LINSEN

AN EFFICIENT APPROACH TO SPLIT IDENTIFIERS AND EXPAND ABBREVIATIONS

Anna Corazza, Sergio Di Martino, Valerio Maggio

Università di Napoli "Federico II"

26th Sept. 2012, ICSM2012@Riva del Garda(Trento), Italy

MOTIVATIONS

IR FOR SE

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

MOTIVATIONS

IR FOR SE

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

IR FOR NATURAL LANGUAGE

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

IR FOR NATURAL LANGUAGE

1. Tokenization

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

IR FOR NATURAL LANGUAGE

1. Tokenization

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

Draws, the, are,
NullHandle,
box, r,
Rectangle, g,
Graphics,
box,
displayBox,
...

IR FOR NATURAL LANGUAGE

1. Tokenization

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

Draws, the, are,
NullHandle,
box, r,
Rectangle, g,
Graphics,
box,
displayBox,
...

2. Normalization

IR FOR NATURAL LANGUAGE

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

1. Tokenization

Draws, the, are,
NullHandle,
box, r,
Rectangle, g,
Graphics,
box,
displayBox,
...

2. Normalization

1. Change to Lower case

draws, the, are,
nullhandle,
box, r,
rectangle, g,
graphics,
box,
displaybox,
...

IR FOR NATURAL LANGUAGE

1. Tokenization

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

Draws, the, are,
NullHandle,
box, r,
Rectangle, g,
Graphics,
box,
displayBox,
...

2. Normalization

1. Change to Lower case
2. Remove StopWords

draws, **the**, **are**,
nullhandle,
box, r,
rectangle, g,
graphics,
box,
displaybox,
...

IR FOR NATURAL LANGUAGE

1. Tokenization

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

Draws, the, are,
NullHandle,
box, r,
Rectangle, g,
Graphics,
box,
displayBox,
...

2. Normalization

1. Change to Lower case
2. Remove StopWords
3. Apply Stemming
4. ...

draw, **the**, **are**,
nullhandl,
box, r,
rectangl, g,
graphic,
box,
displaybox,
...

IR FOR NATURAL LANGUAGE

1. Tokenization

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

Draws, the, are,
NullHandle,
box, r,
Rectangle, g,
Graphics,
box,
displayBox,
...

2. Normalization

1. Change to Lower case
2. Remove StopWords
3. Apply Stemming
4. ...

draw, **the**, **are**,
nullhandl,
box, r,
rectangl, g,
graphic,
box,
displaybox,
...

Implicit assumption: The "same" words are used whenever a particular concept is described

IR FOR NATURAL LANGUAGE

1. Tokenization

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

Draws, the, are,
NullHandle,
box, r,
Rectangle, g,
Graphics,
box,
displayBox,
...

2. Normalization

1. Change to Lower case
2. Remove StopWords
3. Apply Stemming
4. ...

draw, **the**, **are**,
nullhandl,
box, r,
rectangl, g,
graphic,
box,
displaybox,
...

Implicit assumption: The "same" words are used whenever a particular concept is described

IR FOR SOURCE CODE

1. Tokenization

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

1.5 Identifier Splitting

2. Normalization

IR FOR SOURCE CODE

1. Tokenization

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

1.5 Identifier Splitting

2. Normalization

draw, ~~**the**~~, ~~**are**~~,
null, handl,
box, r,
rectangl, g,
graphic,
box,
display, box,
...

- **snake_case** Splitter: `r'(?<=\w)_'`
  - display_box ==> display | box
- **camelCase/PascalCase** Splitter: `r'(?<!^)([A-Z][a-z]+)'`
  - displayBox ==> display | Box

IR FOR SOURCE CODE

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

CODE FOR SOURCE

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

IR FOR SOURCE CODE

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

- camelCase Splitter: `r'(?<=!^)([A-Z][a-z]+)'`
  - `drawXORRect` ==> `drawXOR | Rect`
  - `drawxorrect` ==> NO SPLIT

Splitting algorithms based on *naming conventions*

are not robust enough

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

- camelCase Splitter: `r'(?<=!^)([A-Z][a-z]+)'`
  - `drawXORRect` ==> `drawXOR | Rect`
  - `drawxorrect` ==> NO SPLIT

Splitting algorithms based on *naming conventions*

are not robust enough

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

- Heavy use of **Abbreviations** in the source code

Splitting algorithms based on *naming conventions*

are not robust enough

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

- Heavy use of **Abbreviations** in the source code
- `rect` *as for* `Rectangle`
- `r` *as for* `Rectangle`

Splitting algorithms based on *naming conventions*

are not ~~robust~~ enough

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

- Heavy use of **Abbreviations** in the source code
- `rect` *as for* `Rectangle`
- `r` *as for* `Rectangle`

IDENTIFIER MAPPING

1. Tokenization

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

1.5 Identifier Mapping

2. Normalization

draw, ~~**the**~~, ~~**are**~~, **null**, **handl**, box, r, **rectangl**, g, **graphic**, box, display, box, …

- SAMURAI (Enslen, et.al , 2011)
- TIDIER (Guerrouj, et.al , 2011)
- GenTest+Normalize (Lawrie and Binkley, 2011)
- AMAP (Hill and Pollock, 2008)
- …
- **LINSEN**

LINSEN
ALGO
RITHM

CONTRIBUTION

- Novel technique for the *Identifier Mapping*

LINSEN
ALGO
RITHM

CONTRIBUTION

- Novel technique for the *Identifier Mapping*
- Based on an **efficient** String Matching technique:

  Baeza-Yates&Perlberg Algorithm (BYP)

LINSEN
ALGO
RITHM

CONTRIBUTION

- Novel technique for the *Identifier Mapping*
- Based on an **efficient** String Matching technique:

  Baeza-Yates&Perlberg Algorithm (BYP)
- Applied on a *Graph-based model*

LINSEN
ALGO
RITHM

CONTRIBUTION

- Novel technique for the *Identifier Mapping*
- Based on an **efficient** String Matching technique:
  Baeza-Yates&Perlberg Algorithm (BYP)
- Applied on a *Graph-based model*
- Able to both *Split Identifiers* and *Expand* possible occurring abbreviations

THE AMBIGUITY PROBLEM

There could be multiple and equally correct splitting or expansion solutions

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

THE AMBIGUITY PROBLEM

There could be multiple and equally correct splitting or expansion solutions

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

- `r` *as for* Rectangle **OR** red

THE AMBIGUITY PROBLEM

There could be multiple and equally correct splitting or expansion solutions

```
/**
 * Draws the NullHandle. NullHandles are drawn as a
 * red framed rectangle box.
 */
public void draw(Graphics g) {
    Rectangle r = displayBox();

    g.setColor(Color.black);
    g.drawXORRect(r);
}
```

- r *as for* Rectangle **OR** red
- nsISupport ==> ns|IS|up|ports **OR**
  ==> ns|I|Supports

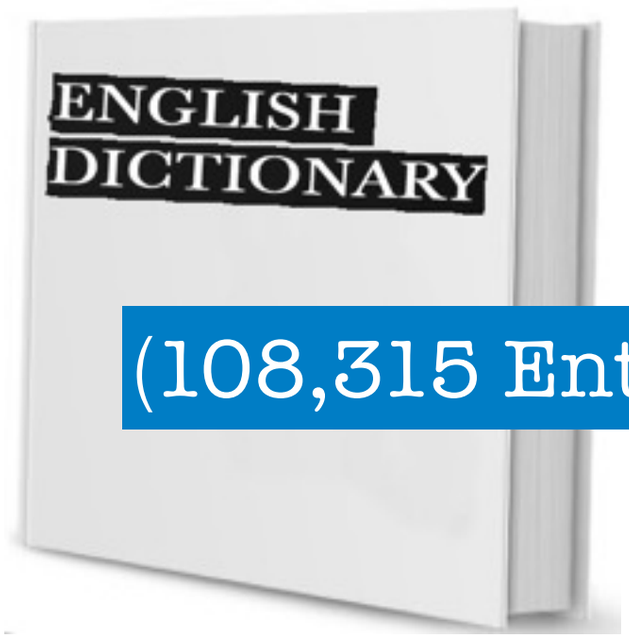
DICTIONARIES

DICTIONARIES

DICTIONARIES

Application-aware Dictionaries

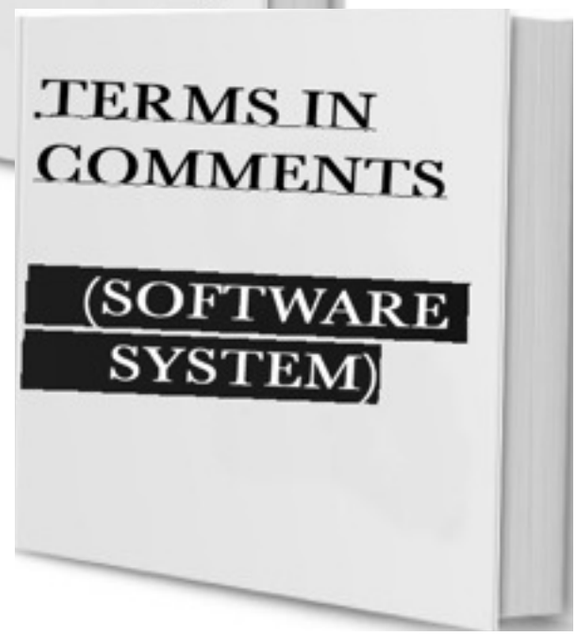
DICTIONARIES



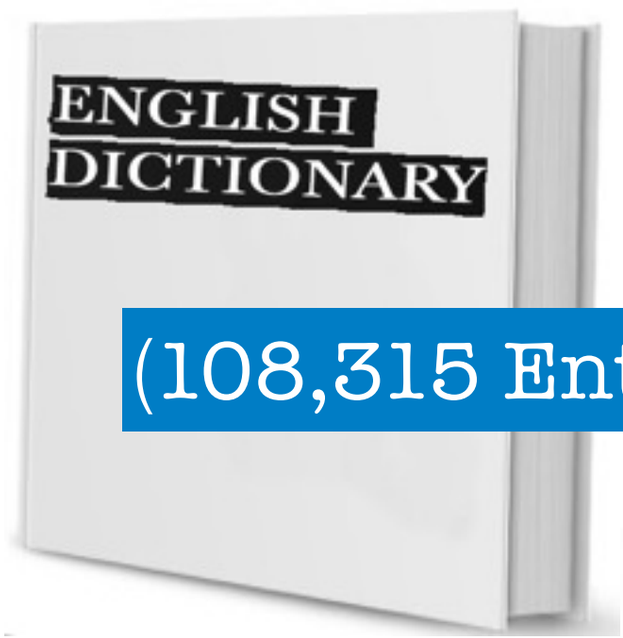
(108,315 Entries)

Application-aware

Dictionaries



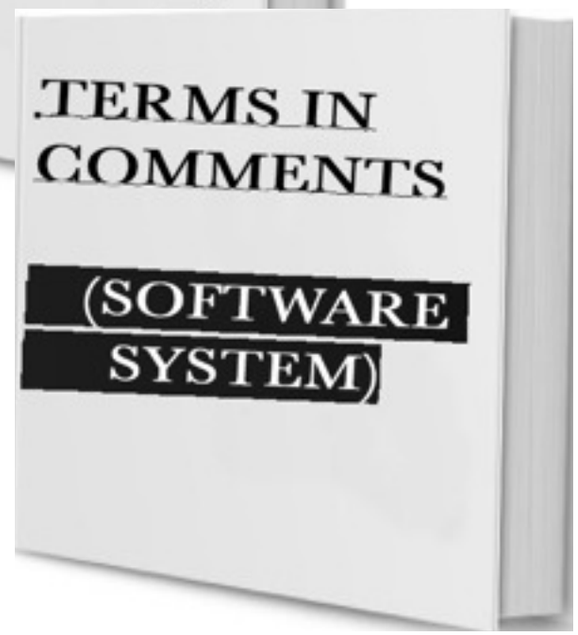
DICTIONARIES

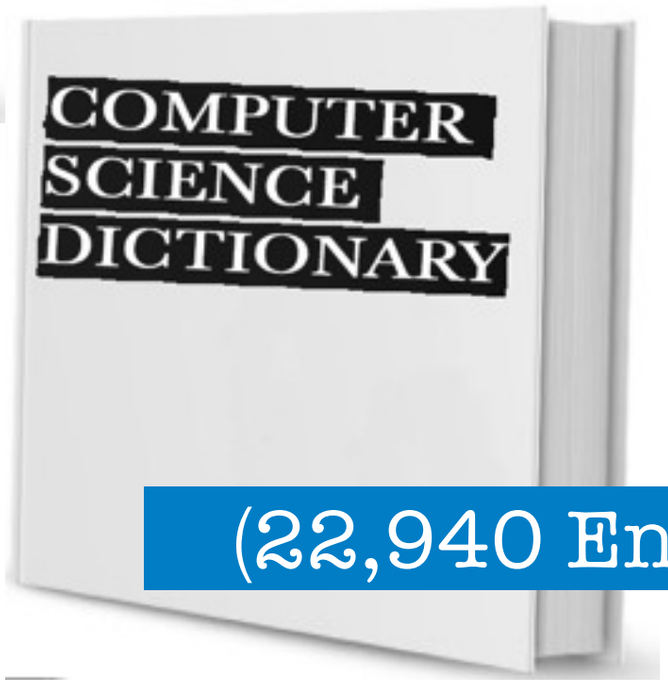


(108,315 Entries)

