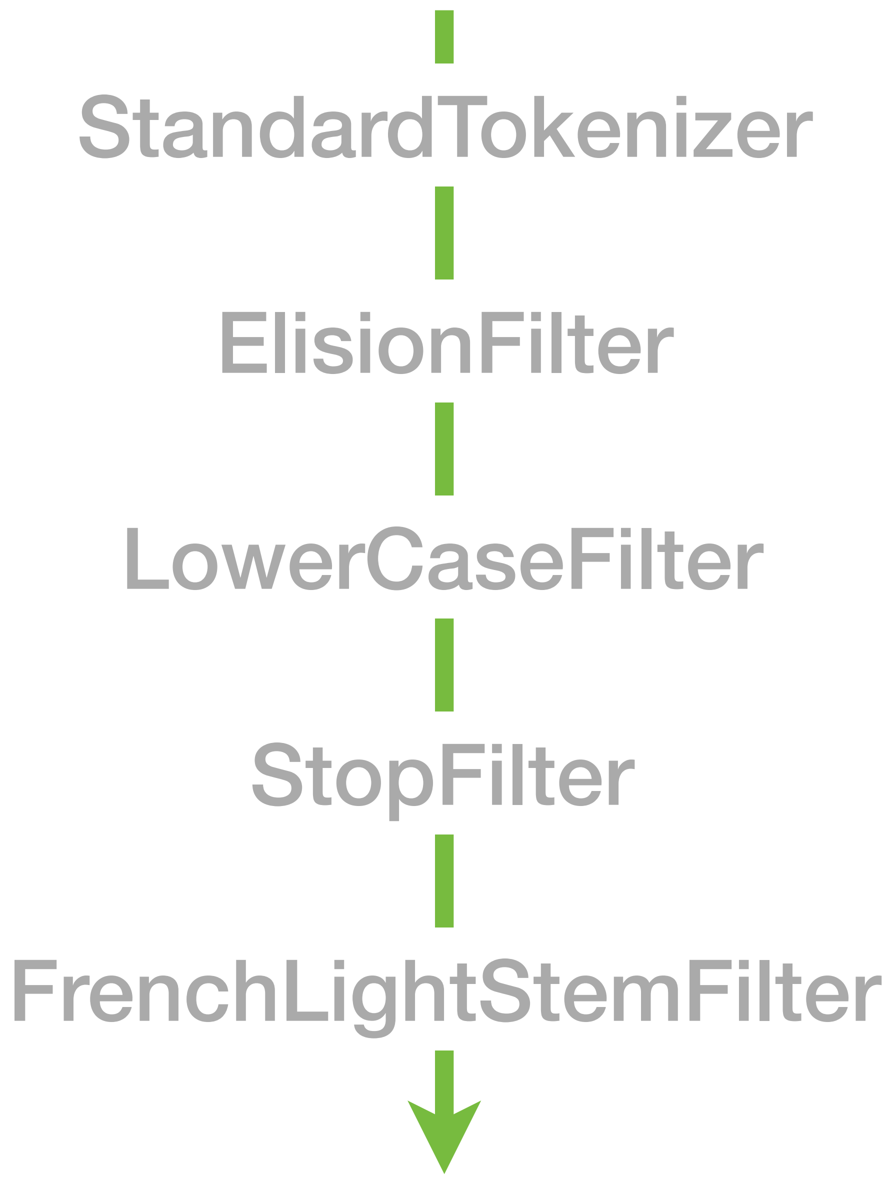

LowerCaseFilter
le
champagne
est
protégé
par
une
appellation
origine
contrôlée
StopFilter
champagne
protégé
appellation
origine
contrôlée


LowerCaseFilter
le
champagne
est
protégé
par
une
appellation
origine
contrôlée
StopFilter
champagne
protégé
appellation
origine
contrôlée
FrenchLightStemFilter


LowerCaseFilter
le | champagne | est | protégé | par | une | appellation | origine | contrôlée
StopFilter
champagne | protégé | appellation | origine | contrôlée
FrenchLightStemFilter
champagn | proteg | apel | origin | control

FrenchAnalyzer

Le champagne est protégé par une appellation d'origine contrôlée.



	champagn		proteg			apel	origin	control

Analyzer processing model

- Analyzers provide a TokenStream
  - Retrieve it by calling tokenStream(field, reader)
  - tokenStream() bundles together tokenizers and any additional filters necessary for analysis
- Input is advanced by incrementToken()
  - Information about the token itself is provided by so-called TokenAttributes attached to the stream
  - Attribute for term text, offset, token type, etc.
  - TokenAttributes are updated on incrementToken()

Hands-on: Working with analyzers in code

Synonyms

Synonyms

- Synonyms are flexible and easy-to-use
  - Very powerful tools for improving recall
- Two types of synonyms
  - One way/mapping **“sparkling wine => champagne”**
  - Two way/equivalence **“aoc, appellation d'origine contrôlée”**
- Can be applied index-time or query-time
  - Apply synonyms on one side - not both
- Best practice is to apply synonyms query-side
  - Allows for updating synonyms without reindexing
  - Allows for turning synonyms on and off easily

Hands-on: French analysis with synonyms

Linguistics in ElasticSearch (quick intro)

ElasticSearch linguistics highlights

- Uses Lucene analyzers, tokenizers & filters
- Analyzers are made available through a provider interface
- Some analyzers available through plugins, i.e. kuromoji, smartcn, icu, etc.
- Analyzers can be set up in your mapping
- Analyzers can also be chosen based on a field in your document, i.e. a lang field

elasticsearch

Hands-on: Simple multi-language example

Linguistics in Solr

Linguistics in Solr

- Uses Lucene analyzers, tokenizers & filters
- Linguistic processing is defined by field types in schema.xml
- Different processing can be applied on indexing and querying side if desired
- A rich set of pre-defined and ready-to-use per-language field types are available
- Defaults can be used as starting points for further configuration or as they are

French in schema.xml

```
<!-- French -->
<field name="title" type="text_fr" indexed="true" stored="true"/>
<field name="body" type="text_fr" indexed="true" stored="true"/>

<dynamicField name="*_fr" type="text_fr" indexed="true" stored="true"/>


<!-- French -->
<fieldType name="text_fr" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <!-- removes l', etc -->
    <filter class="solr.ElisionFilterFactory" ignoreCase="true"
            articles="lang/contractions_fr.txt"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true"
            words="lang/stopwords_fr.txt" format="snowball"
            enablePositionIncrements="true"/>
    <filter class="solr.FrenchLightStemFilterFactory"/>
    <!-- less aggressive: <filter class="solr.FrenchMinimalStemFilterFactory"/> -->
    <!-- more aggressive: <filter class="solr.SnowballPorterFilterFactory"
                                 language="French"/> -->
  </analyzer>
</fieldType>
```

Arabic in schema.xml

```
<!-- Arabic -->
<field name="title" type="text_ar" indexed="true" stored="true"/>
<field name="body" type="text_ar" indexed="true" stored="true"/>

<dynamicField name="*_ar" type="text_ar" indexed="true" stored="true"/>


<!-- Arabic -->
<fieldType name="text_ar" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <!-- for any non-arabic -->
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true"
            words="lang/stopwords_ar.txt" enablePositionIncrements="true"/>
    <!-- normalizes alef maksura to yeh, etc -->
    <filter class="solr.ArabicNormalizationFilterFactory"/>
    <filter class="solr.ArabicStemFilterFactory"/>
  </analyzer>
</fieldType>
```

Field types in schema.xml

- text_ar **Arabic**
- text_bg **Bulgarian**
- text_ca **Catalan**
- text_cjk **CJK**
- text_cz **Czech**
- text_da **Danish**
- text_de **German**
- text_el **Greek**
- text_es **Spanish**
- text_eu **Basque**
- text_fa **Farsi**
- text_fi **Finnish**
- text_fr **French**
- text_ga **Irish**
- text_gl **Galician**
- text_hi **Hindi**
- text_hu **Hungarian**
- text_hy **Armenian**
- text_id **Indonedian**
- text_it **Italian**
- text_lv **Latvian**
- text_nl **Dutch**
- text_no **Norwegian**
- text_pt **Portuguese**
- text_ro **Romanian**
- text_ru **Russian**
- text_sv **Swedish**
- text_th **Thai**
- text_fr **Turkish**

Field types in schema.xml

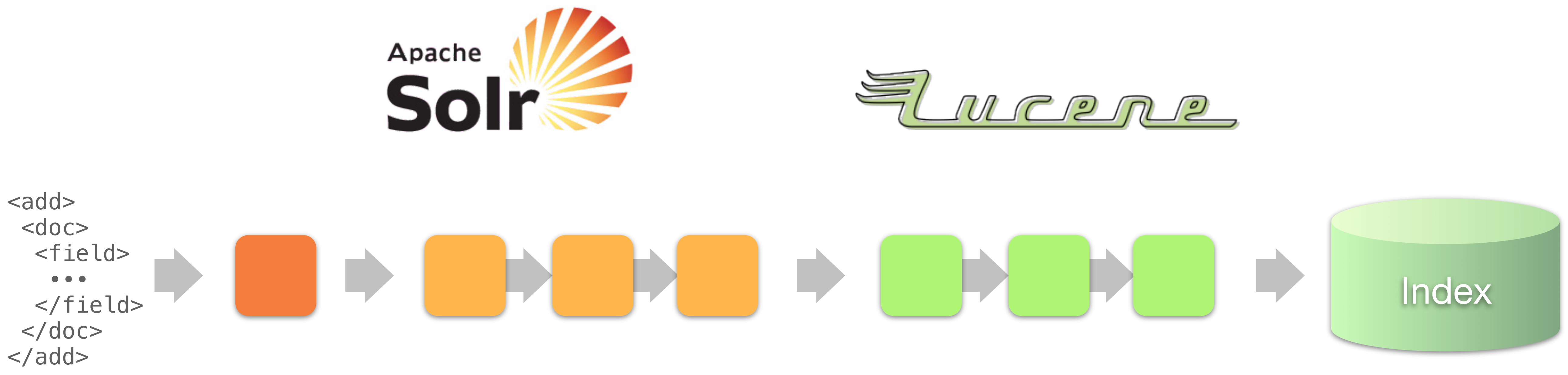
- text_ar **Arabic**
- text_bg **Bulgarian**
- text_ca **Catalan**
- text_cjk **CJK**
- text_cz **Czech**
- text_da **Danish**
- text_de **German**
- text_el **Greek**
- text_es **Spanish**
- text_eu **Basque**
- text_fa **Farsi**
- text_fi **Finnish**
- text_fr **French**
- text_ga **Irish**
- text_gl **Galician**
- text_hi **Hindi**
- text_hu **Hungarian**
- text_hy **Armenian**
- text_id **Indonedian**
- text_it **Italian**
- text_lv **Latvian**
- text_nl **Dutch**
- text_no **Norwegian**
- text_pt **Portuguese**
- text_ro **Romanian**
- text_ru **Russian**
- text_sv **Swedish**
- text_th **Thai**
- text_fr **Turkish**
- text_ko **Korean**

Coming soon!

LUCENE-4956

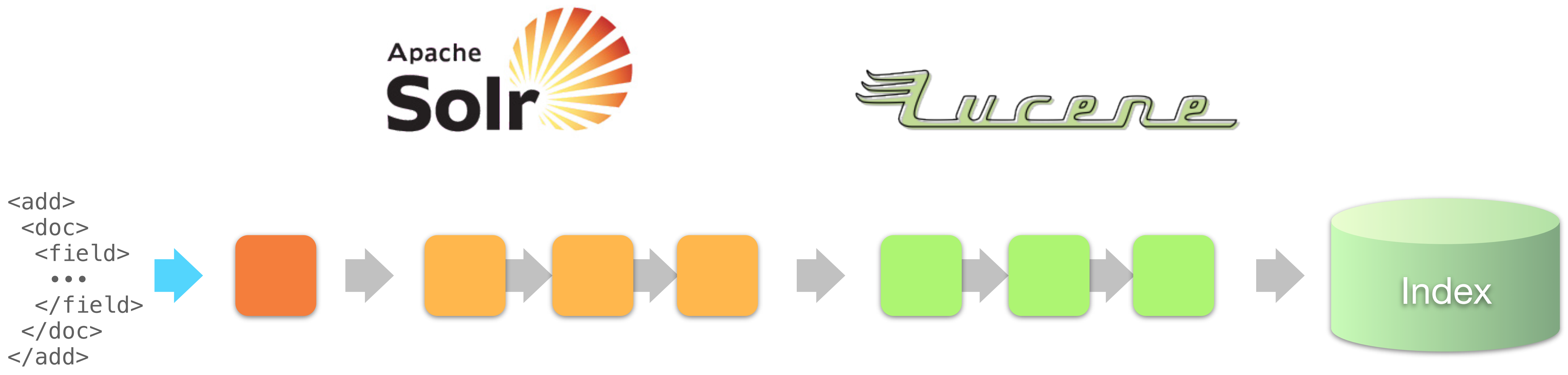
Solr processing

Adding document details



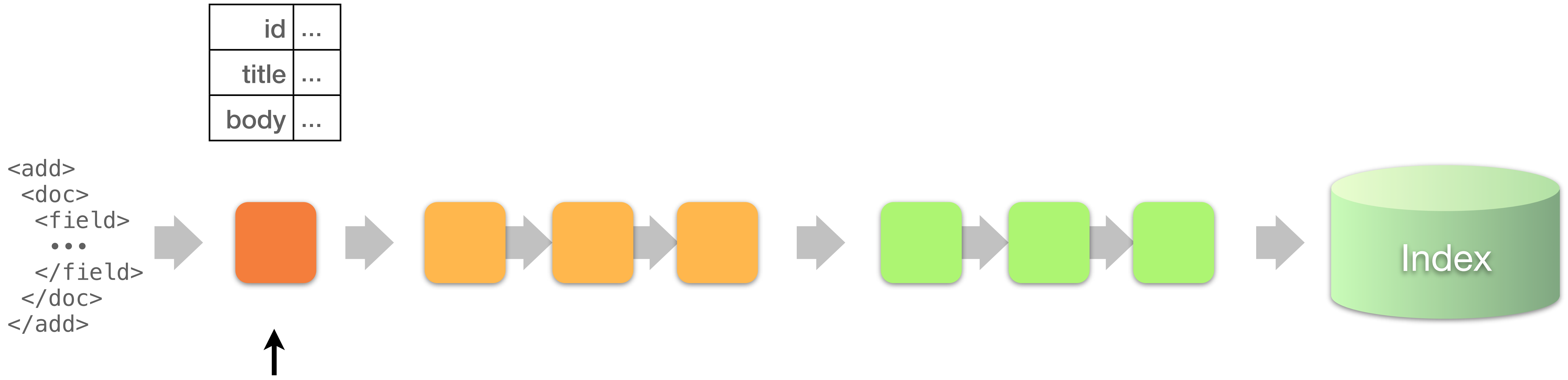
```
<add>
 <doc>
  <field>
   •••
  </field>
 </doc>
</add>
```

