Dehyphenation of Words and Guessing Ligatures

Sumitra Magdalin Corraya

Computer Science Department
University of Freiburg

UNI FREIBURG

Goal

(1) Deciding if a hyphen in a hyphenated word is mandatory or not

(2) resolving the individual characters of ligatures.

In both goals we try to achieve high accuracy and fast response time.

Text Extraction

Is a process for extracting meaningful text from electronic documents such as PDF.

Meaningful text: how human understand language.

For example: English

Motivation

- has been extensively studied in information retrieval.
- The accurate extraction of the text from PDF document is important.

Motivation

- has been extensively studied in information retrieval.
- The accurate extraction of the text from PDF document is important.

Challenging

Reason

- PDF can contains hyphenated word.
- PDF can contains ligatures.

Reason

- PDF can contains hyphenated word.
- PDF can contains ligatures.

What is hyphenated word?

Reason

- PDF can contains hyphenated word.
- PDF can contains ligatures.

What is hyphenated word?

What is ligatures?

Why they are making difficulties?

Hyphenated word

- Is a word containing hyphen (-).
- When: two or more words joined to form a new word.
- For example: ex-president.

Ligature

Is a typographical letter.

Occurs when two or more characters are joined into a single glyph.

Glyph is a visual representation of a character in a specific font and style.

$$AE \rightarrow Æ \quad OE \rightarrow Œ$$
$$ae \rightarrow æ \quad oe \rightarrow œ$$
$$fi \rightarrow ﬁ \quad ffi \rightarrow ﬃ$$

Example of ligature[1]

[1] https://de.wikipedia.org/wiki/Ligatur_(Typografie)

Is PDF only having hyphenated word which is necessary to have hyphen?

Is PDF only having hyphenated word which is necessary to have hyphen?

No.

Reason

- PDF is a layout-based format which specifies the fonts and positions of the individual characters.
- For example: When a long word cannot fit in end of the line, PDF split the long word into two part. And two part appear in different lines.
- Example: So the first question comes to our mind why not to keep using well-known JDK collection?

So the first question comes to our mind why not to keep using well-known JDK col
lections?

Reason

How we will understand it is a one word?

Solution : Put‘s a hyphen in between of the broken words.

So the first question comes to our mind why not to keep using well-known JDK col-
lections?

Text extraction result: col-lections

Is the hyphen necessary?

Dehyphenation of words

Deciding if the hyphen in the word is necessary or not.

How we will understand hyphen is necessary or not?

Dehyphenation of words

Deciding if the hyphen in the word is necessary or not.

How we will understand hyphen is necessary or not?

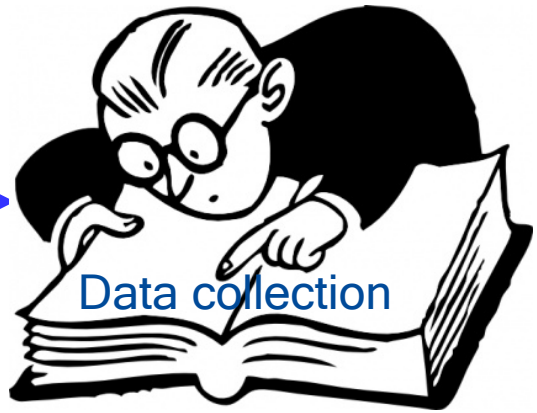
Approach

decide the necessity of the hyphen by using the huge data/words collection.

Example

Col-lections

available?