Application-aware

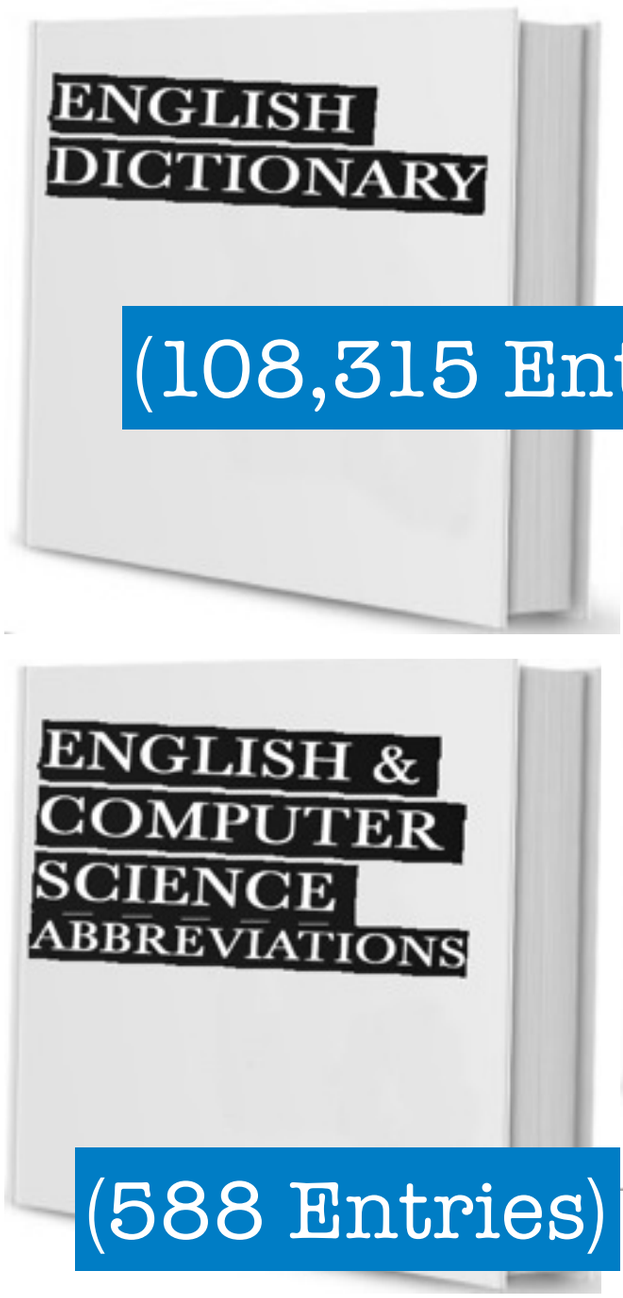
Dictionaries





(22,940 Entries)

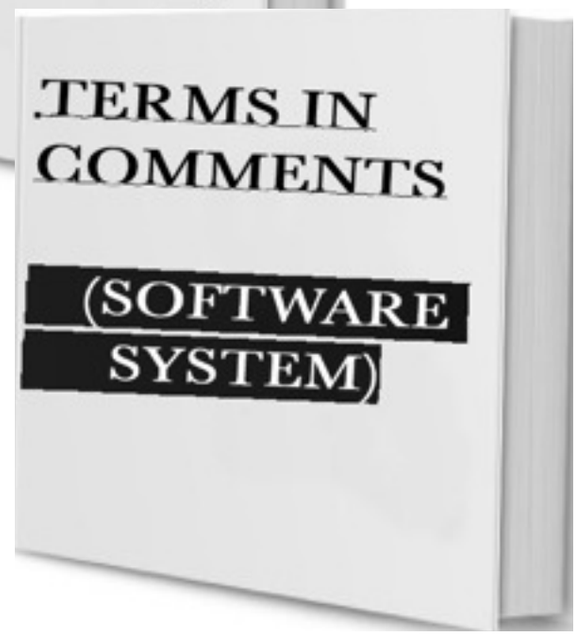
DICTIONARIES



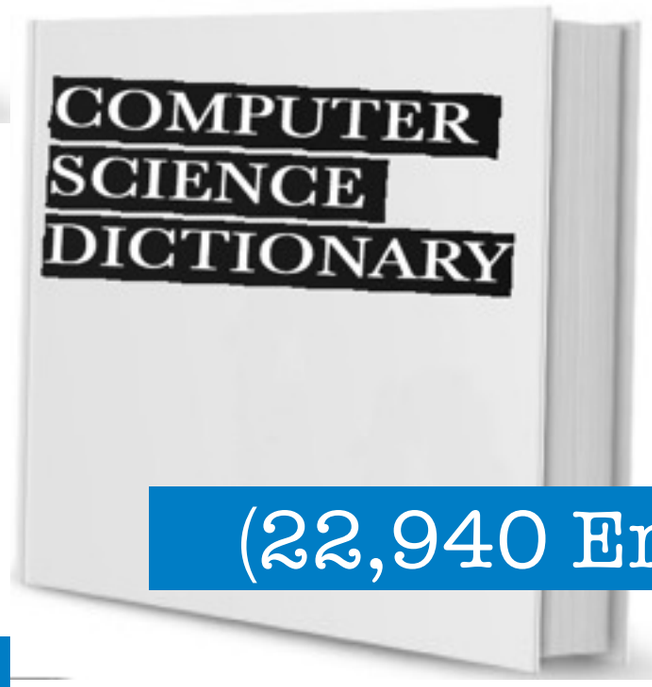
(108,315 Entries)

Application-aware

Dictionaries



(588 Entries)



(22,940 Entries)

Model: *Weighted* Directed Graph

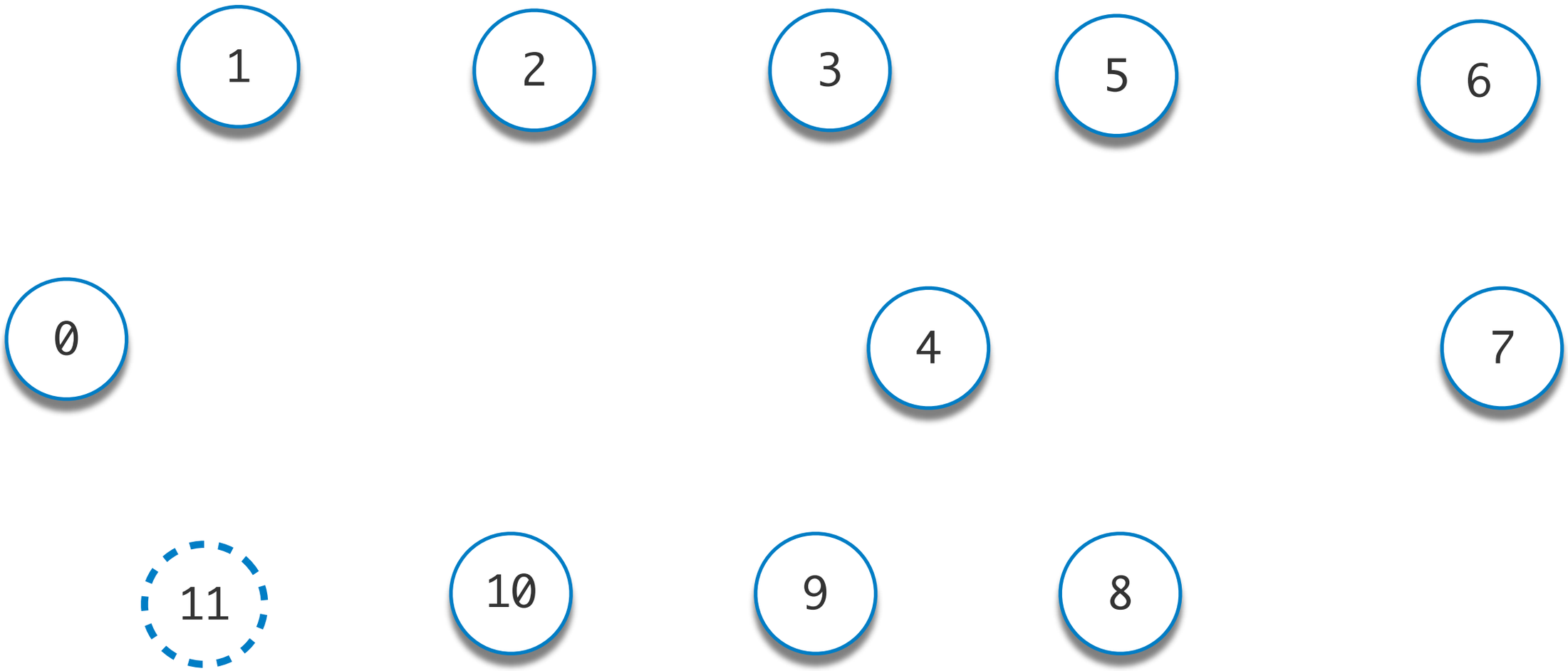
Example: `drawXORRect` identifier

GRAPH MODEL

GRAPH MODEL

Model: *Weighted* Directed Graph

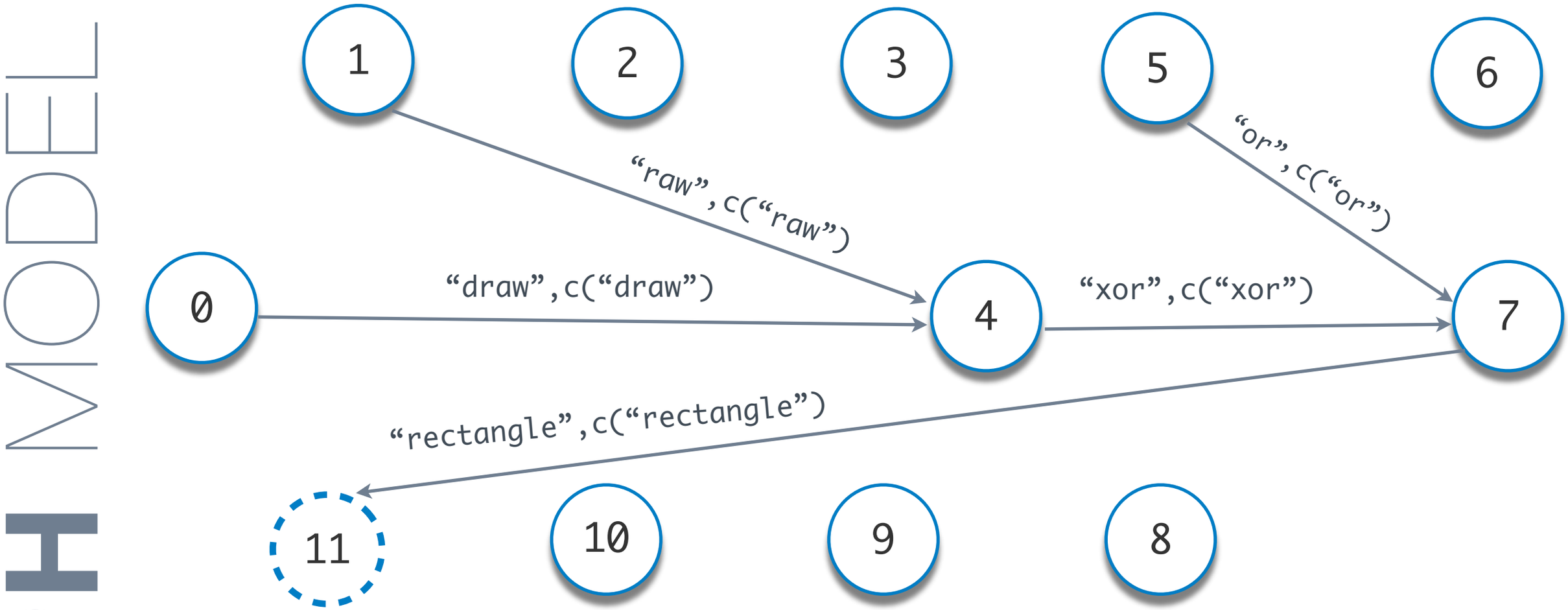
Example: `drawXORRect` identifier



- **NODES** correspond to characters of the current identifier

Model: *Weighted* Directed Graph

Example: `drawXORRect` identifier

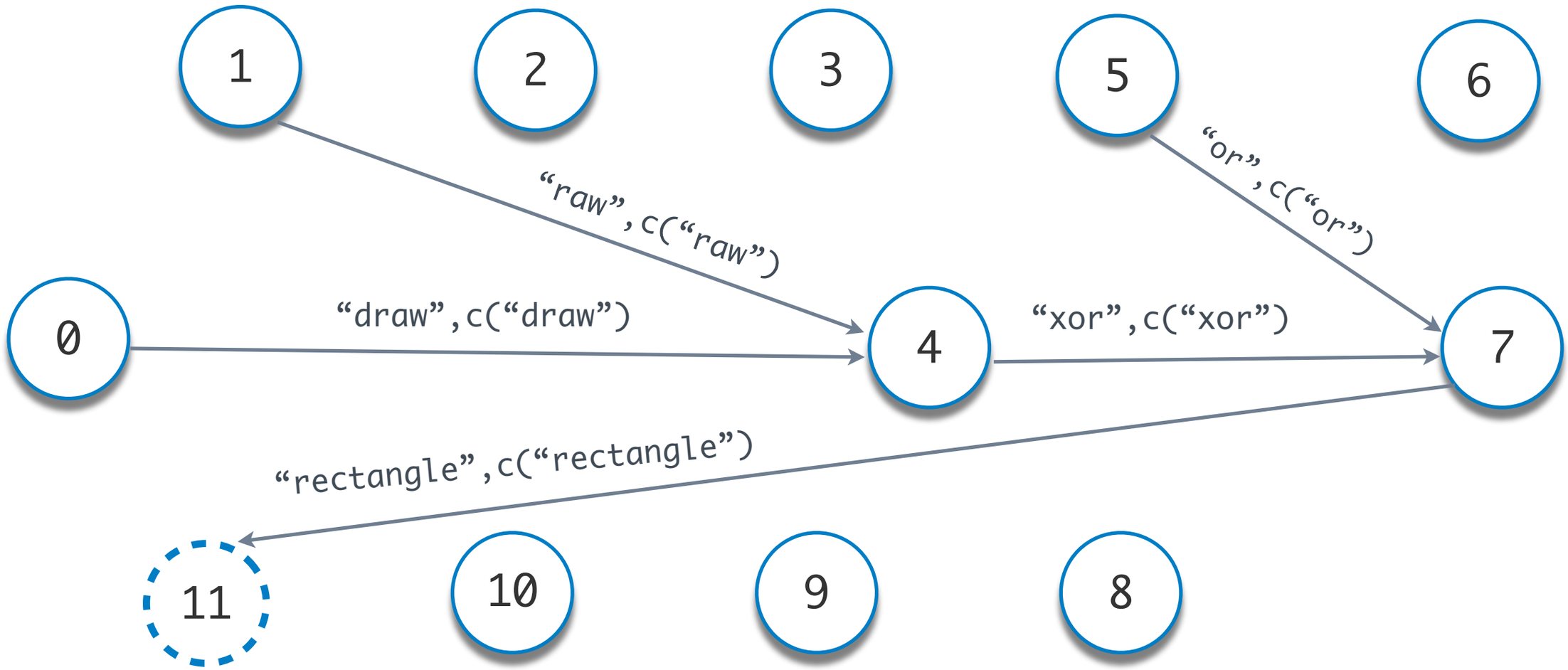


- **NODES** correspond to characters of the current identifier
- **ARCS** corresponds to matchings between *identifier substrings* and *dictionary words*