Adding document details



```
<add>
 <doc>
  <field>
   •••
  </field>
 </doc>
</add>
```

Adding document details



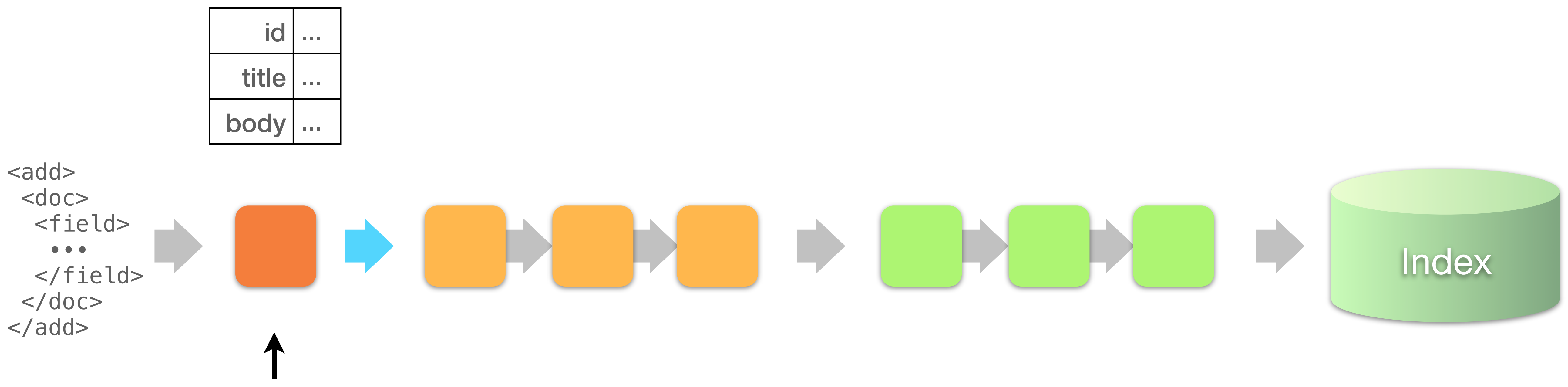
id	...
title	...
body	...

```
<add>
 <doc>
  <field>
   •••
  </field>
 </doc>
</add>
```

UpdateRequestHandler handles request

1. Receives a document via HTTP in XML (or JSON, CSV, ...)
2. Converts document to a SolrInputDocument
3. Activates the update chain

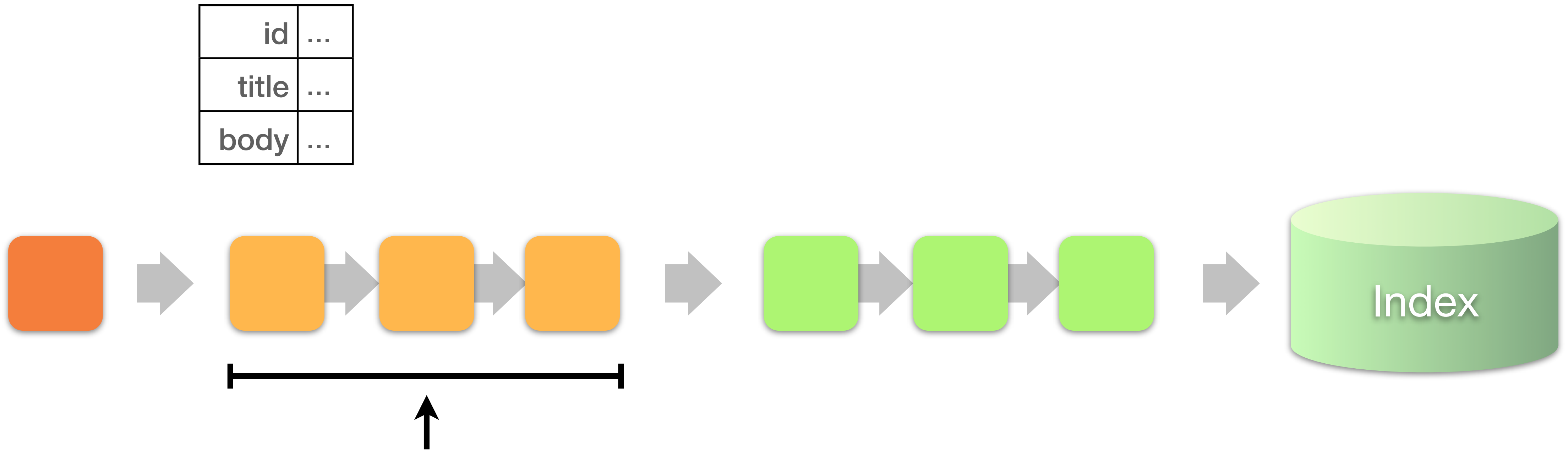
Adding document details



UpdateRequestHandler handles request

1. Receives a document via HTTP in XML (or JSON, CSV, ...)
2. Converts document to a SolrInputDocument
3. Activates the update chain

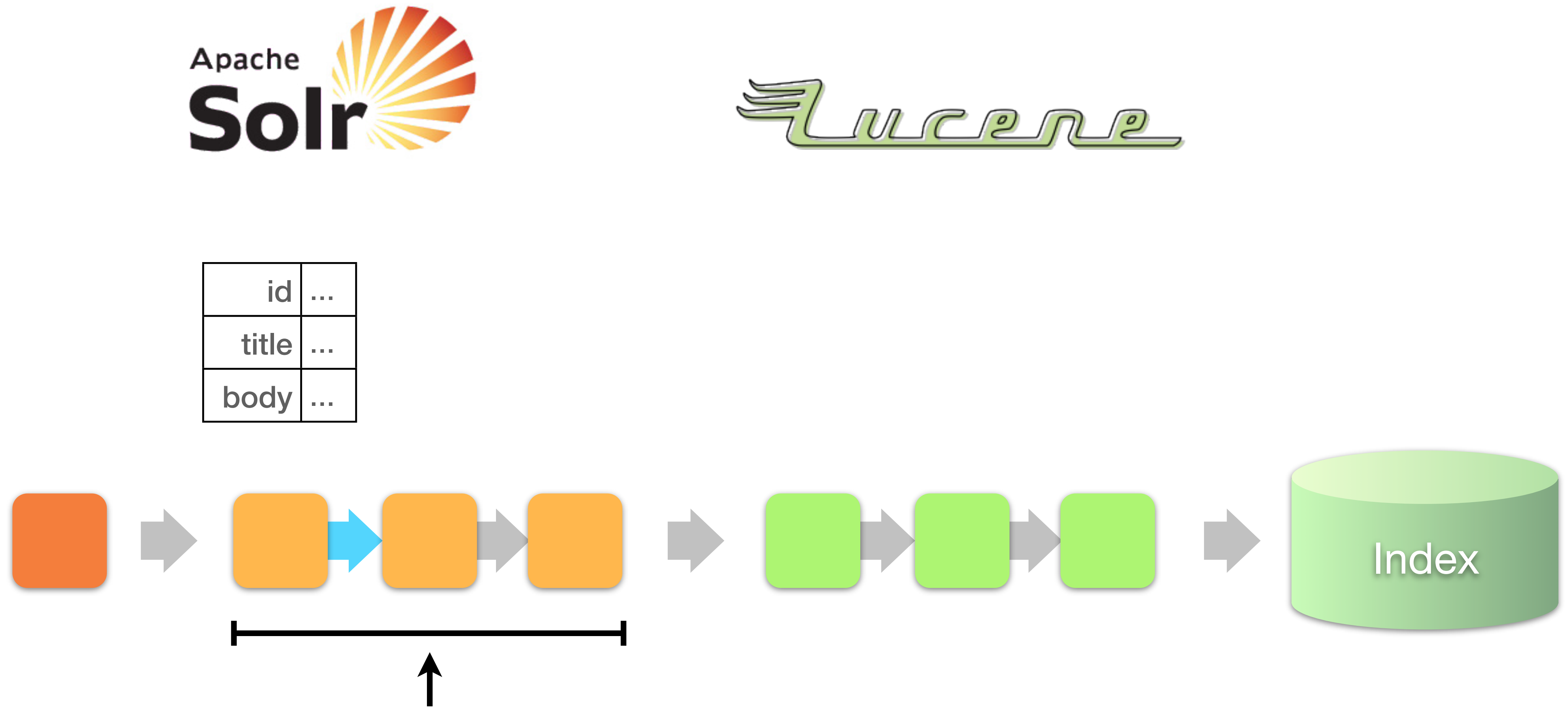
Adding document details



Update chain of **UpdateRequestProcessors**

1. Processes a document at a time with operation (add)
2. Plugin logic can mutate SolrInputDocument, i.e. add fields or do other processing as desired

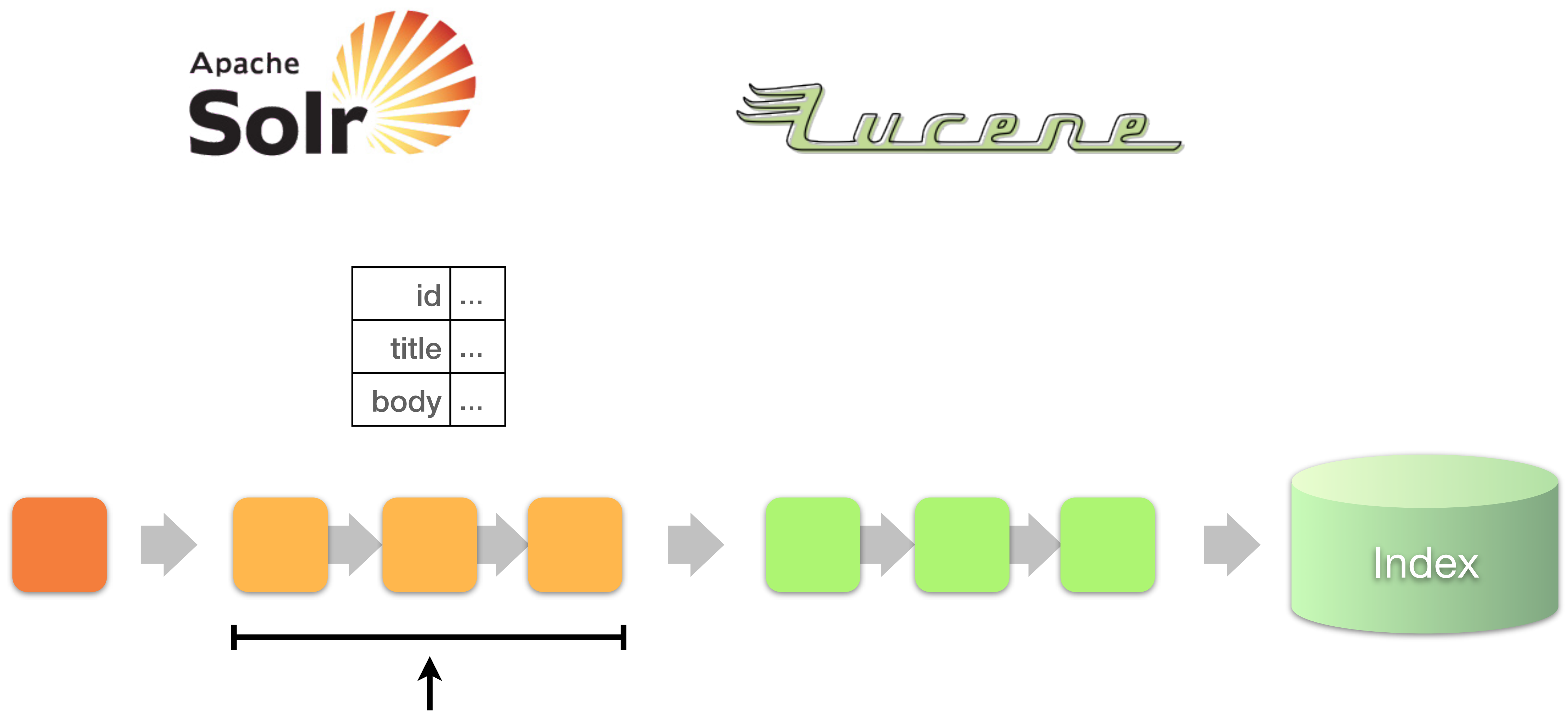
Adding document details



Update chain of **UpdateRequestProcessors**

1. Processes a document at a time with operation (add)
2. Plugin logic can mutate SolrInputDocument, i.e. add fields or do other processing as desired