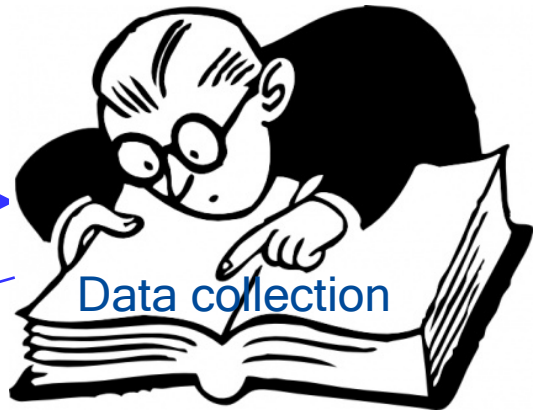
Example

Col-lections

available?

found

yes : hyphen necessary



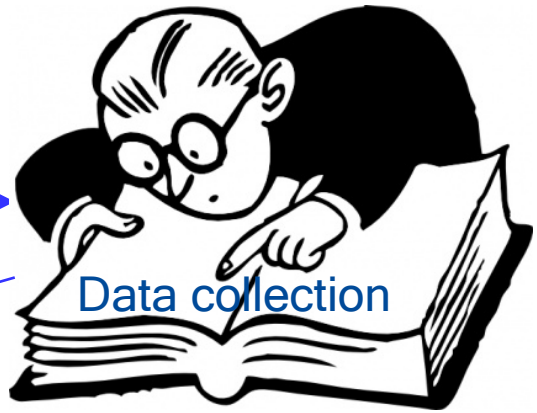
Example

Col-lections

available?

found

No : hyphen not necessary



Difficulties

Is it that easy?

Difficulties

Is it that easy?

No

Why?

There are words, which can be found with and also without hyphen in compound words.

Compound Word

- When words have a combined meaning (e.g. sugar-free)
- When there is a relationship between the words. For example "rock-forming minerals", are minerals that form rocks.
- Compound word can be both with or without a hyphen. For example: sub-tube or subtube.

Approach

We look up the frequency of the words in the huge data/words collection.

word	frequency
Sub-tube	12
Subtube	10

As result: Hyphen is necessary.

Frequency : is the number of times that a word appears in the text file.

How we get the huge data/words collection and frequency of the words?

Data/words collection

From : arXiv

ClueWeb

arXiv

is a repository that consists of scientific papers in the fields of computer science.

ClueWeb

The ClueWeb is a data-set.

Data/words collection

- arXiv has around 40,515,568 words
- ClueWeb has around 34,702,294 words
- without words that contain special character, i.e. “%,@" and numbers, i.e. 1,2.

Data/words collection

We got intotal 16,820,541 unique words with frequency (without numbers, short forms of names)

```
netherlands 1958
acid 1958
involves 1958
slowly 1958
whilst 1957
well-known 1955
miami 1953
favor 1953
hidden 1953
occasion 1953
branches 1952
composition 1950
escaped 1949
abilities 1948
manufacturing 1947
```

Ligature

Is a typographical letter.

Occurs when two or more characters are joined into a single glyph.

Glyph is a visual representation of a character in a specific font and style.

$$AE \rightarrow Æ \quad OE \rightarrow Œ$$
$$ae \rightarrow æ \quad oe \rightarrow œ$$
$$fi \rightarrow ﬁ \quad ffi \rightarrow ﬃ$$

Example of ligature[1]

[1] https://de.wikipedia.org/wiki/Ligatur_(Typografie)

Ligature in word

Mathematical formulæ are notoriously difficult to deal with for several reasons. Some of the characters might be drawn with vector operations, the vertical position varies much and is obviously significant. Outside of running text they might conceivably be stripped out, but inlined in the text they have to be dealt with. Since it is very difficult to make any sense of the extracted equations, the most important thing is not to break the segmentation of surrounding text.

When this text is extracted from the PDF, we wants to extract the individual characters within the word and not the special character.

How can get the individual characters/meaning for glyph?

Guessing ligature

- There are 20 different ligature in English language.

For example: aa, ae, ao, au, av, ay, et, ff, ffi, ffl, fi, fl, oe, oo, fs, st, ft, tz, ue, vy.

- For the glyph as an illustration we have used “$”.

For example: formulae as formul$.