Model: *Weighted* Directed Graph

Example: `drawXORRect` identifier

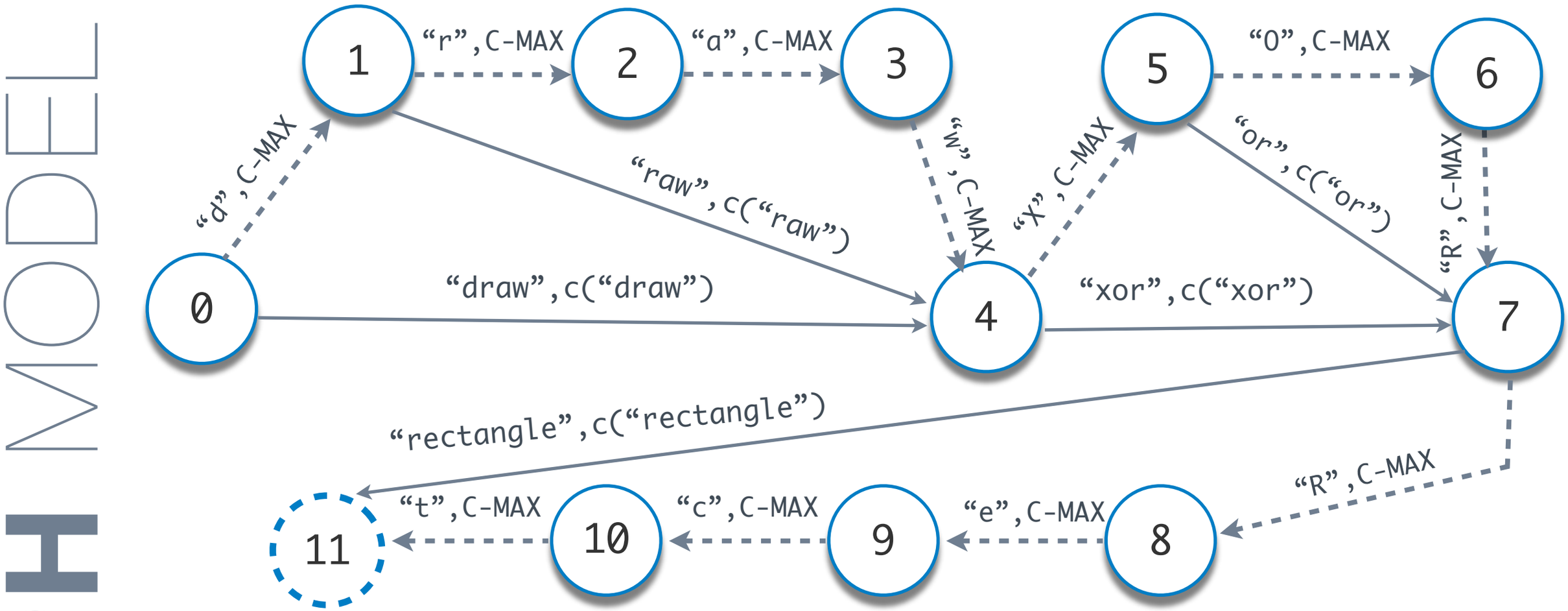
MODEL GRAPH



- **NODES** correspond to characters of the current identifier
- **ARCS** corresponds to matchings between *identifier substrings* and *dictionary words*
  - Application of the *String Matching Algorithm (BYP)*

Model: *Weighted* Directed Graph

Example: `drawXORRect` identifier

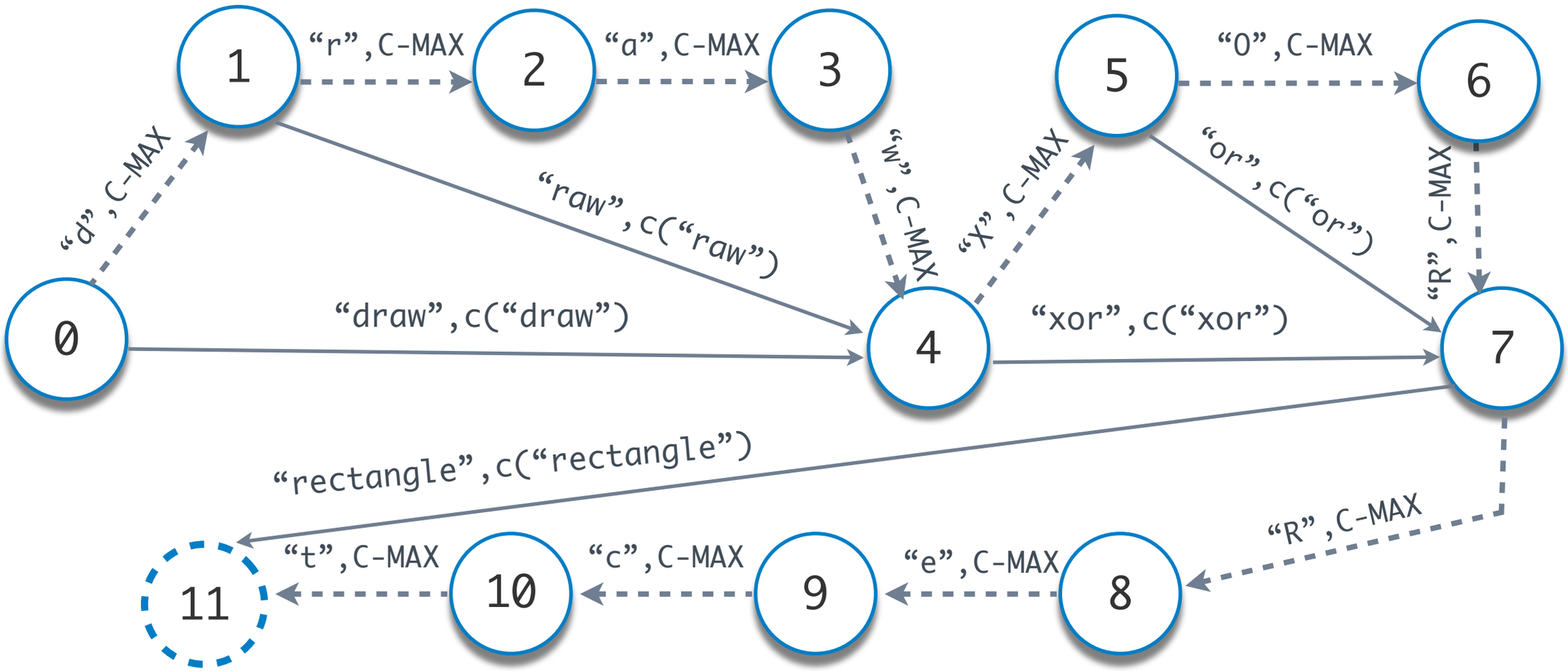


- **NODES** correspond to characters of the current identifier
- **ARCS** corresponds to matchings between *identifier substrings* and *dictionary words*
  - Application of the *String Matching Algorithm (BYP)*
  - *Padding Arcs* to ensure the Graph always connected

Model: *Weighted* Directed Graph

Example: `drawXORRect` identifier

MODEL GRAPH

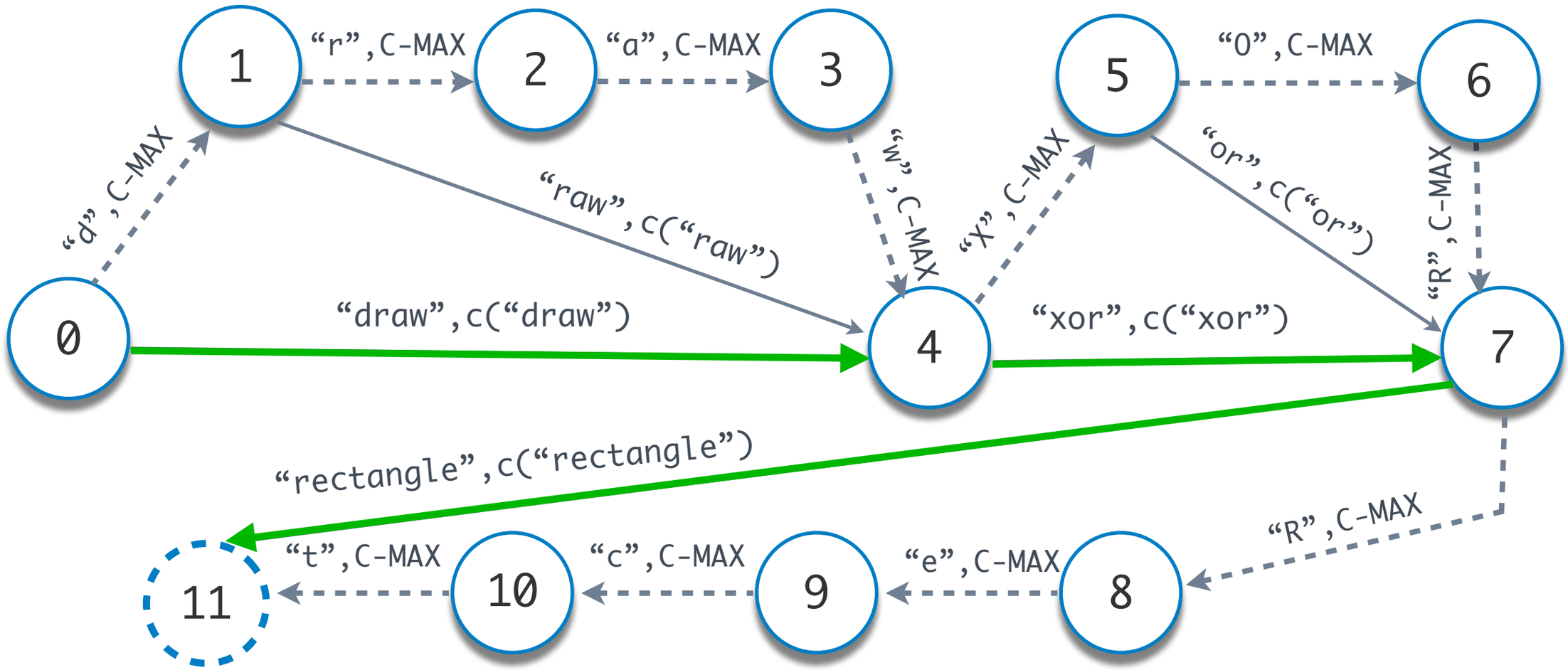


- Every Arc is **Labelled** with the corresponding dictionary word
- **Weights** represent the "cost" of each matching
- *Cost function* `[c("word")]` favors *longest words* and *words* coming from the *application-aware* dictionaries

GRAPH MODEL

Model: *Weighted* Directed Graph

Example: `drawXORRect` identifier



- The final **Mapping Solution** corresponds to the **sequence of labels** in the path with the **minimum cost** (*Djikstra Algorithm*)

STRING MATCHING

- Application of the *Baeza-Yates and Perlberg* (**BYP**) Algorithm
- **Signature**: `BYP(identifier, word, φ(word))`
  - `identifier`: target string
  - `word`: string to match
  - `φ(·)`: Tolerance (Error) function
    - *Bounds the length of acceptable matchings*

> ***Advantage**: Use the **same algorithm** for both the splitting and the expansion step* with different input *Tolerance function*

BYP FOR SPLITTING

- BYP(identifier, word, φ_{Split}(word))

  φ_{Split} : **Exact Matching** (i.e., No Errors allowed)

Identifier:
drawXORRect

D_{File}
..
draw,
the,
are,
null,
handle,
box,
red,
rectangle,
..

$D_{COMPUTER\text{-}SCIENCE}$
...
echo,
testing,
threading,
xpm,
xor,
....

$D_{English}$
...
abort
absolute
abstract
...
or
...
raw
...