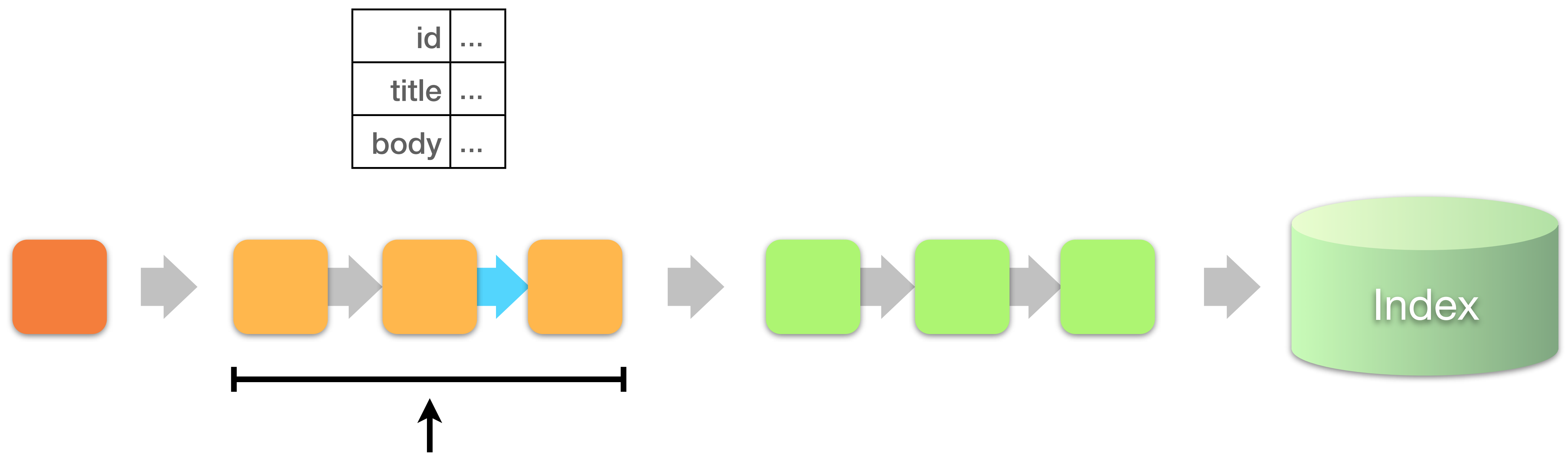
Adding document details



Update chain of **UpdateRequestProcessors**

1. Processes a document at a time with operation (add)
2. Plugin logic can mutate SolrInputDocument, i.e. add fields or do other processing as desired

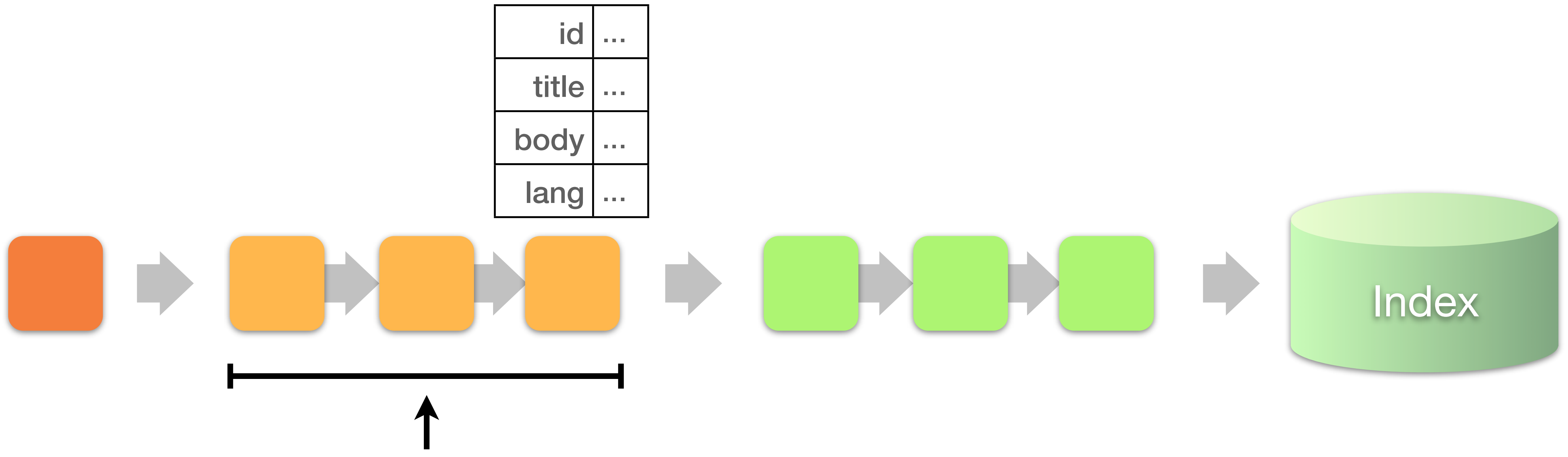
Adding document details



Update chain of **UpdateRequestProcessors**

1. Processes a document at a time with operation (add)
2. Plugin logic can mutate SolrInputDocument, i.e. add fields or do other processing as desired