Guessing ligature

- There are 20 different ligature in English language.

For example: aa, ae, ao, au, av, ay, et, ff, ffi, ffl, fi, fl, oe, oo, fs, st, ft, tz, ue, vy.

- For the glyph as an illustration we have used “$”.

For example: formulae as formul$.

Approach

- "trial and error" style.
- try one after the other possible ligatures and check is the word available in the huge data/words collection.

Example

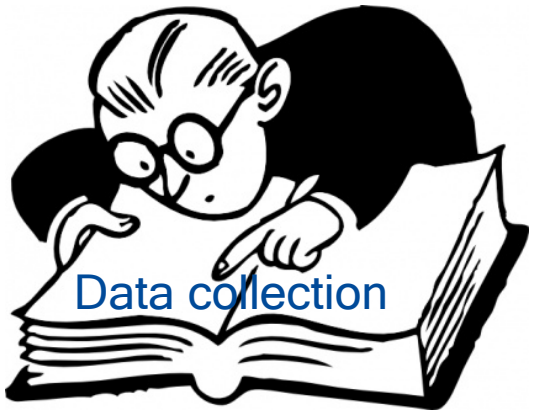
aa
ae
ao
au
av
ay
et
ff
ffi
ffl
fi
fl
oe
oo
fs
So on.....

formul$

Example

aa
ae
ao
au
av
ay
et
ff
ffi
ffl
fi
fl
oe
oo
fs
So on.....

formulaa → Available?

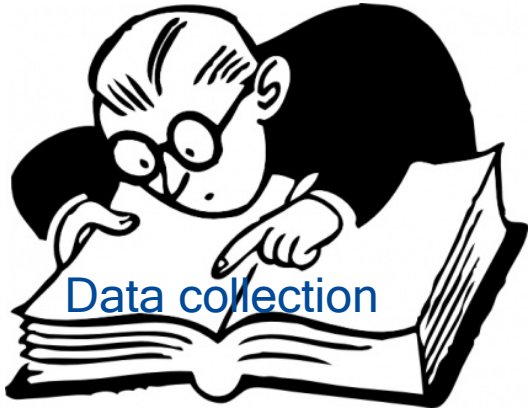


Example

aa
ae
ao
au
av
ay
et
ff
ffi
ffl
fi
fl
oe
oo
fs
So on.....

formulaa

Available?
No



Example

aa
ae
ao
au
av
ay
et
ff
ffi
ffl
fi
fl
oe
oo
fs
So on.....

formulae → Available? Yes

Difficulties

1. There is more then one ligature in word. For example: Fluffy as uy.
2. Ambiguities

```
Input:   $u$y
Output:  stuffy
Output:  fluffly
Output:  fluffy
Output:  flusty
```

Overcome from difficulties

aa
ae
ao
au
av
ay
et
ff
ffi
ffl
fi
fl
oe
oo
fs
So on.....

uy

Overcome from difficulties

aa
ae
ao
au
av
ay
et
ff
ffi
ffl
fi
fl
oe
oo
fs
So on.....

aau$y

Overcome from difficulties

aa	aa
ae	ae
ao	ao
au	au
av	av
ay	ay
et	et
ff	ff
ffi	ffi
ffl	ffl
fi	fi
fl	fl
oe	oe
oo	oo
fs	fs
So on.....	So on.....

aauaay

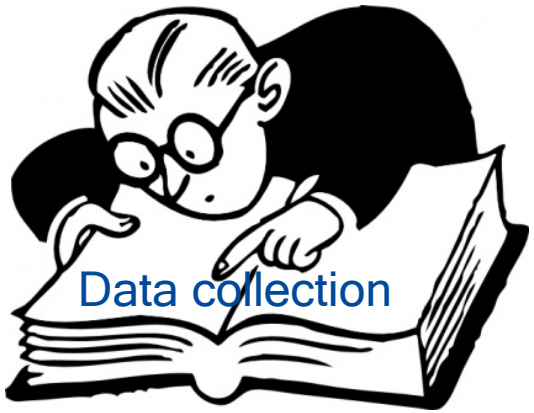
Overcome from difficulties

aa	aa
ae	ae
ao	ao
au	au
av	av
ay	ay
et	et
ff	ff
ffi	ffi
ffl	ffl
fi	fi
fl	fl
oe	oe
oo	oo
fs	fs
So on.....	So on.....

aauaay

Available?