BYP FOR SPLITTING

- BYP(identifier, word, φ_{Split}(word))

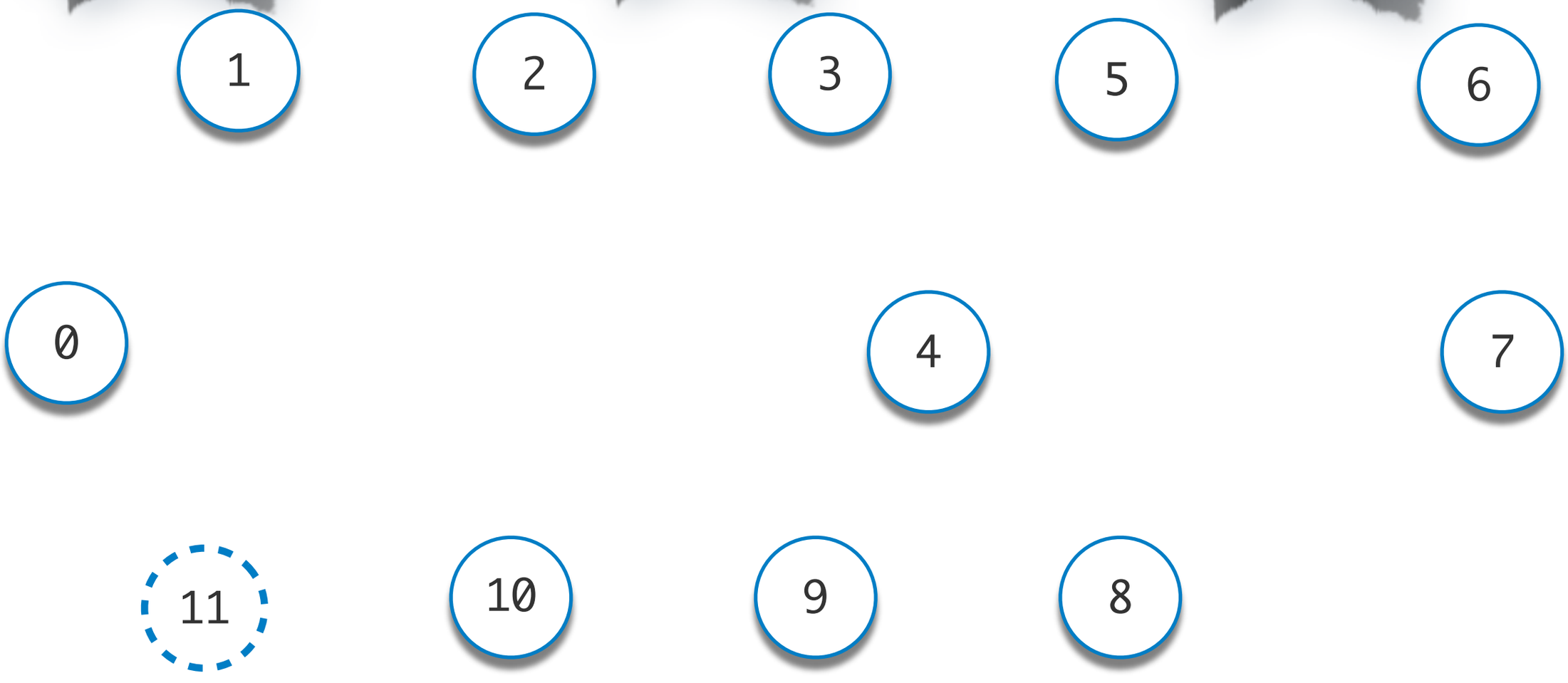
  φ_{Split} : **Exact Matching** (i.e., No Errors allowed)

Identifier:
drawXORRect

D_{File}
..
draw,
the,
are,
null,
handle,
box,
red,
rectangle,
..

$D_{COMPUTER-SCIENCE}$
...
echo,
testing,
threading,
xpm,
xor,
....

$D_{English}$
...
abort
absolute
abstract
...
or
...
raw
...



BYP FOR SPLITTING

- BYP(identifier, word, φ_{Split}(word))

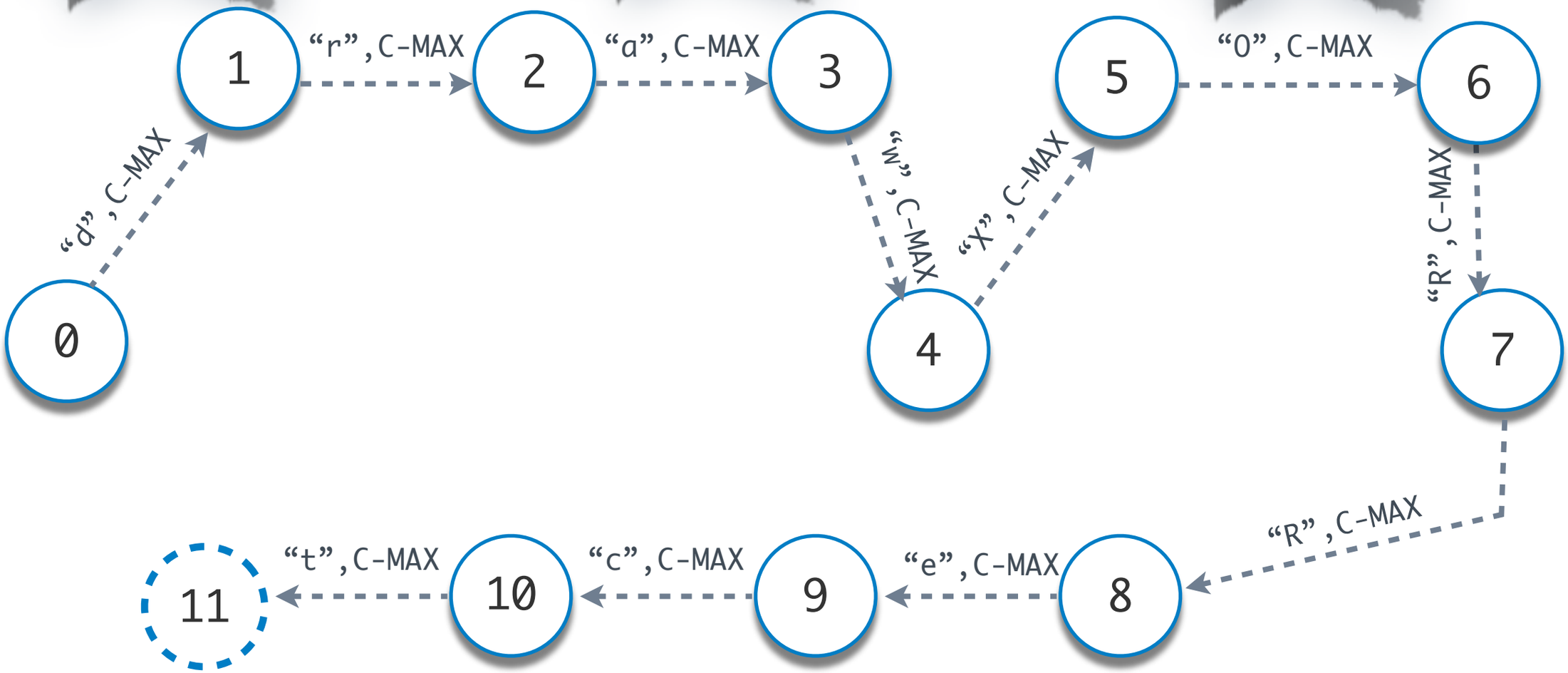
  φ_{Split} : **Exact Matching** (i.e., No Errors allowed)

Identifier:
drawXORRect

D_{File}
..
draw,
the,
are,
null,
handle,
box,
red,
rectangle,
..

$D_{COMPUTER-SCIENCE}$
...
echo,
testing,
threading,
xpm,
xor,
....

$D_{English}$
...
abort
absolute
abstract
...
or
...
raw
...



BYP FOR SPLITTING

- BYP(identifier, word, φ_{Split}(word))

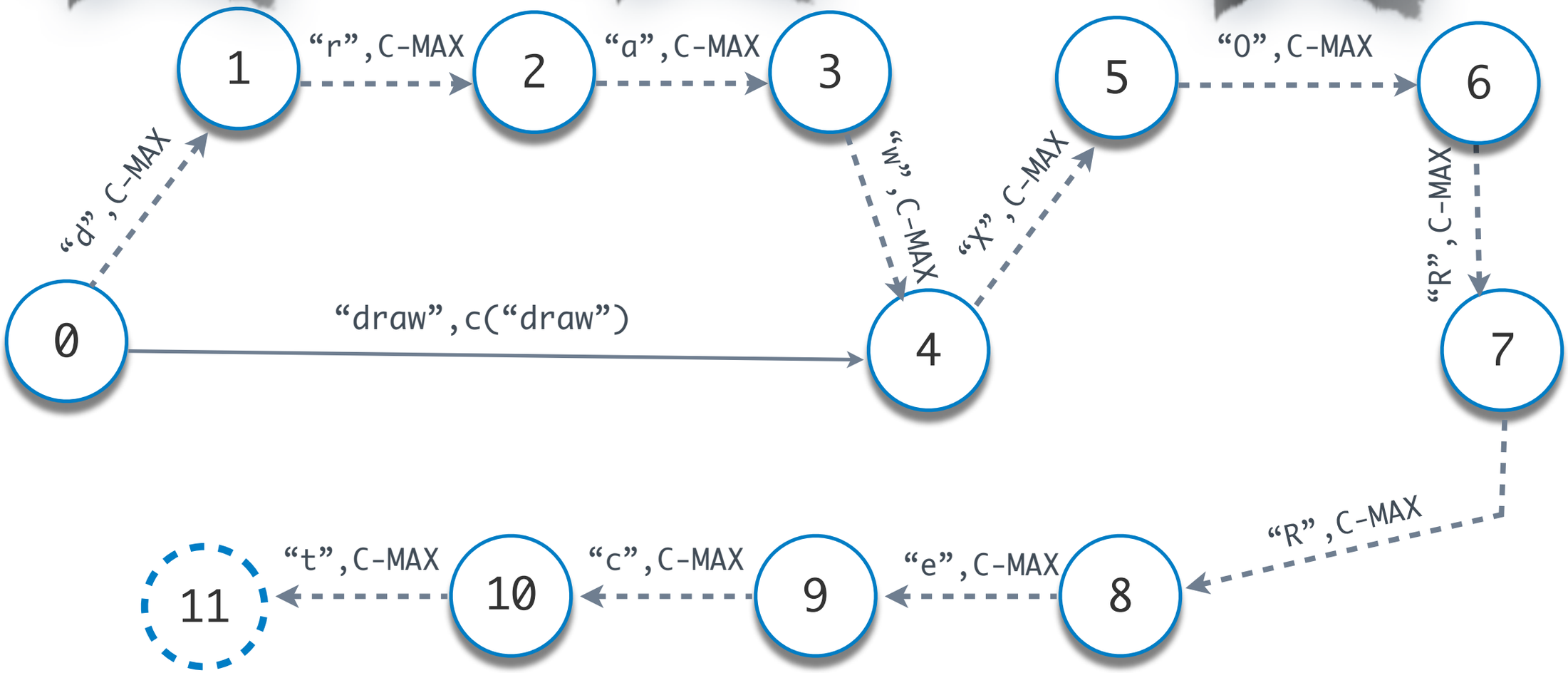
  φ_{Split} : **Exact Matching** (i.e., No Errors allowed)

Identifier:
drawXORRect

D_{File}
..
draw,
the,
are,
null,
handle,
box,
red,
rectangle,
..

$D_{COMPUTER-SCIENCE}$
...
echo,
testing,
threading,
xpm,
xor,
....

$D_{English}$
...
abort
absolute
abstract
...
or
...
raw
...



BYP FOR SPLITTING

- BYP(identifier, word, φ_{Split}(word))

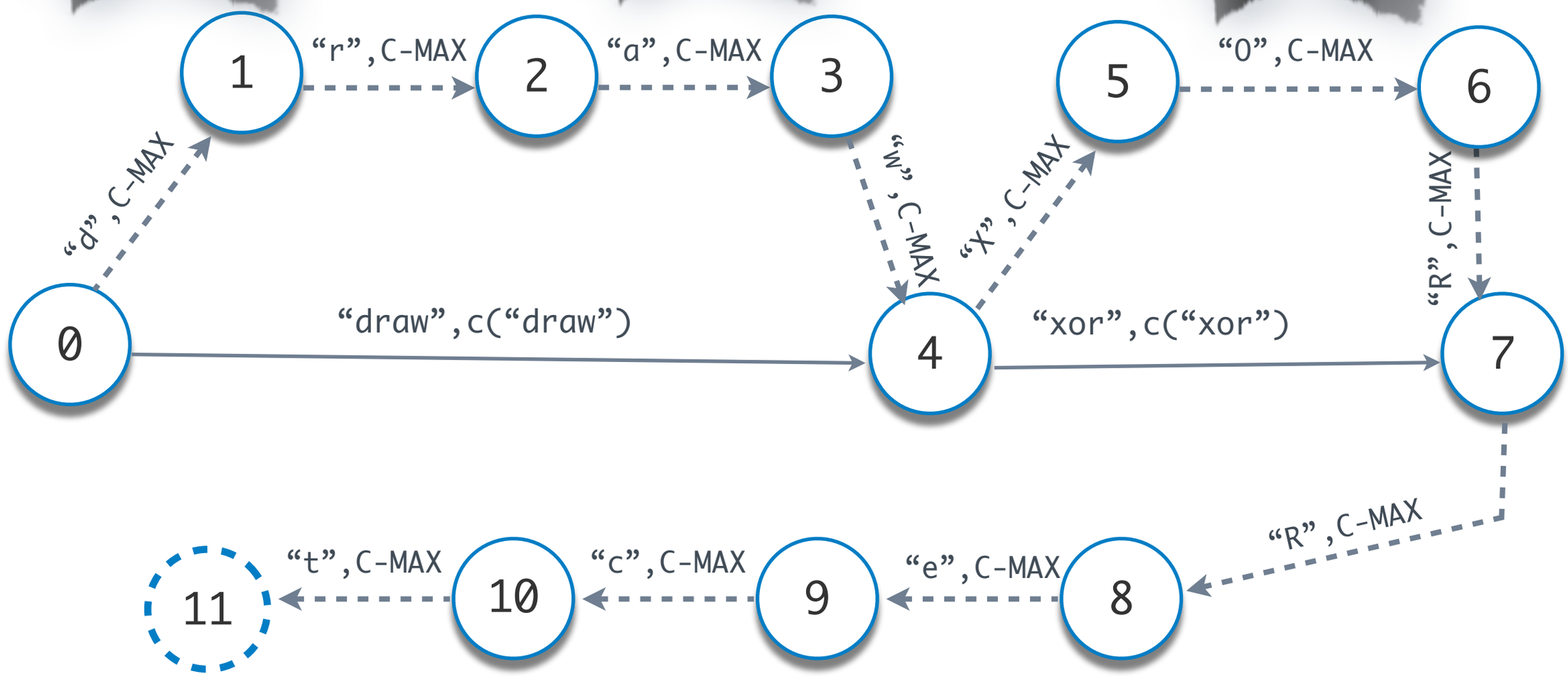
  φ_{Split} : **Exact Matching** (i.e., No Errors allowed)

Identifier:
drawXORRect

D_{File}
..
draw,
the,
are,
null,
handle,
box,
red,
rectangle,
..