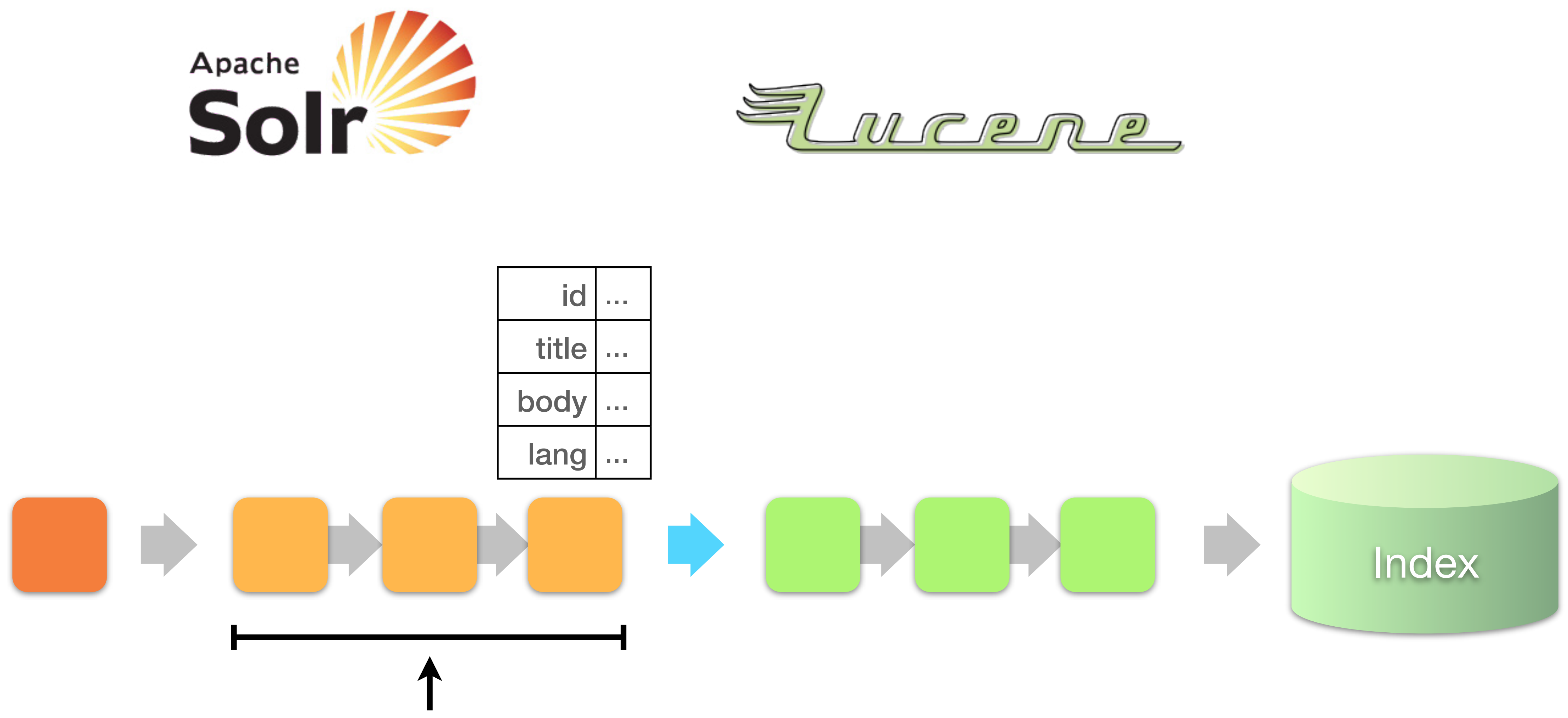
Adding document details



Update chain of **UpdateRequestProcessors**

1. Update processor added a lang field by analyzing body
2. Finish by calling **RunUpdateProcessor** (usually)

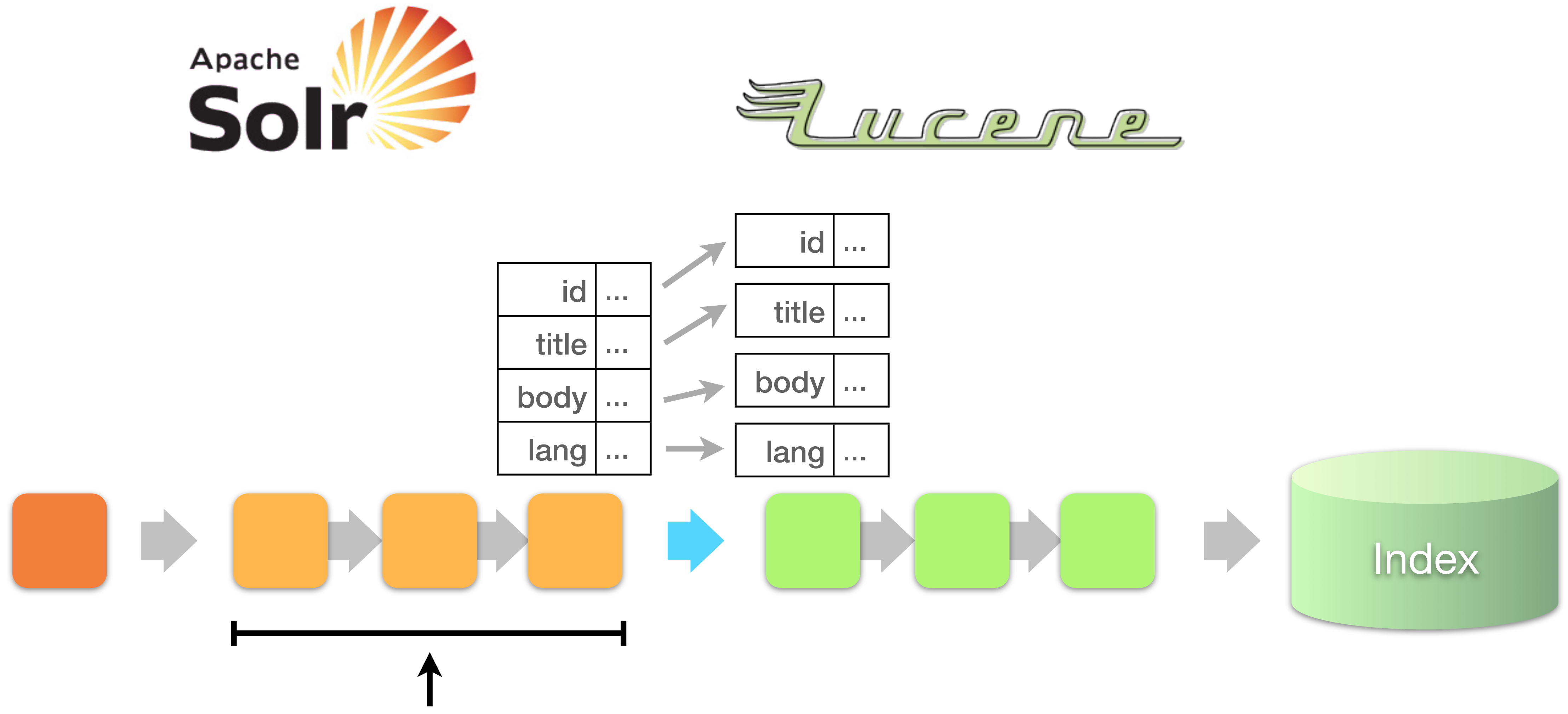
Adding document details



Update chain of **UpdateRequestProcessors**

1. Update processor added a lang field by analyzing body
2. Finish by calling **RunUpdateProcessor** (usually)

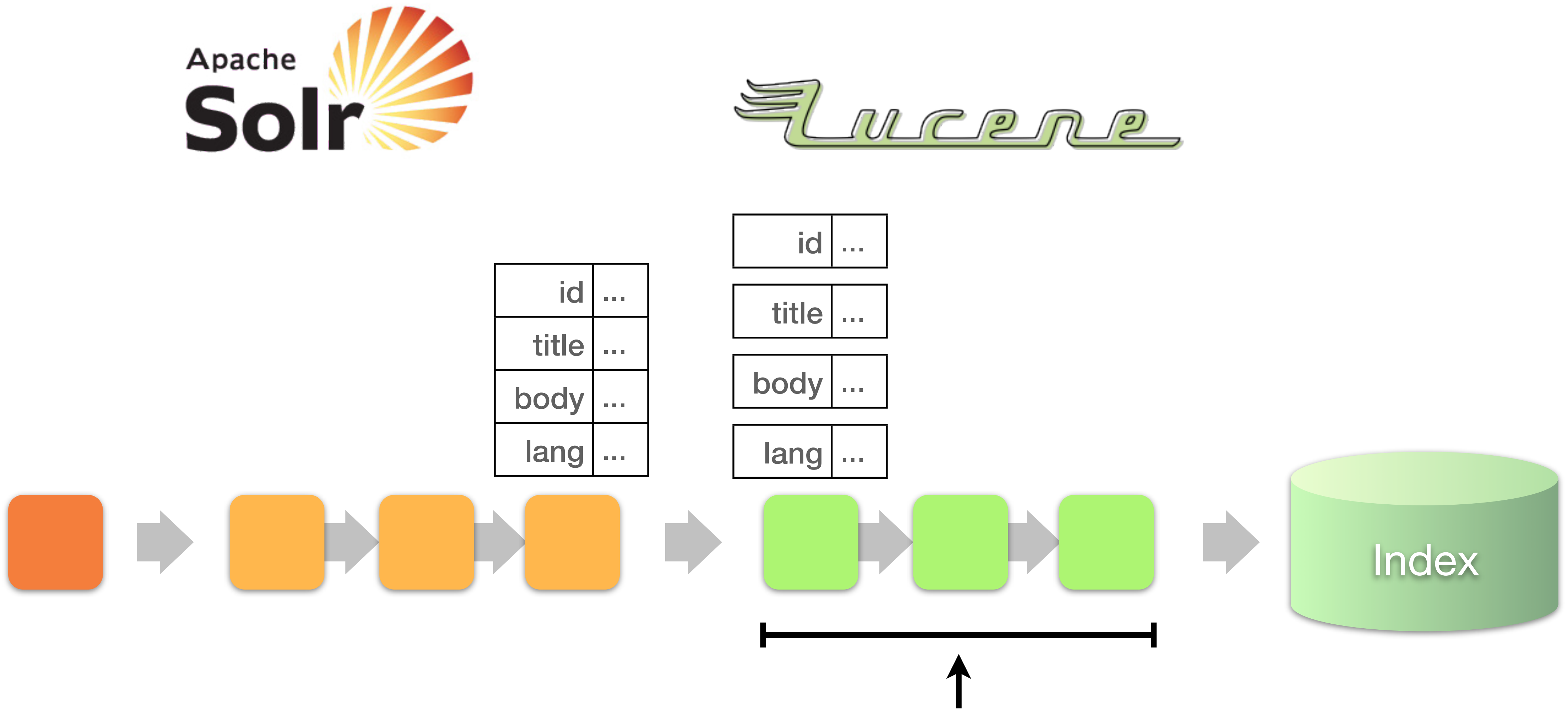
Adding document details



Update chain of **UpdateRequestProcessors**

1. Update processor added a lang field by analyzing body
2. Finish by calling **RunUpdateProcessor** (usually)

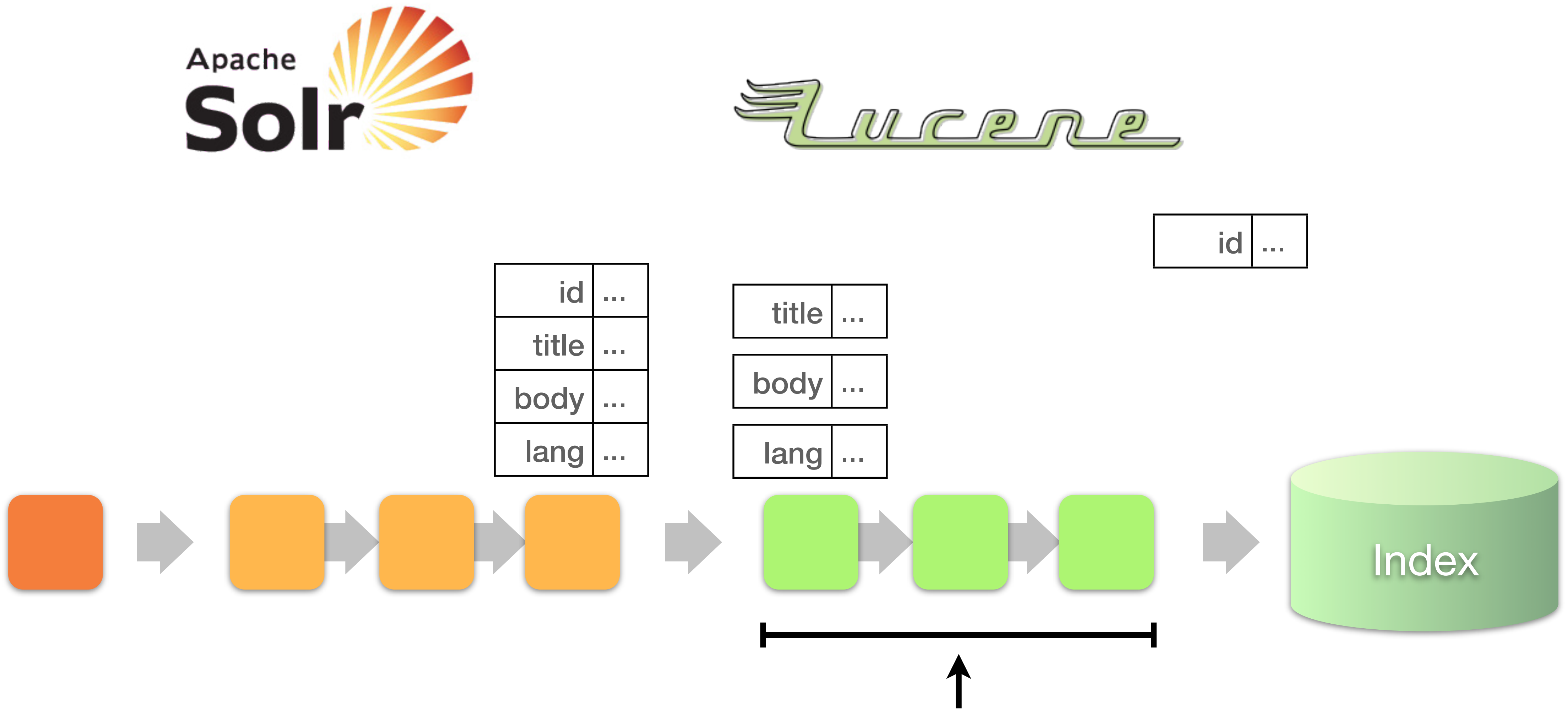
Adding document details



Lucene analyzer chain

1. Fields are analyzed individually

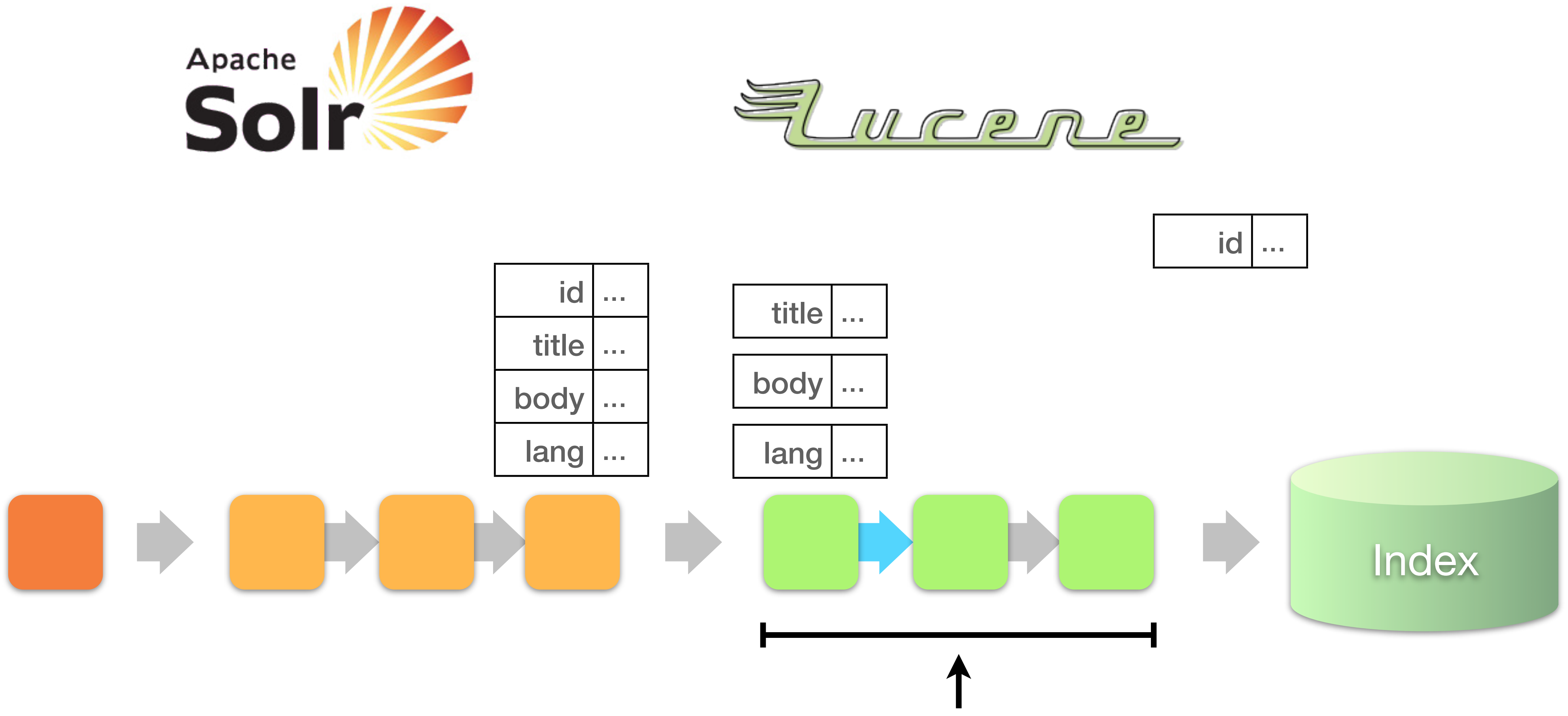
Adding document details



Lucene analyzer chain

1. No analysis on id

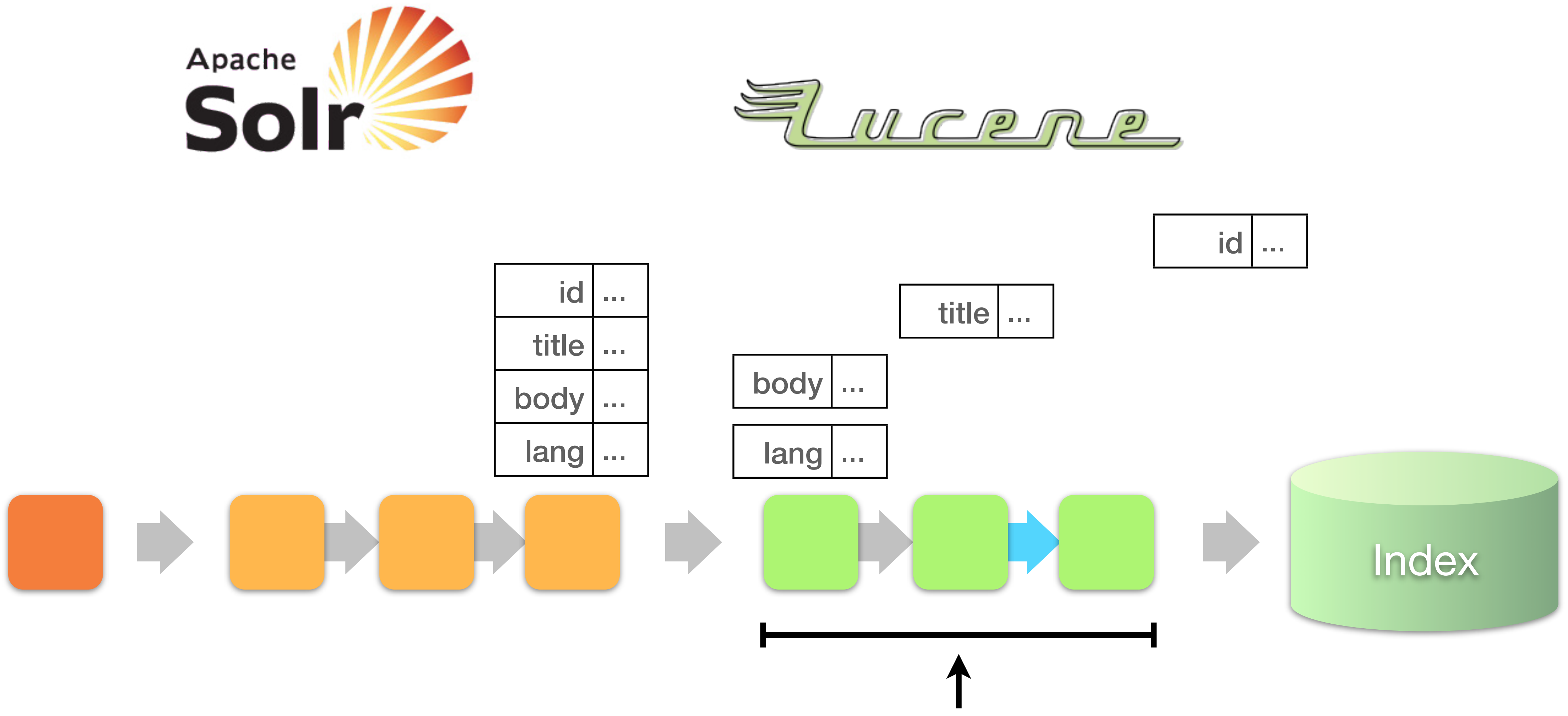
Adding document details



Lucene analyzer chain

1. Field title being processed

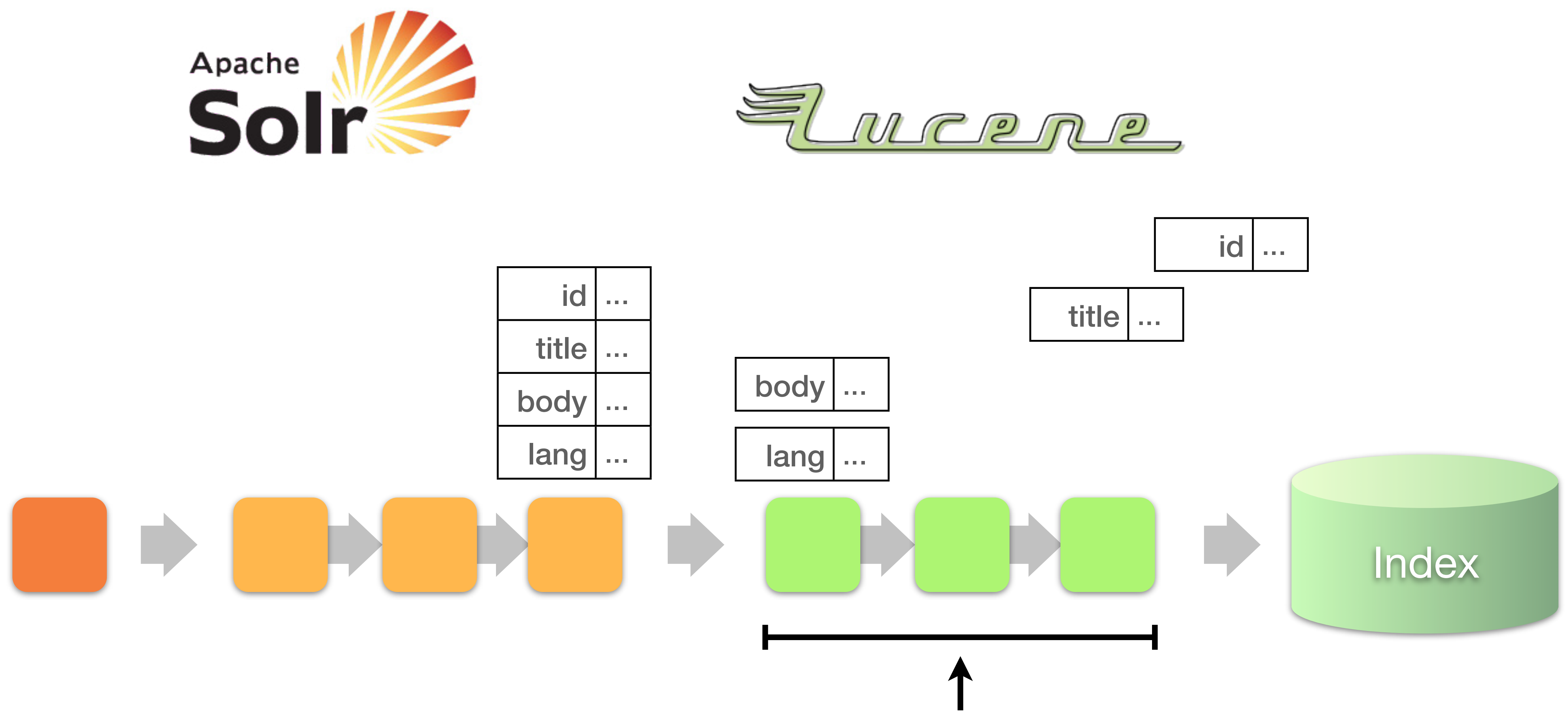
Adding document details



Lucene analyzer chain

1. Field title being processed

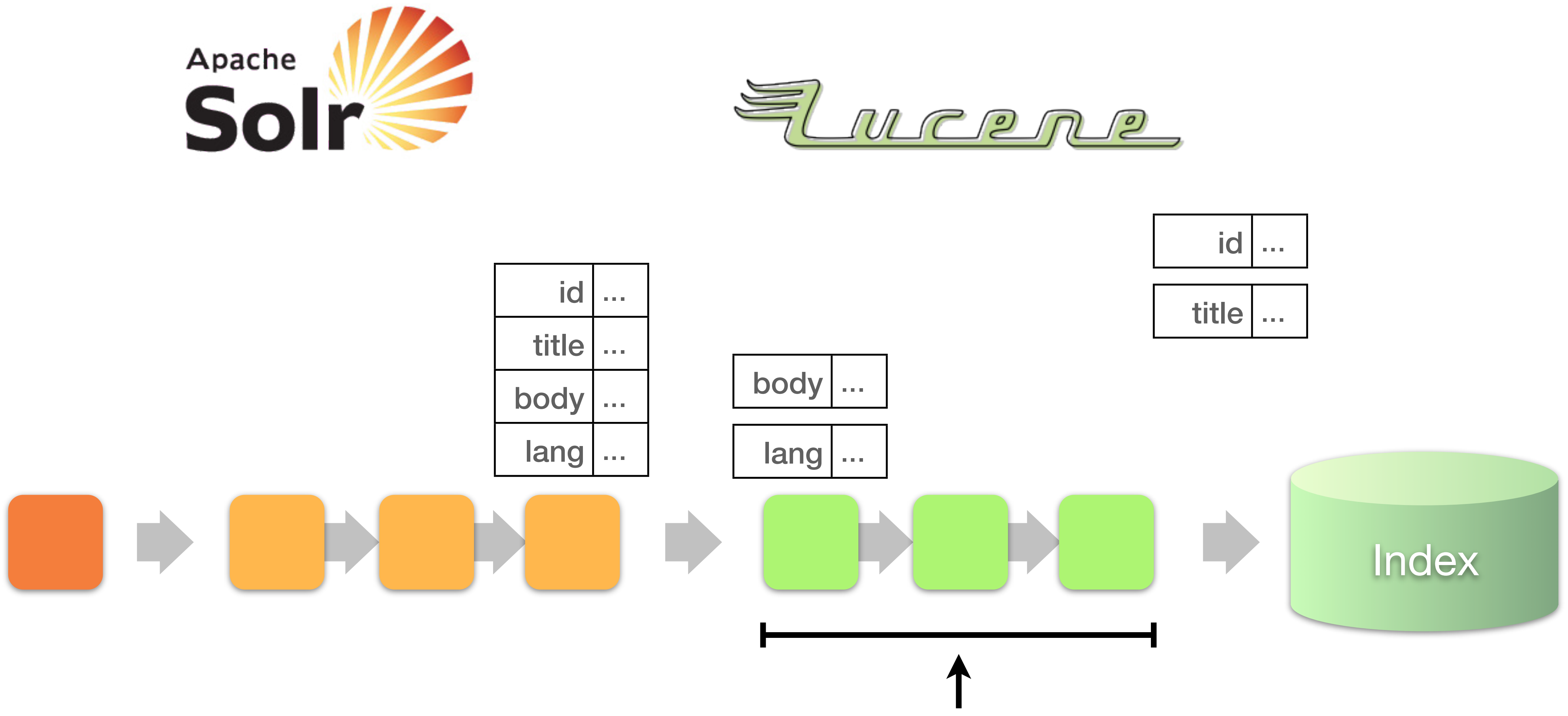
Adding document details



Lucene analyzer chain

1. Field title being processed