No



Overcome from difficulties

aa	aa
ae	ae
ao	ao
au	au
av	av
ay	ay
et	et
ff	ff
ffi	ffi
ffl	ffl
fi	fi
fl	fl
oe	oe
oo	oo
fs	fs
So on.....	So on.....

Approach for ambiguities

```
Input:   $u$y
Output:  stuffy
Output:  fluffly
Output:  fluffy
Output:  flusty
```

Approach is to look for the frequency of the words.

Approach for ambiguities

For example:

Word	Frequency
stuffy	3
fluffy	7
fluffly	1
flusty	1

Result: fluffy.

Goal

(1) Deciding if a hyphen in a hyphenated word is mandatory or not

(2) resolving the individual characters of ligatures.

In both goals we try to achieve high accuracy and fast response time.

Run time

Name of 3 third party libraries Java support

- GNU Trove
- LMDB
- MapDB

GNU Trove

- contains a set of primitive collection and supports the use of custom hashing strategies.
- GNU Trove gives better performance and memory consumption than JDK collection [Vor14].
- On each startup GNU Trove load data into memory, performs calculations and exits.

[Vor14] Vorontsov, Mikhail. Trove library: using primitive collections for performance. 2014.

MapDB

- MapDB provides a reliable, full-featured and "tune-able" database engine using the Java Collections API.
- Collection from MapDB (for example HashMap) can be stored in a file.

Code for store hashMap in memory or in a file

```
DB db = DBMaker.fileDB("file.db").make();
ConcurrentMap map = db.hashMap("map").make();
```

- Prevent to load data in database every startup.

Check map.getHasValues*();*

LMDB

- The lightning memory-mapped database manager.
- LMDB create/open database as followed

```
Env env = new Env();
env.setMapSize(5 * 1024 * 1024 * 1024);
env.open(System.getProperty("user.dir"));
Database db = env.openDatabase();
```

- Prevent to load data in database every startup.

```
Stat stat = db.stat();
long entries = stat.getEntries();
check entries > 0;
```

Evaluation result

Six experiments for each library to decide hyphenation for 4,987,829 words

Libraries	Average runtime for searching word	Accuracy
GNU Trove	0.0378	86%
Java	0.0682	86%
LMDB	0.2521	86%
MapDB	1.3991	86%

The table shows average runtime in millisecond from six experimental results of the runtime and accuracy in percentage for deciding hyphenation in word.

Evaluation result

Five experiments for each library to find ligature for 4,546,311 words.

Libraries	Average runtime for searching word	Accuracy
GNU Trove	0.0808	93%
Java	0.0720	93%
LMDB	0.1270	93%
MapDB	0.0187	93%

The table shows average runtime in millisecond from five experimental results of the runtime and accuracy in percentage to find ligature.

Accuracy

- is the number of correct results while comparing the result with expected output.

```
$u$y fluffy
$lm film
o$en often
le$ left
$u$y stuffy
di$cult difficult
signi$cant significant
e$cient efficient
```

```
sub-arctic true
neo-christian true
trans-siberian true
pro-canadian true
un-american true
re-cover true
re-press true
un-happy false
```

Input text files with expecting output

- 86% accuracy to decide hyphen in the word necessary or not and 93% accuracy to find out expected ligature.

Future work

Deciding hyphen in the word where word contain more then one hyphen.

For example: Pick-me-up.

Questions?