$D_{COMPUTER\text{-}SCIENCE}$
...
echo,
testing,
threading,
xpm,
xor,
...

$D_{English}$
...
abort
absolute
abstract
...
or
...
raw
...



BYP FOR SPLITTING

- BYP(identifier, word, φ_{Split}(word))

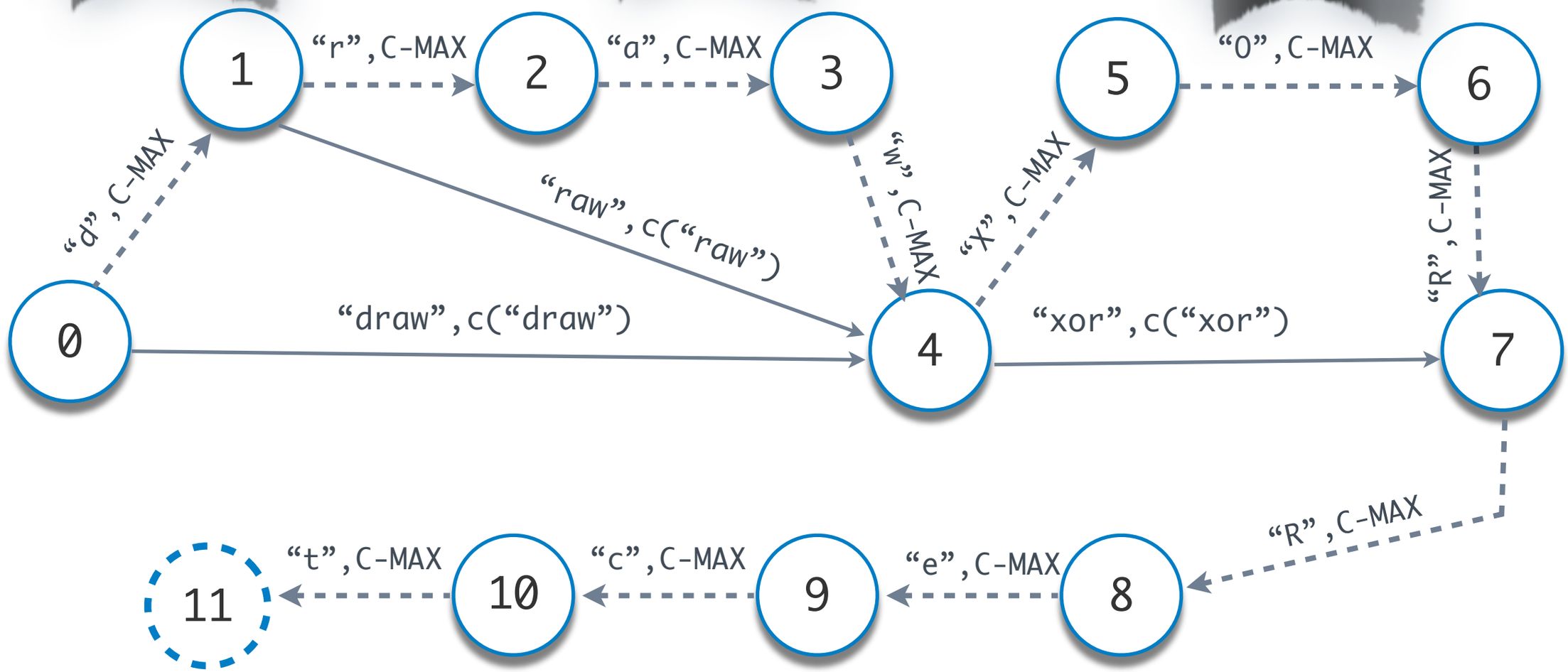
  φ_{Split} : **Exact Matching** (i.e., No Errors allowed)

Identifier:
drawXORRect

D_{File}
...
draw,
the,
are,
null,
handle,
box,
red,
rectangle,
...

$D_{COMPUTER\text{-}SCIENCE}$
...
echo,
testing,
threading,
xpm,
xor,
...

$D_{English}$
...
abort
absolute
abstract
...
or
...
raw
...



BYP FOR SPLITTING

- BYP(identifier, word, φ_{Split}(word))

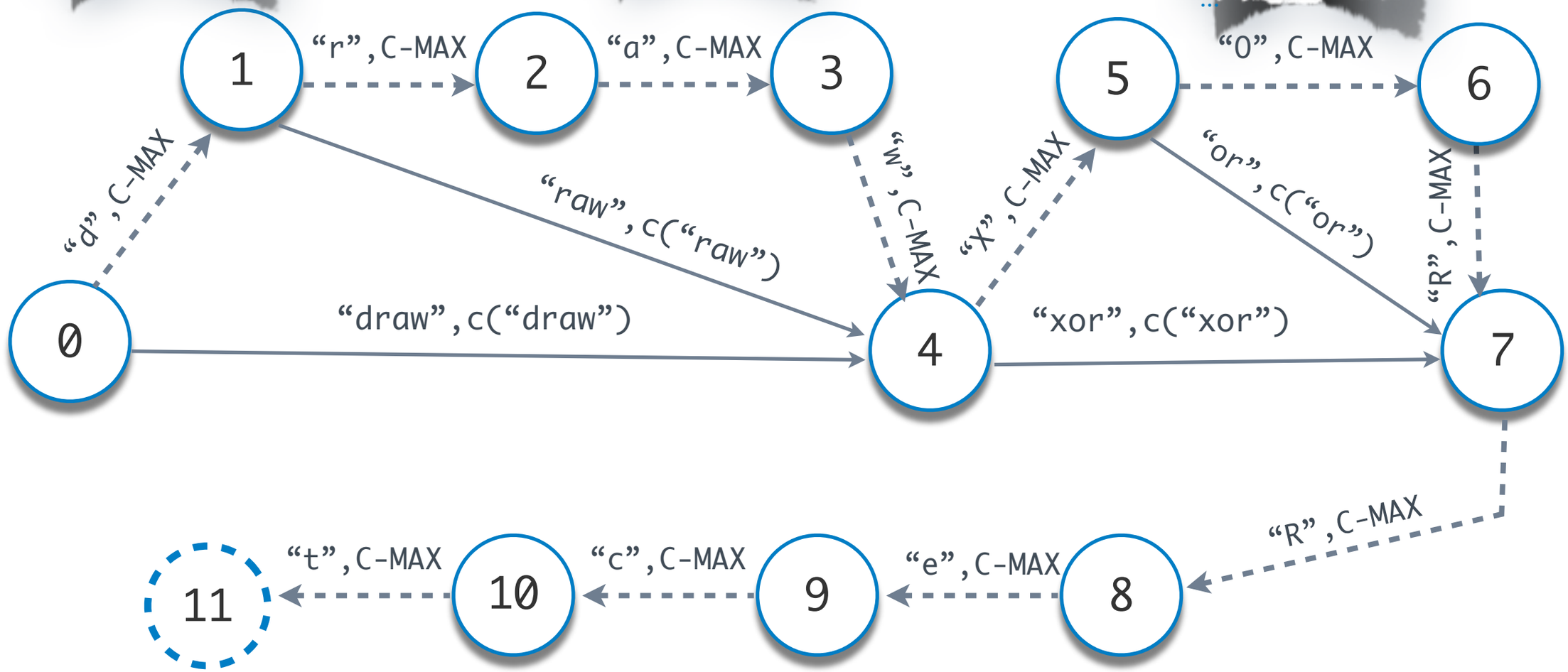
  φ_{Split} : **Exact Matching** (i.e., No Errors allowed)

Identifier:
drawXORRect

D_{File}
..
draw,
the,
are,
null,
handle,
box,
red,
rectangle,
..

$D_{COMPUTER-SCIENCE}$
...
echo,
testing,
threading,
xpm,
xor,
....

$D_{English}$
...
abort
absolute
abstract
...
or
...
raw
..



BYP FOR SPLITTING

- BYP(identifier, word, φ_{Split}(word))

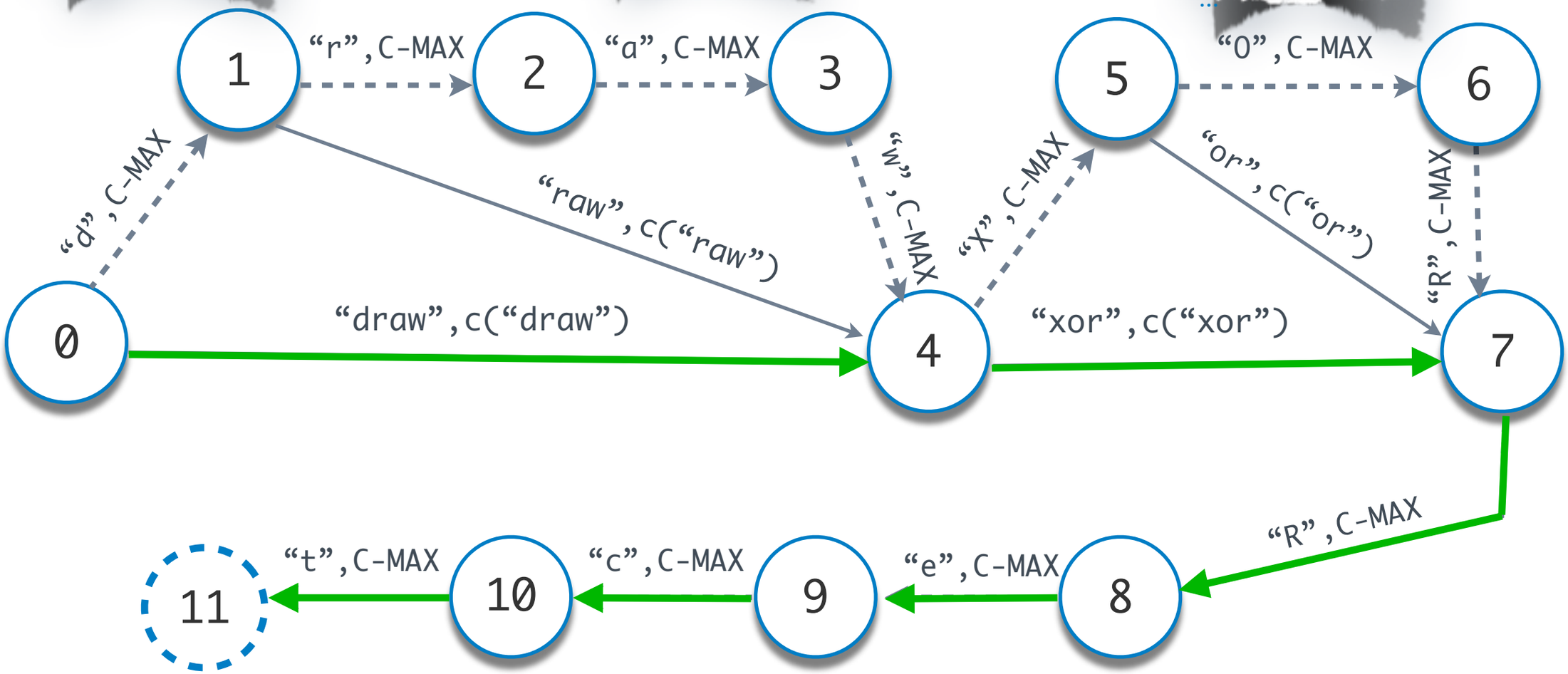
  φ_{Split} : **Exact Matching** (i.e., No Errors allowed)

Identifier:
drawXORRect

D_{File}
..
draw,
the,
are,
null,
handle,
box,
red,
rectangle,
..

$D_{COMPUTER-SCIENCE}$
...
echo,
testing,
threading,
xpm,
xor,
....

$D_{English}$
...
abort
absolute
abstract
...
or
...
raw
..