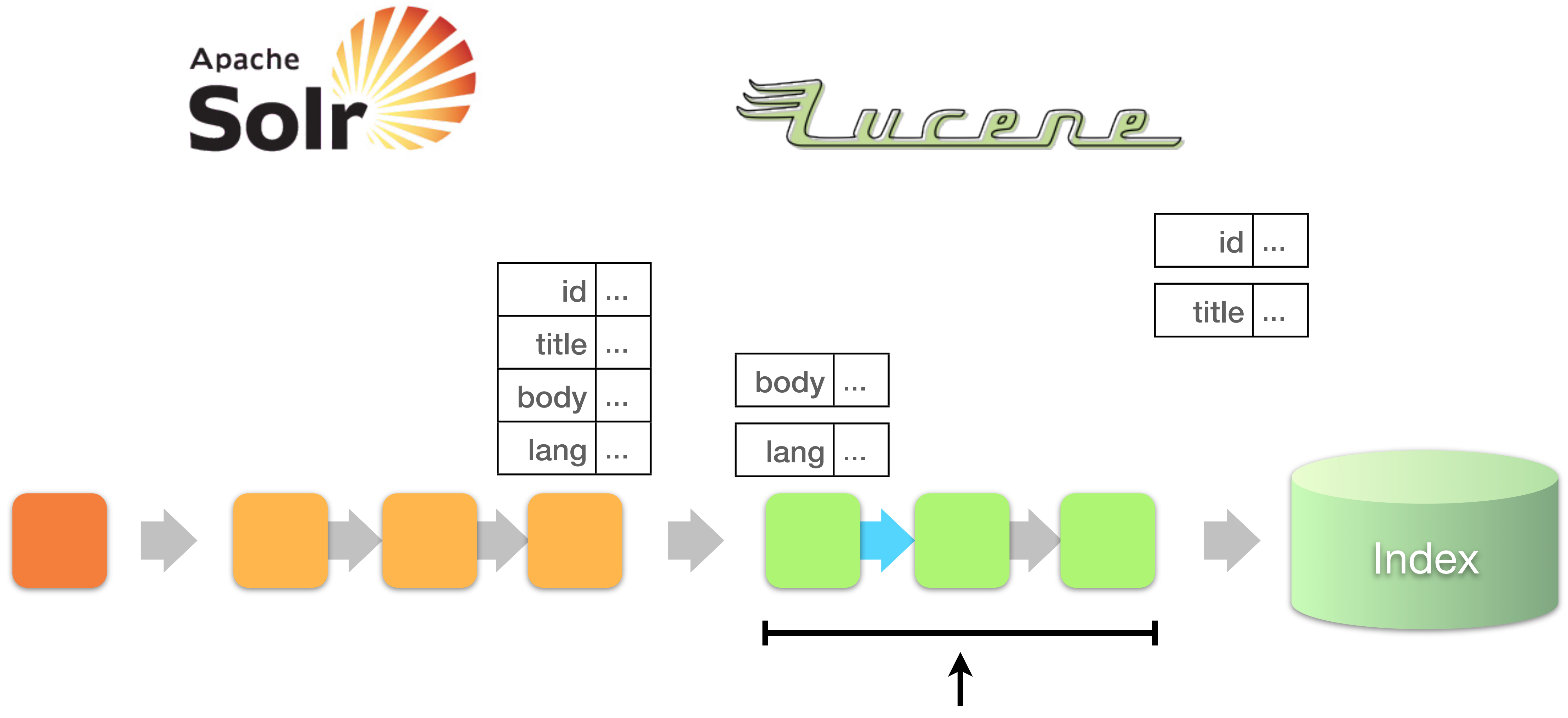
Adding document details



Lucene analyzer chain

1. Field title being processed

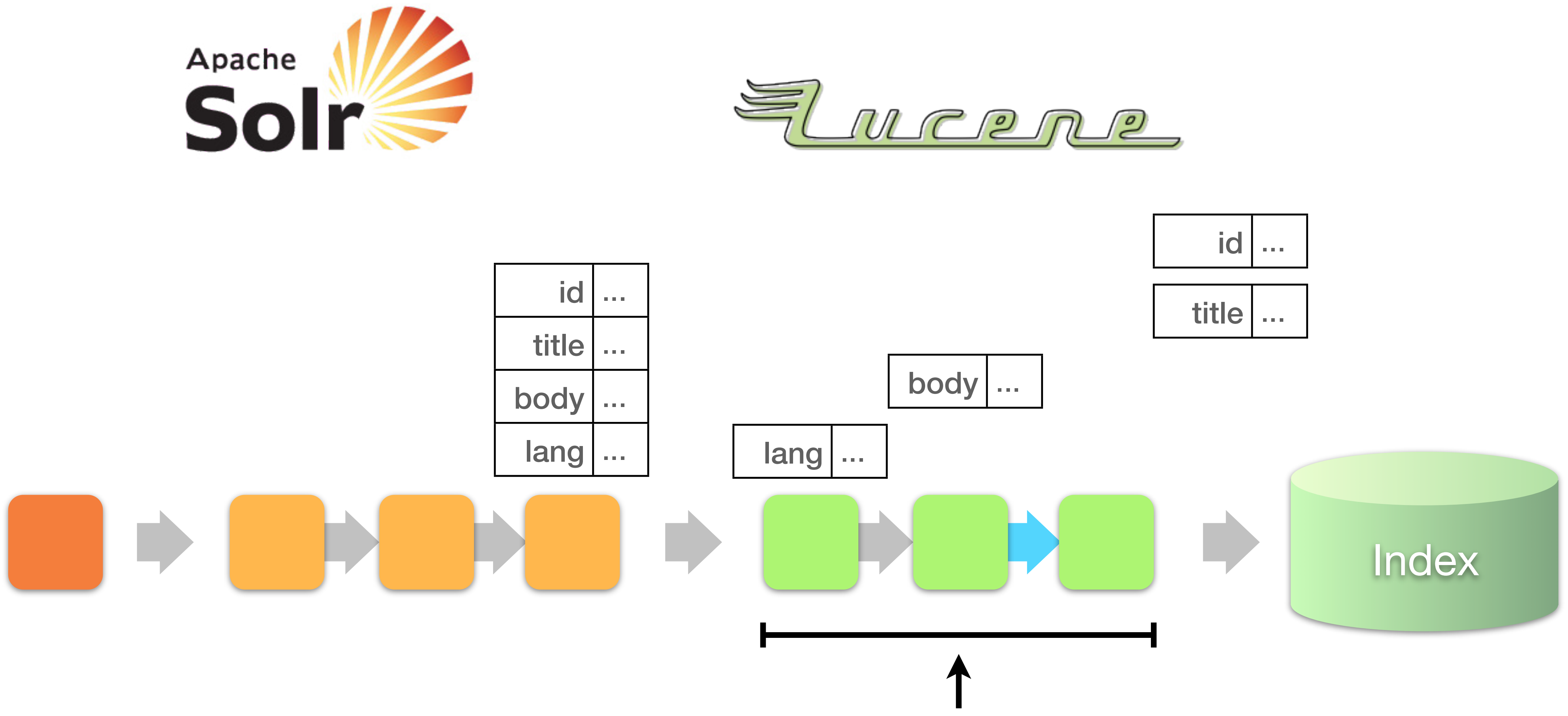
Adding document details



Lucene analyzer chain

1. Field <u>body</u> being processed

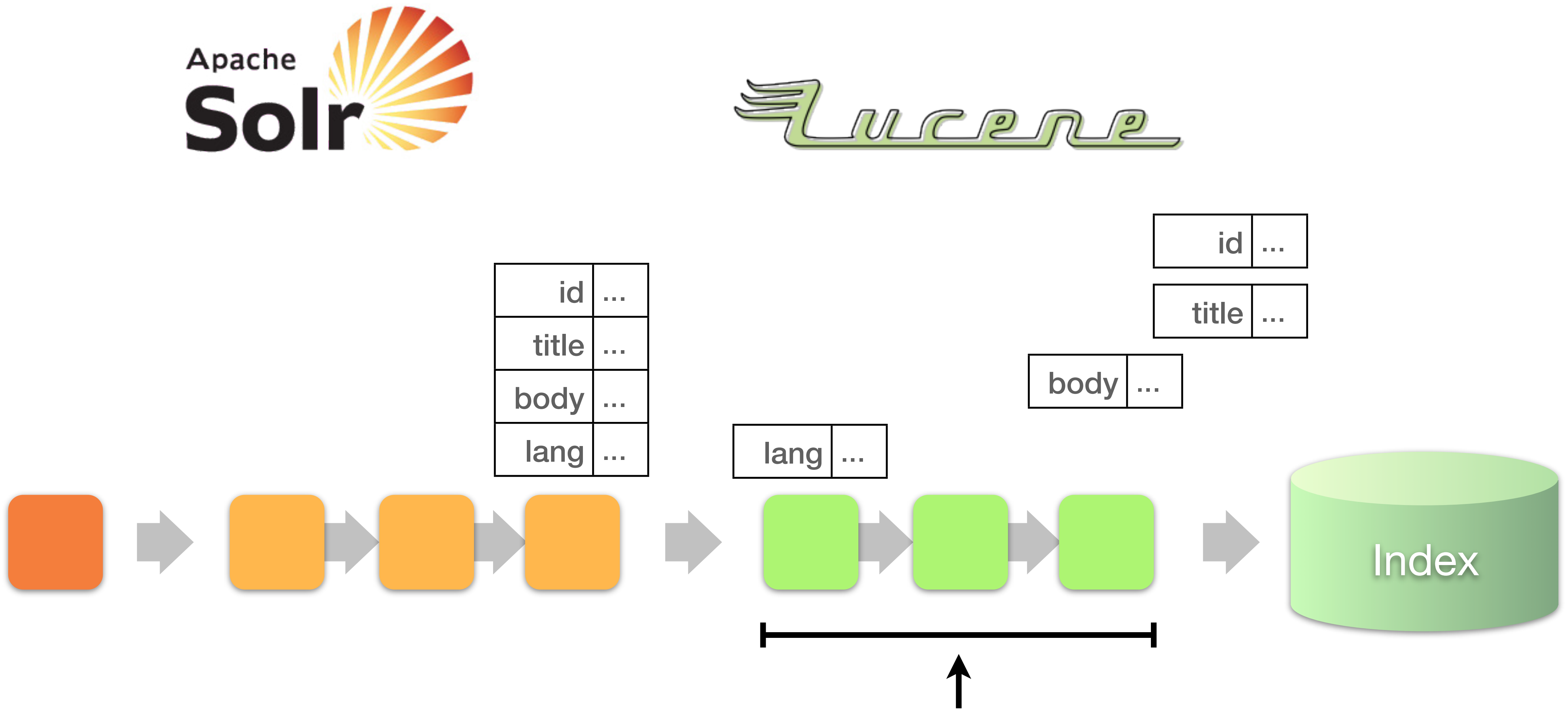
Adding document details



Lucene analyzer chain

1. Field <u>body</u> being processed

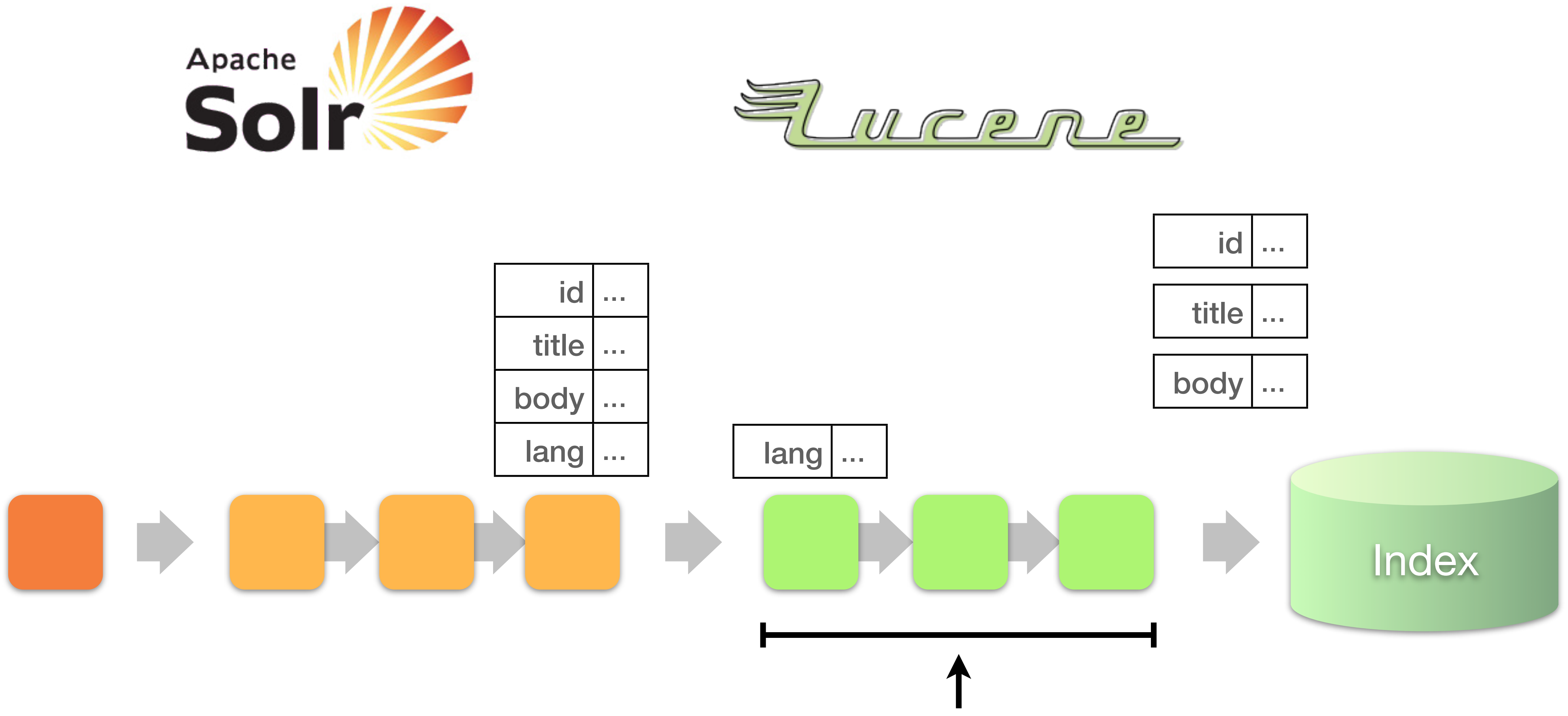
Adding document details



Lucene analyzer chain

1. Field body being processed

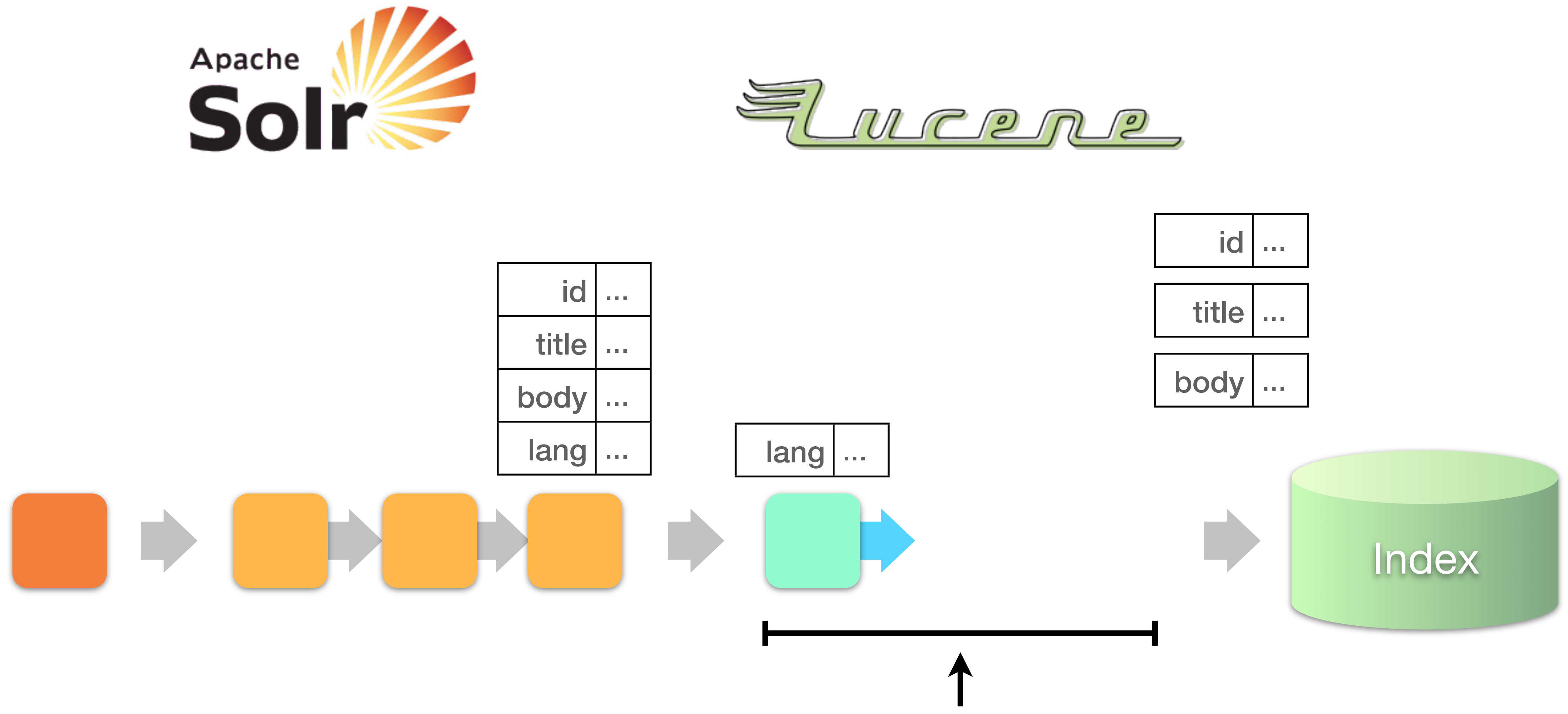
Adding document details



Lucene analyzer chain

1. Field body being processed

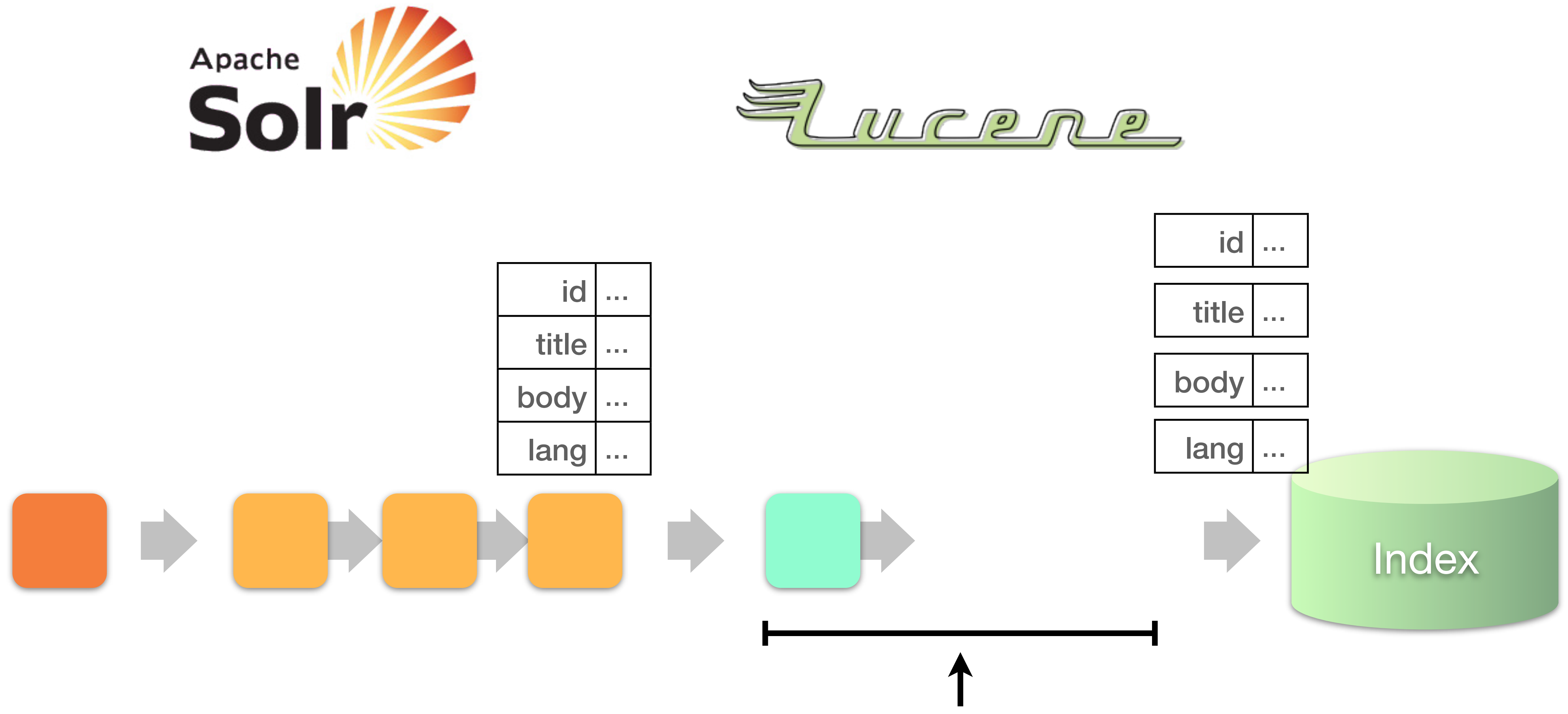
Adding document details



Lucene analyzer chain

1. Field <u>lang</u> being processed
2. User a different analyzer chain

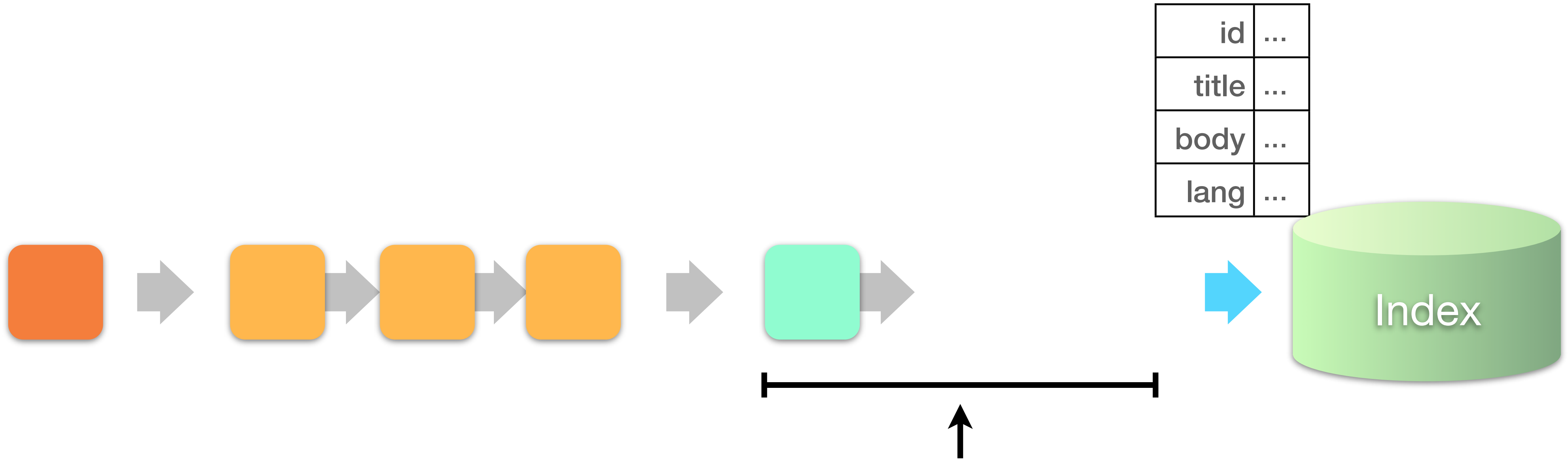
Adding document details



Lucene analyzer chain

1. Field <u>lang</u> being processed
2. User a different analyzer chain

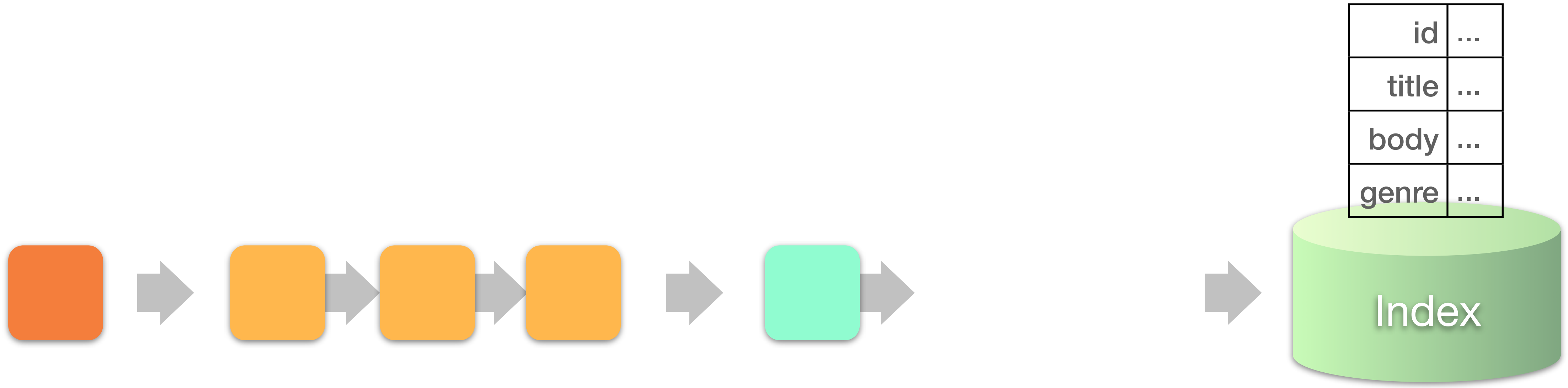
Adding document details



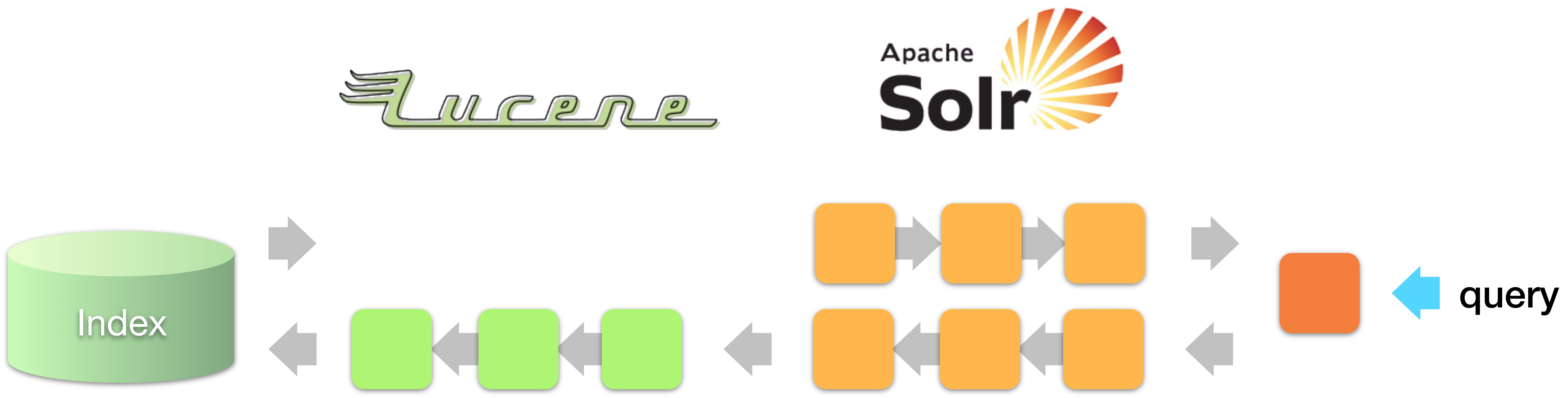
Lucene analyzer chain

1. All fields analyzed

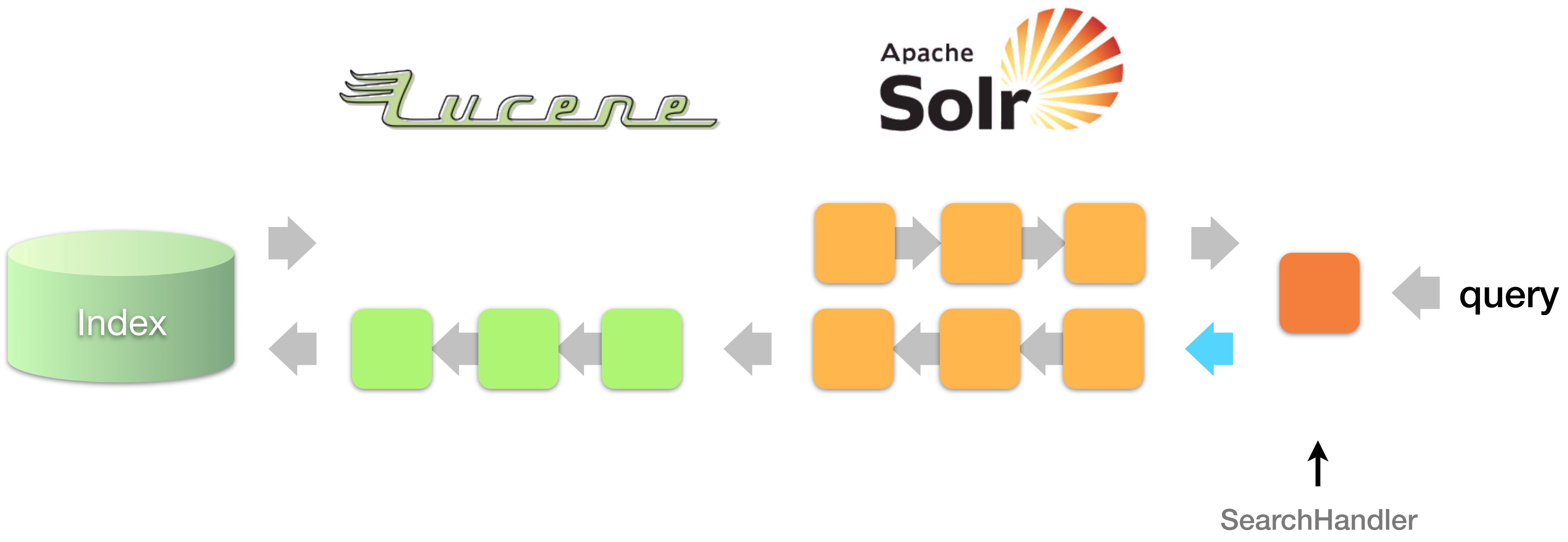
Adding document details



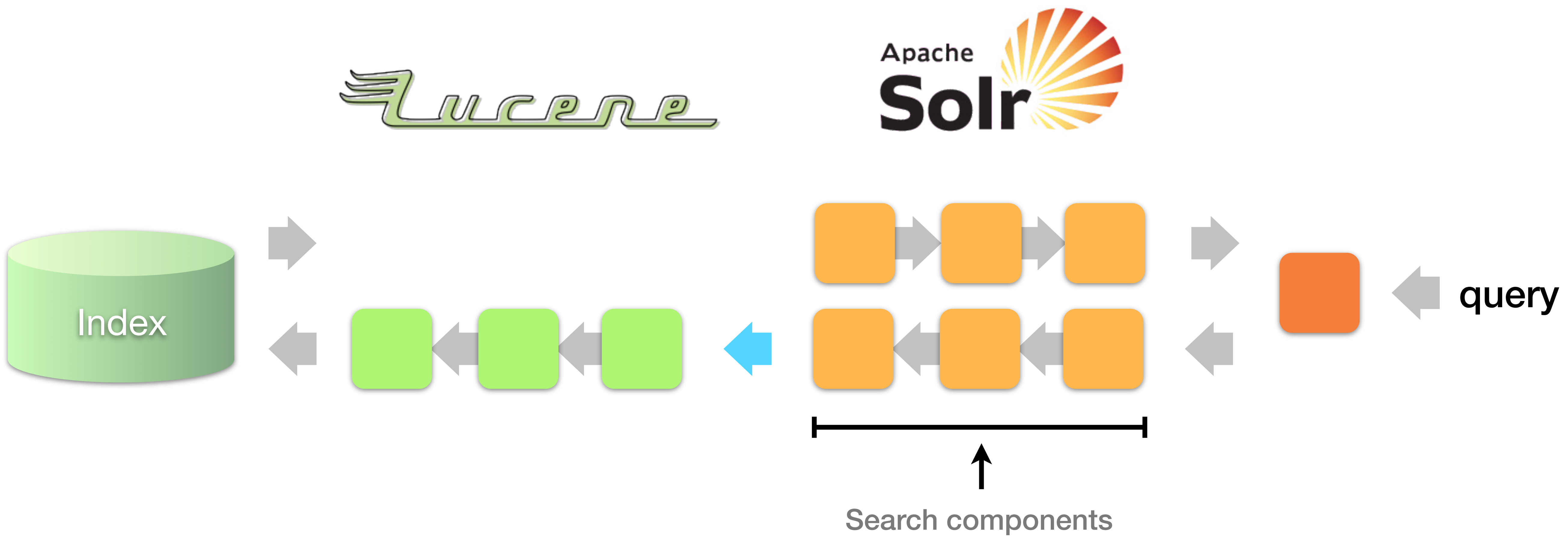
Search details



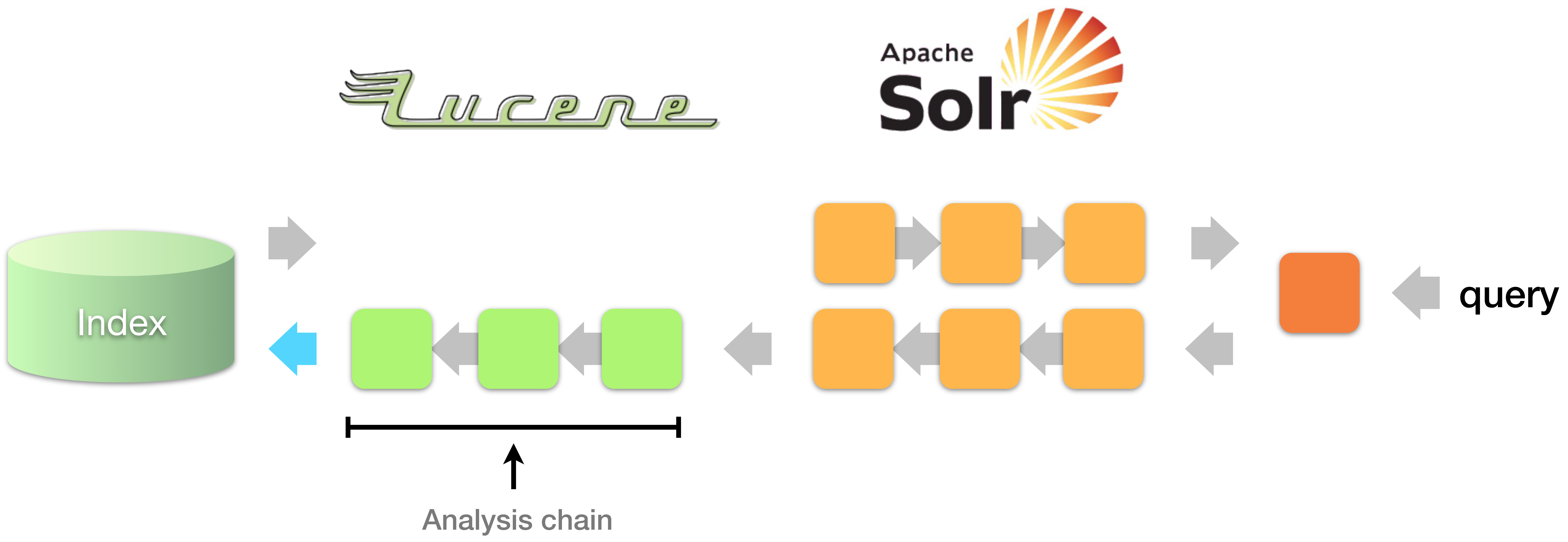
Search details



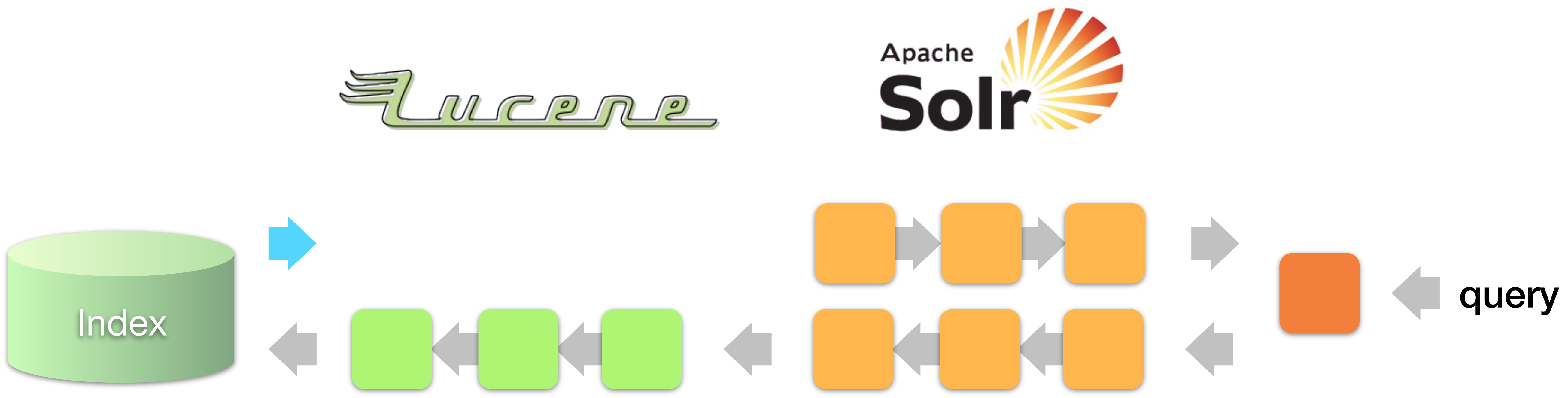
Search details



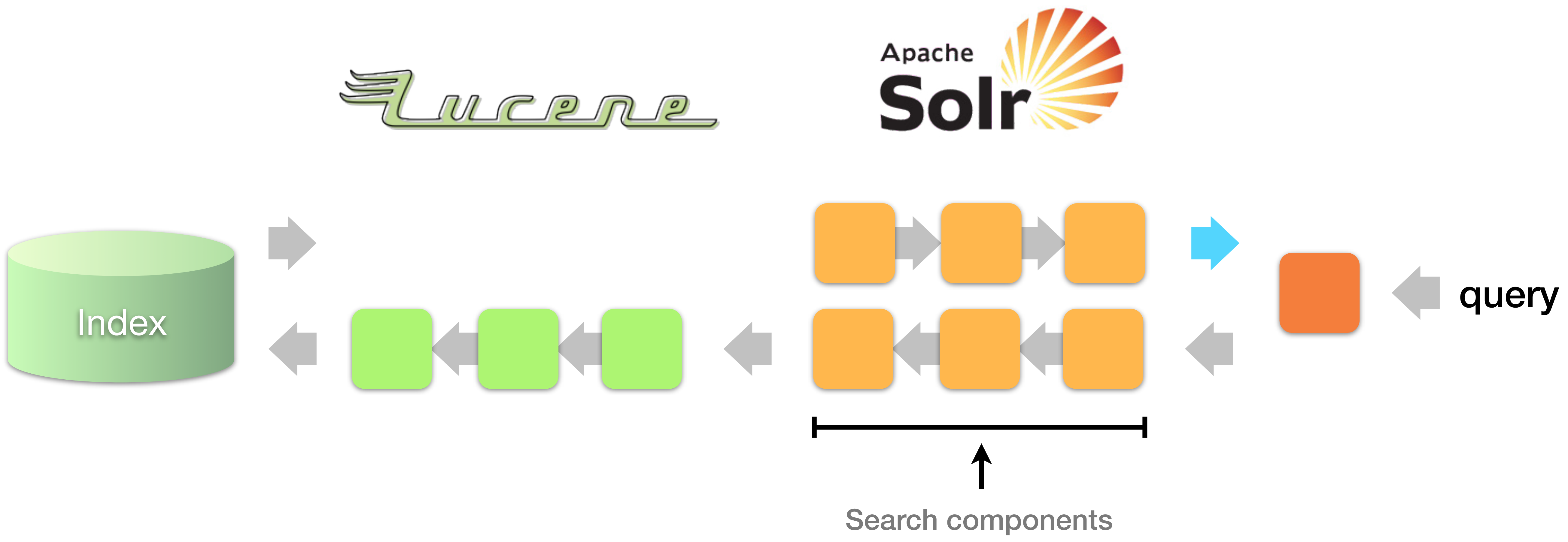
Search details



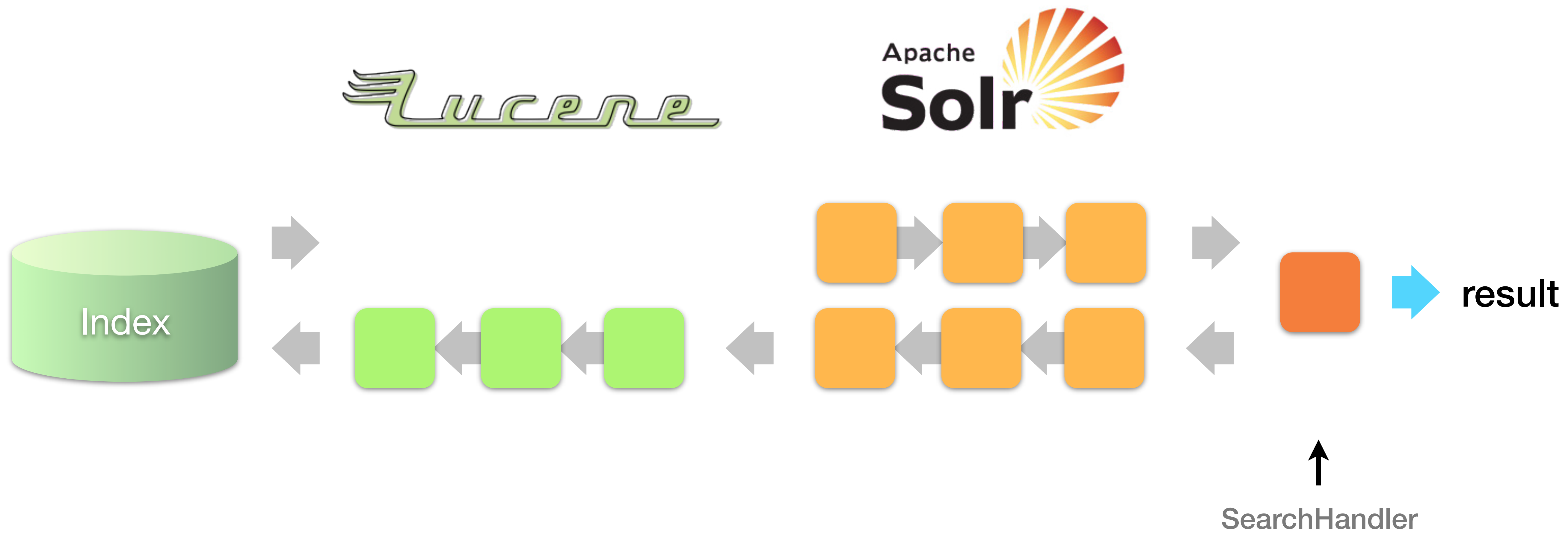
Search details



Search details



Search details



Hands-on: Multi-lingual search with Solr

Multi-language challenges

- How do we detect language accurately?
  - Indexing side is feasible (accuracy > 99.1%), but query side is hard because of ambiguity
- How to deal with language query side?
  - Supply language to use in the application (best if possible)
  - Search all relevant language variants (OR query)
  - Search a fallback field using n-gramming
  - Boost important language or content

Not knowing query term language will most likely impact negatively on overall rank

NLP eco-system

Basis Technology

- High-end provider of text analytics software
- Rosette Linguistics Platform (RLP) highlights
  - Language and encoding identification
    (55 languages and 45 encodings)
  - Segmentation for Chinese, Japanese and Korean
  - De-compounding for German, Dutch, Korean, etc.
  - Lemmatization for a range of languages
  - Part-of-speech tagging for a range of language
  - Sentence boundary detection
  - Named entity extraction
  - Name indexing, transliteration and matching