BYP FOR EXPANSION

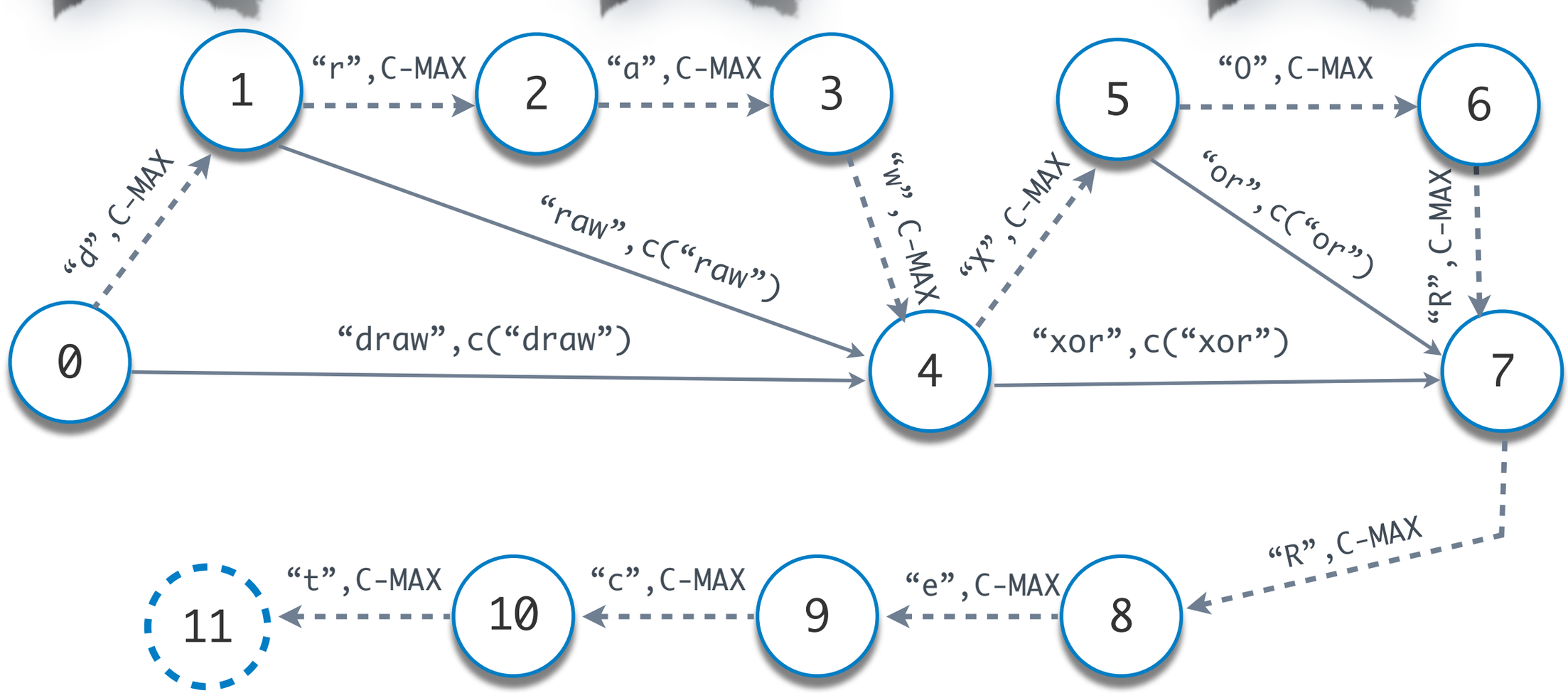
- BYP(identifier, word, φ_{Exp}(word))
- φ_{Exp} : *Approximate Matching*

Identifier:
`drawXORRect`

D_{File}
..
draw,
the,
are,
null,
handle,
box,
red,
rectangle,
..

$D_{COMPUTER\text{-}SCIENCE}$
...
echo,
testing,
threading,
xpm,
xor,
....

$D_{English}$
...
abort
absolute
abstract
...
or
...
raw
...



BYP FOR EXPANSION

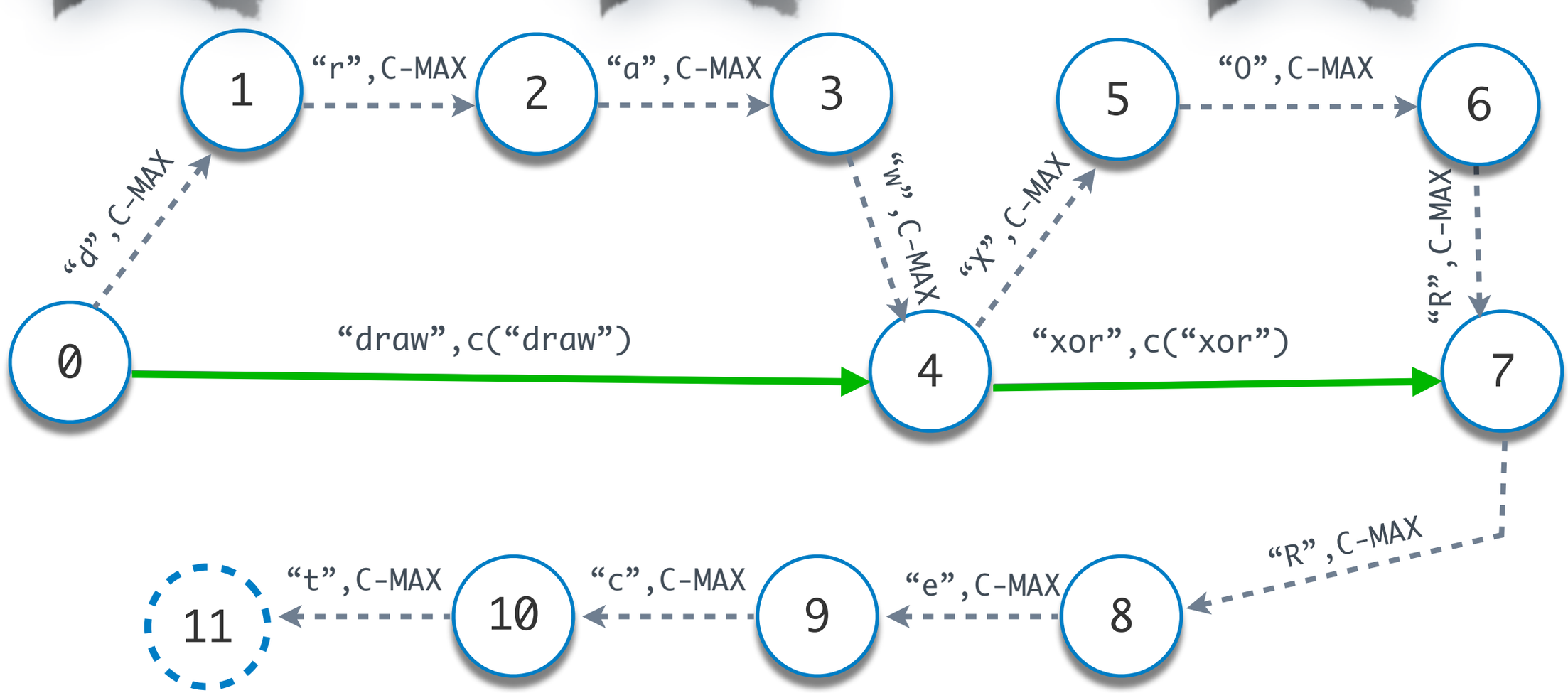
- BYP(identifier, word, φ_{Exp}(word))
- φ_{Exp} : *Approximate Matching*

Identifier:
`drawXORRect`

D_{File}
..
draw,
the,
are,
null,
handle,
box,
red,
rectangle,
..

$D_{COMPUTER\text{-}SCIENCE}$
...
echo,
testing,
threading,
xpm,
xor,
....

$D_{English}$
...
abort
absolute
abstract
...
or
...
raw
...



BYP FOR EXPANSION

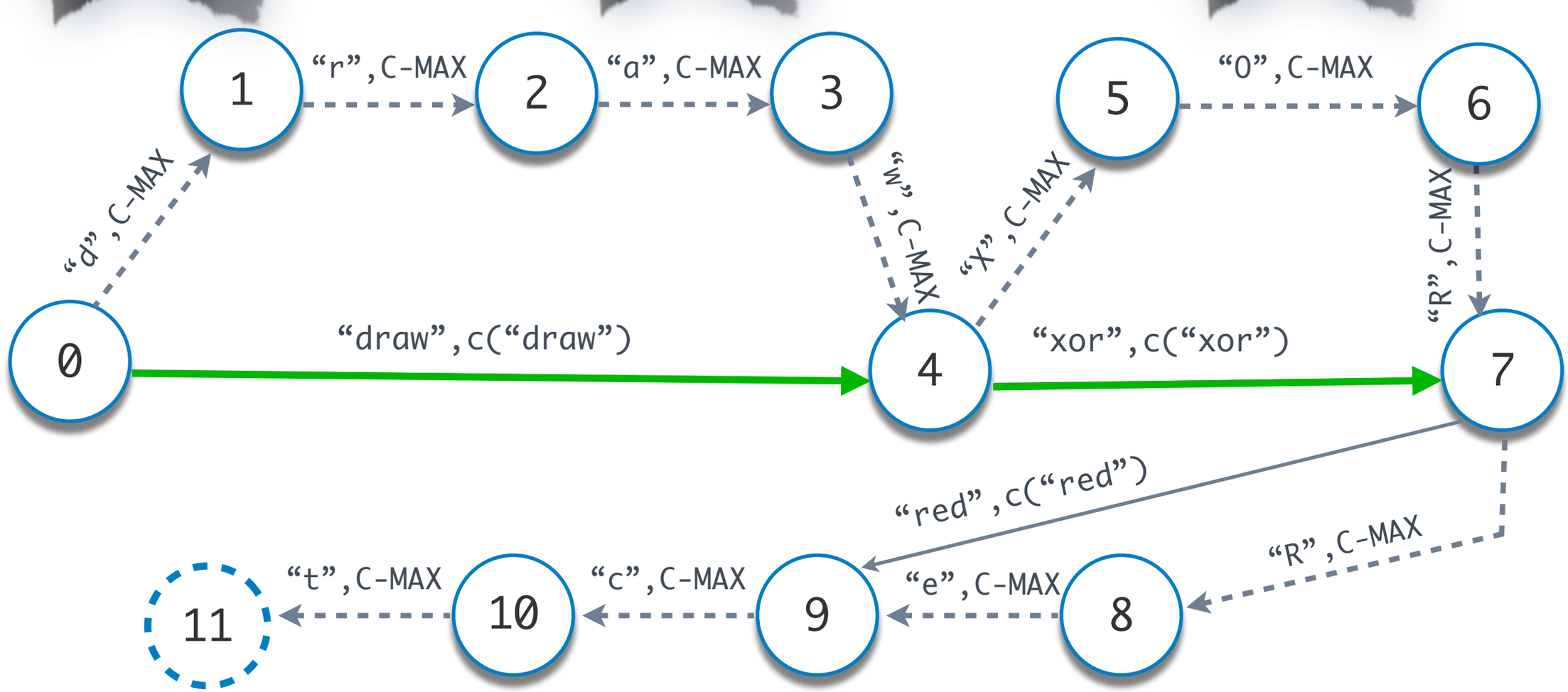
- BYP(identifier, word, φ_{Exp}(word))
- φ_{Exp} : *Approximate Matching*

Identifier:
`drawXORRect`

D_{File}
...
draw,
the,
are,
null,
handle,
box,
red,
rectangle,
...

$D_{COMPUTER\text{-}SCIENCE}$
...
echo,
testing,
threading,
xpm,
xor,
...

$D_{English}$
...
abort
absolute
abstract
...
or
...
raw
...



BYP FOR EXPANSION

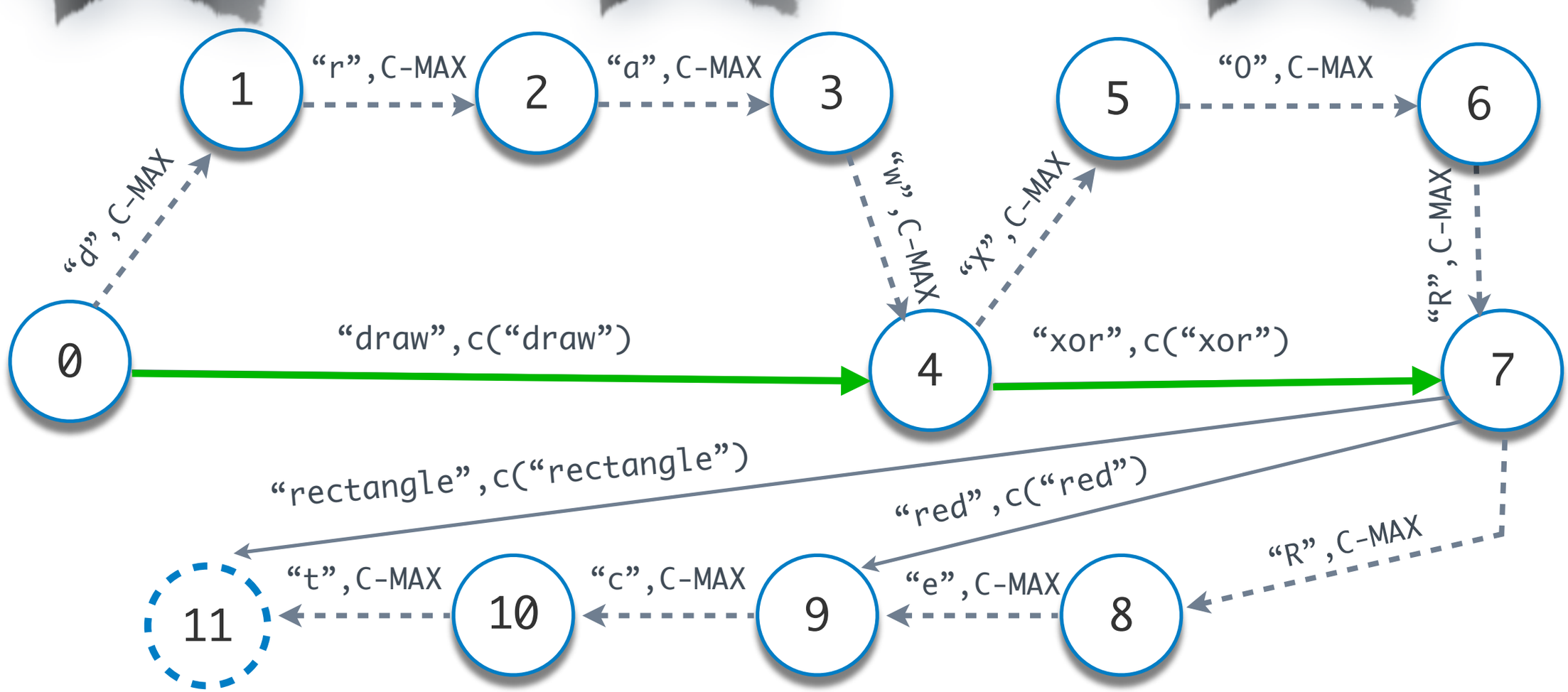
- BYP(identifier, word, φ_{Exp}(word))
- φ_{Exp} : *Approximate Matching*

Identifier:
`drawXORRect`

D_{File}
..
draw,
the,
are,
null,
handle,
box,
red,
rectangle,
...

$D_{COMPUTER\text{-}SCIENCE}$
...
echo,
testing,
threading,
xpm,
xor,
....

$D_{English}$
...
abort
absolute
abstract
...
or
...
raw
...



BYP FOR EXPANSION

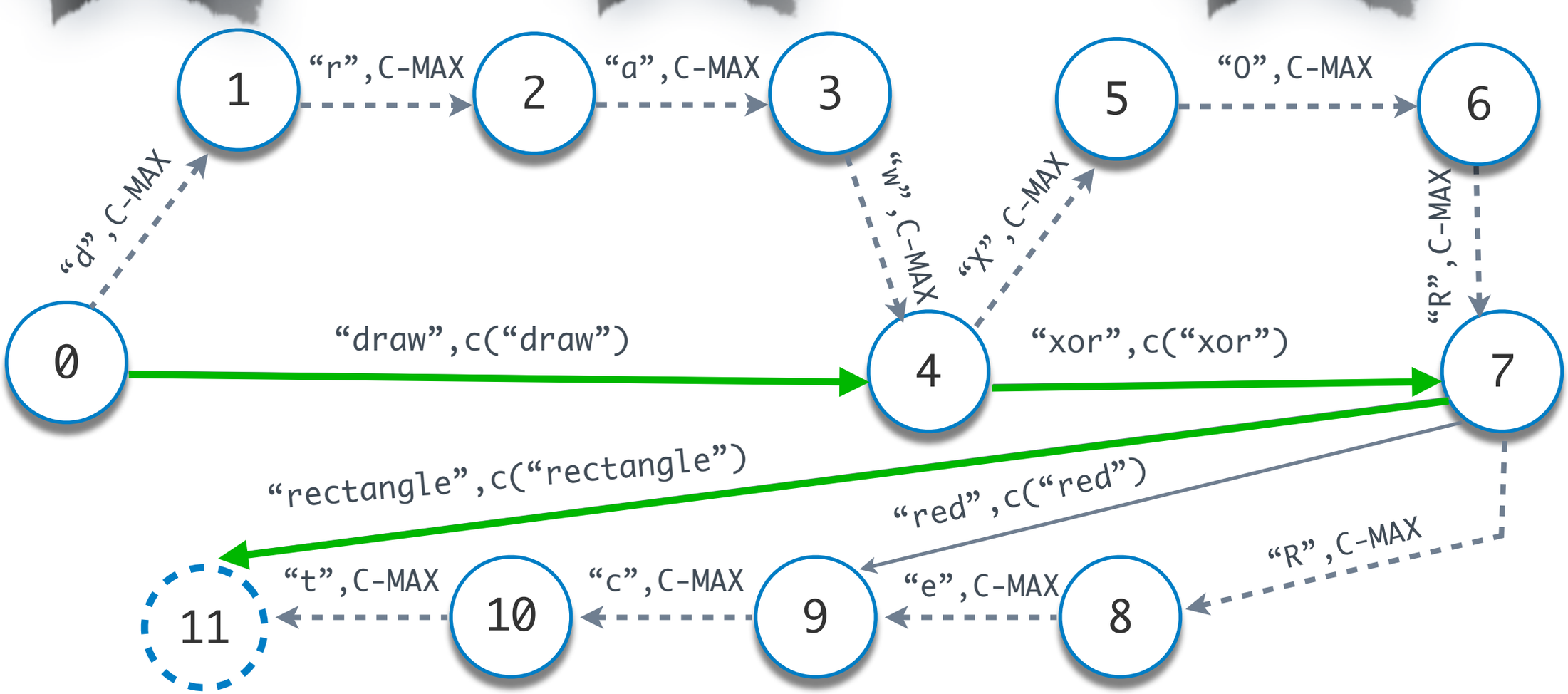
- BYP(identifier, word, φ_{Exp}(word))
- φ_{Exp} : *Approximate Matching*

Identifier:
`drawXORRect`

D_{File}
..
draw,
the,
are,
null,
handle,
box,
red,
rectangle,
...

$D_{COMPUTER\text{-}SCIENCE}$
...
echo,
testing,
threading,
xpm,
xor,
....

$D_{English}$
...
abort
absolute
abstract
...
or
...
raw
...



EMPIRICAL EVALUATION

RESEARCH QUESTIONS

- **RQ1**:

  How does LINSEN compare with state-of-the-art approaches as for the splitting of identifiers?

EMPIRICAL EVALUATION

RESEARCH QUESTIONS

- **RQ1**:

  How does LINSEN compare with state-of-the-art approaches as for the splitting of identifiers?

- **RQ2**:

  How does LINSEN compare with state-of-the-art approaches as for the mapping of identifiers to dictionary words?

EMPIRICAL EVALUATION

RESEARCH QUESTIONS

- **RQ1**:

  How does LINSEN compare with state-of-the-art approaches as for the splitting of identifiers?

- **RQ2**:

  How does LINSEN compare with state-of-the-art approaches as for the mapping of identifiers to dictionary words?

- **RQ3**:

  What is the ability of the LINSEN approach in dealing with different types of abbreviations?

EMPIRICAL EVALUATION

CASE STUDIES

DTW (Madani et. al 2010)

GenTest+Normalize (Lawrie and Binkley, 2011)

LUDISO Dataset

(2012)

15 out of 750 software systems

Covering the **58%** *of total identifiers*

AMAP (Hill and Pollock, 2008)

System	Version	Ids in Oracle	KLOC	
JHotDraw	5.1	957	16	**RQ1** and **RQ2**
Lynx	2.8.5	3,085	174	
a2ps	4.14	211	6	
which	2.20	487	174	
Mozilla-source	1.0	573	4,595	
MySQL	5.0.17	194	2,028	
Cinelerra	2.0	191	3,533	
eMule	0.46	92	262	
Quake3	1.32b	80	705	
Gcc	2.95	70	1,289	
Ghostscript	7.07	66	437	
Samba	3.0.0	49	662	**RQ1** only
Asterisk	1.21	44	459	
Minux	2.0	31	334	
Mozilla-source	1.3	29	11,458	
Mozilla-source	1.2	28	4,681	
Mozilla-source	1.4	27	4,710	
Mozilla-source	1.1	24	4,676	
Httpd	2.0.48	22	558	
Azureus	3.0	551	2,682	
iText.Net	1.4.1	751	3,380	
Liferay Portal	4.3.2	651	3,949	**RQ3** only
OOPortable	2.2.1	699	3,442	
Tiger Envelopes	0.8.9	577	2,647	

EMPIRICAL EVALUATION

EVALUATION METRICS

Comparability of Results:

Accuracy rate

Qualitative Evaluation:

Precision/Recall/F-1

[Guerrouj, et.al , 2011]

EMPIRICAL EVALUATION

EVALUATION METRICS

Comparability of Results:

Accuracy rate

Qualitative Evaluation:

Precision/Recall/F-1

[Guerrouj, et.al , 2011]

EMPIRICAL EVALUATION

EVALUATION METRICS

Comparability of Results:

Accuracy rate

Qualitative Evaluation:

Precision/Recall/F-1

[Guerrouj, et.al , 2011]

- **Identifier Level evaluation**:

  *Each mapping result **must** be completely correct*

- **Soft-word Level evaluation**:

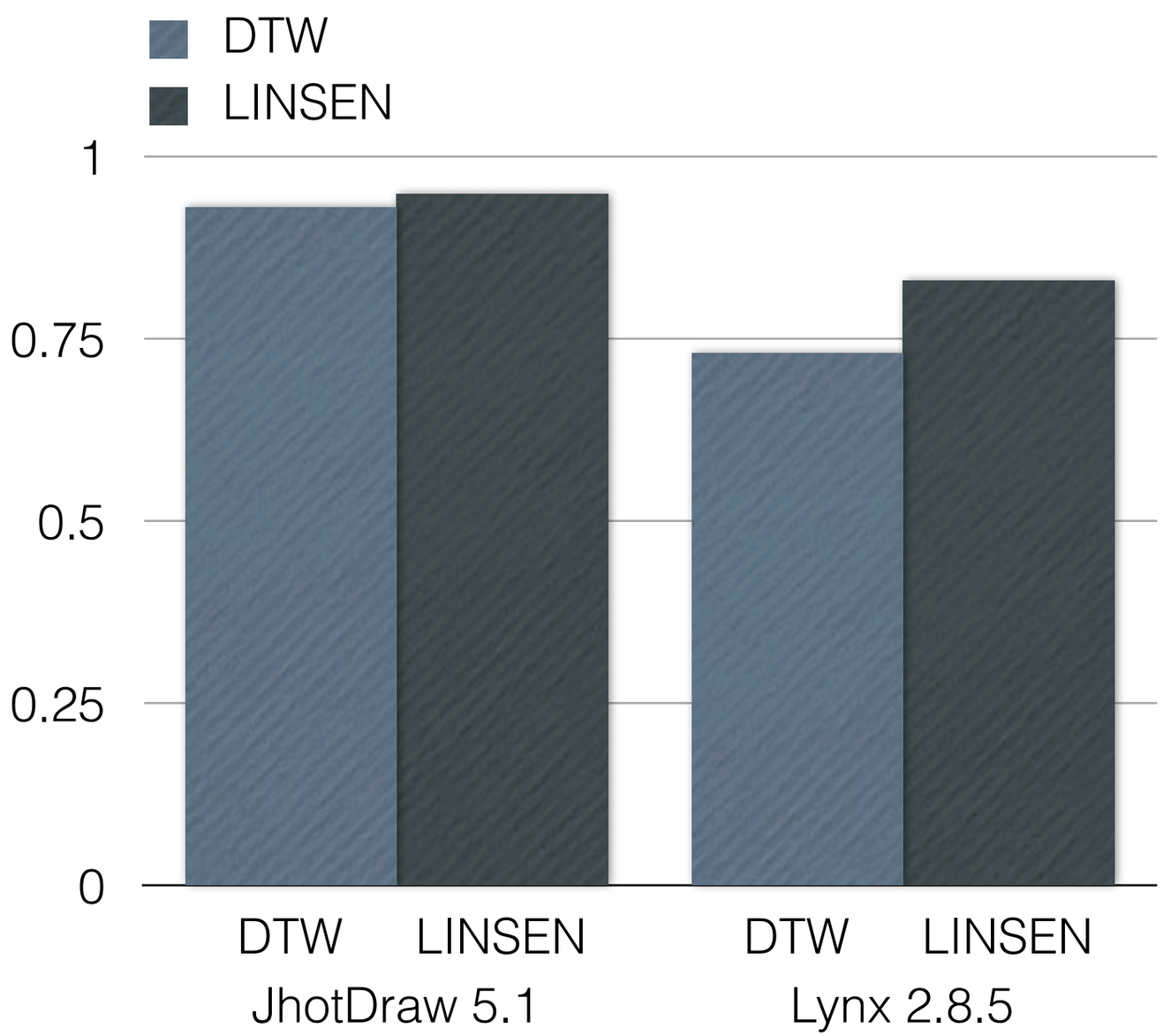
  “Partial credit” given to each word correctly mapped

As for the comparison with **GenTest+Normalize** (Lawrie and Binkley, 2011)