- Integrates well with Lucene/Solr

Apache OpenNLP

- Machine learning toolkit for NLP
  - Implements a range of common and best-practice algorithms
  - Very easy-to-use tools and APIs targeted towards NLP
- Features and applications
  - Tokenization
  - Sentence segmentation
  - Part-of-speech tagging
  - Named entity recognition
  - Chunking
- Licensing terms
  - Code itself has an Apache License 2.0
  - Some models are available, but licensing terms and F-scores are unclear...
- See LUCENE-2899 for OpenNLP a Lucene Analyzer (work-in-progress)

Hands-on: Basic text processing with OpenNLP

Other eco-system options

Stanford University S NLP Natural Language Processing

GATE general architecture for text engineering

UIMA Unstructured Information Management Architecture An Apache Project. TM

CLEAR NLP

Summary

Summary

- Getting languages right is a hard problem
  - Linguistics helps improve search quality
- Linguistics in Lucene, ElasticSearch and Solr
  - A wide range of languages are supported out-of-the-box
  - Considerations to be made on indexing and query side
  - Lucene Analyzers work on a per-field level
  - Solr UpdateRequestProcessors work on the document level
  - Solr has functionality for automatically detecting language (available in ElasticSearch as a plugin)
- Linguistics options also available in the eco-system

Practical advice

Practical advice

- Understand your content and your users' needs
  - Understand your language and its issues
  - Understand what users want from search
- Do you have issues with recall?
  - Consider synonyms, stemming
  - Consider compound-segmentation for European languages
  - Consider WordDelimiterFilter, phonetic matching
- Do you have issues with precision?
  - Consider using ANDs instead of ORs for terms
  - Consider improving content quality? Search fewer fields?
- Is some content more important than other?
  - Consider boosting content with a boost query

Thanks you

Jan Høydahl www.cominvent.com

Thanks for some slide material

Bushra Zawaydeh

Thanks for fun Arabic language lessons

Gaute Lambertsen

Thanks for helping talk preparations

Example code

- Example code will be available on Github

  https://github.com/atilika/berlin-buzzwords-2013

- Get started using

  ```
  git clone git://github.com/atilika/berlin-buzzwords-2013.git
  less berlin-buzzwords-2013/README.md
  ```

- Contact us if you have questions

  hello@atilika.com

Thank you very much

Vielen Dank

Merci beaucoup

شكرا جزيلا

ありがとうございました

atilika
applied search innovation