RE
SULTS
R Q 1

RQ1: SPLITTING

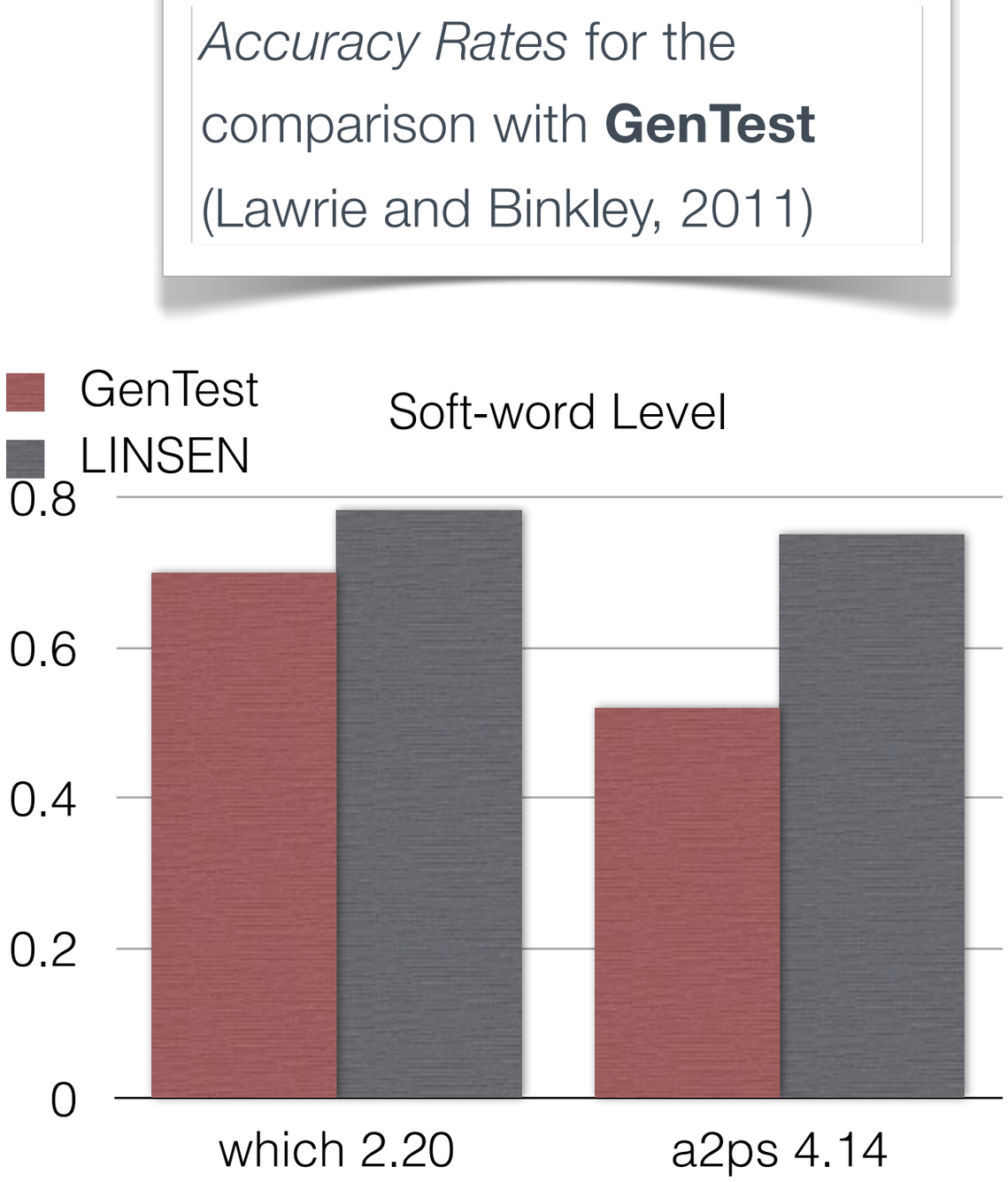
Accuracy Rates for the comparison with **DTW** (Madani et. al 2010)

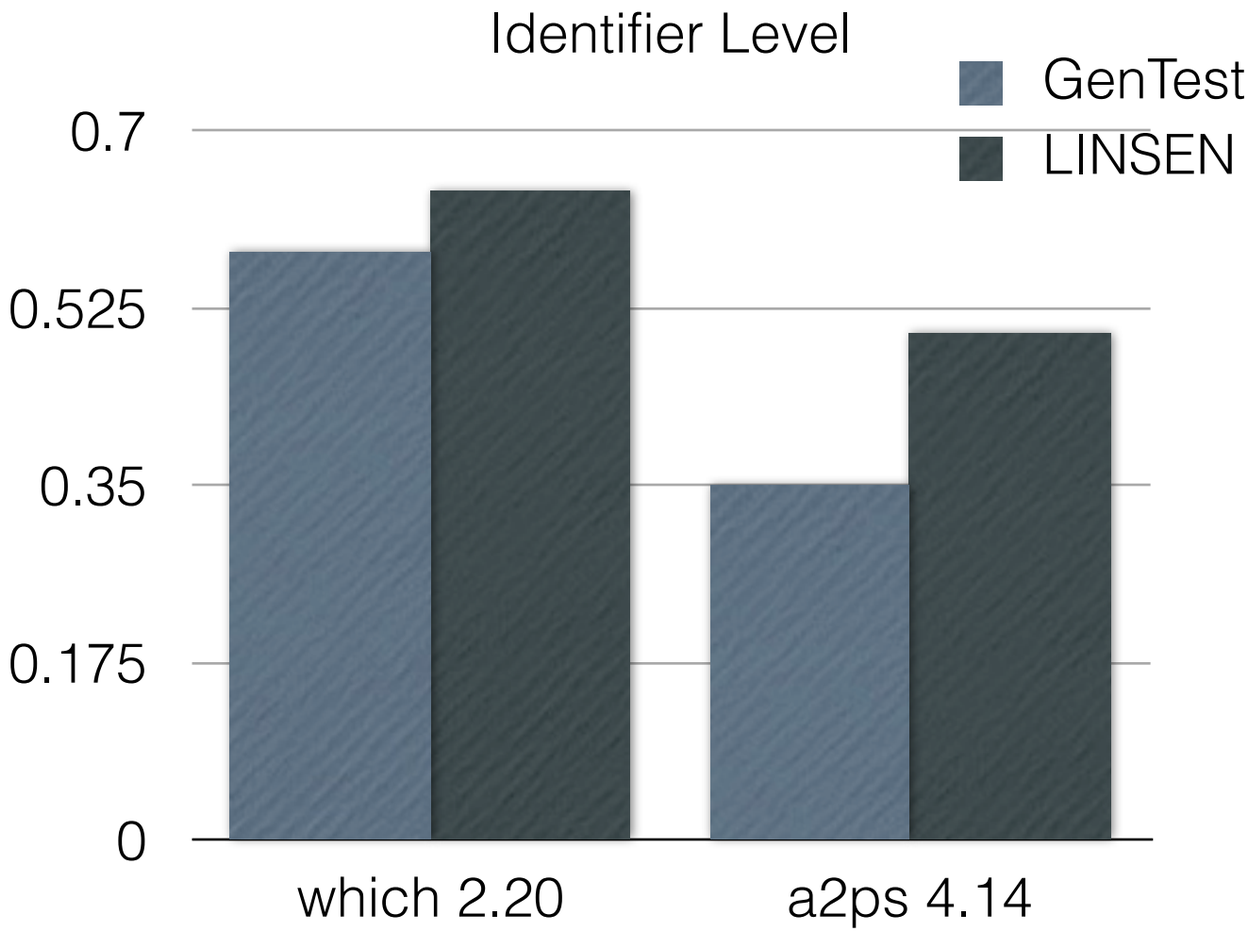


RESULTS
RQ1

RQ1: SPLITTING

Accuracy Rates for the comparison with **GenTest** (Lawrie and Binkley, 2011)

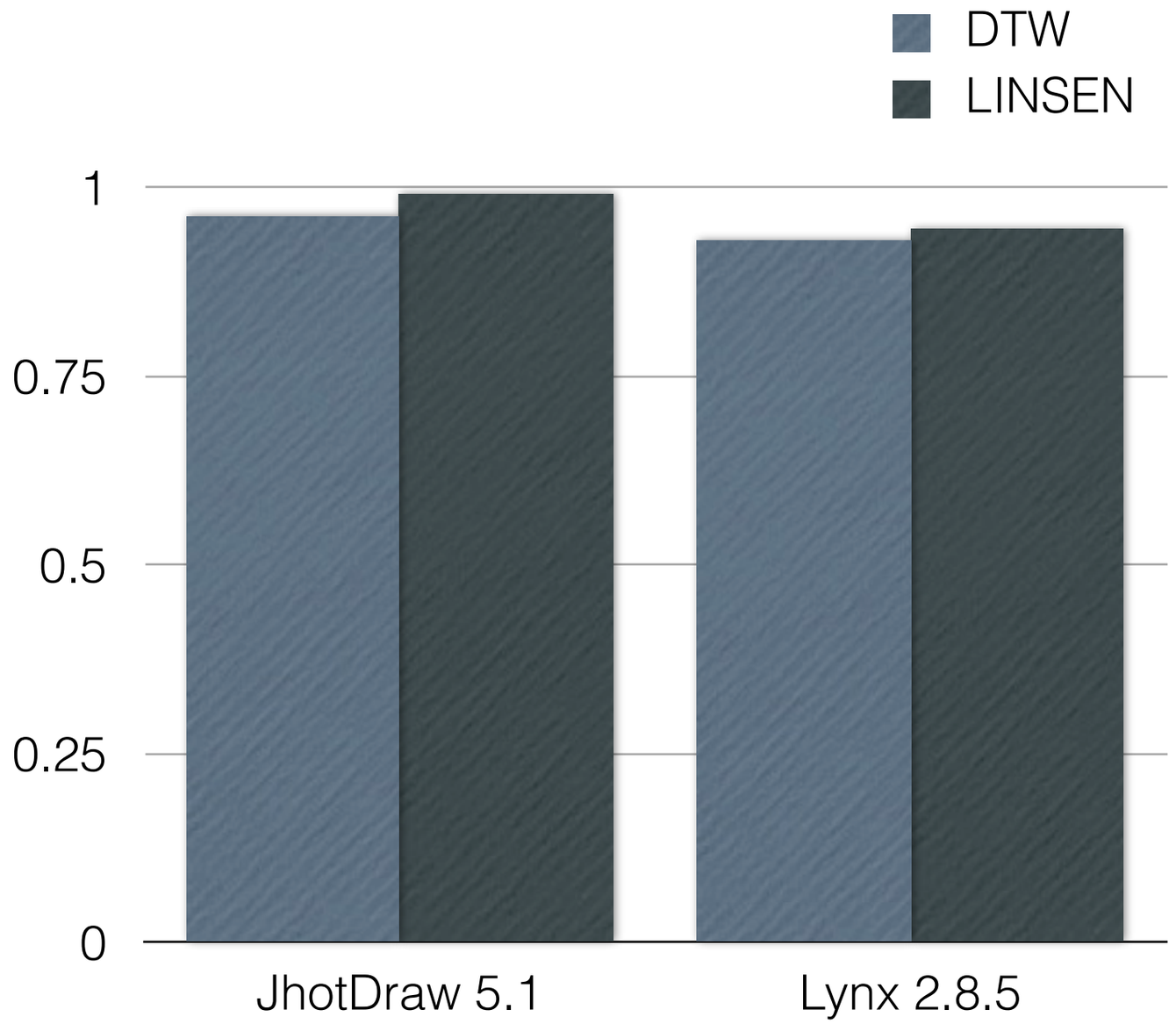




RE
SULTS
R Q 2

RQ2: MAPPING

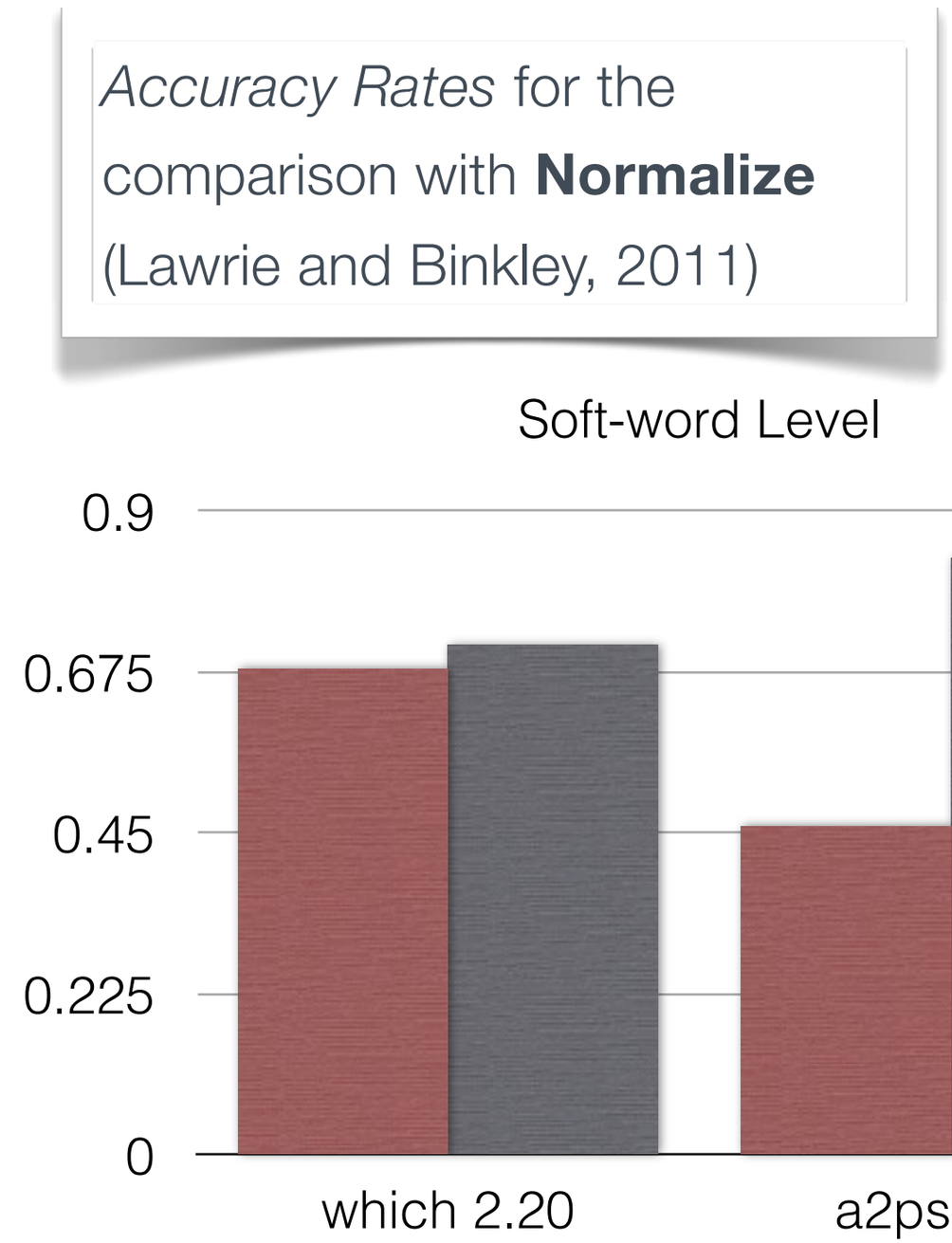
Accuracy Rates for the comparison with **DTW** (Madani et. al 2010)

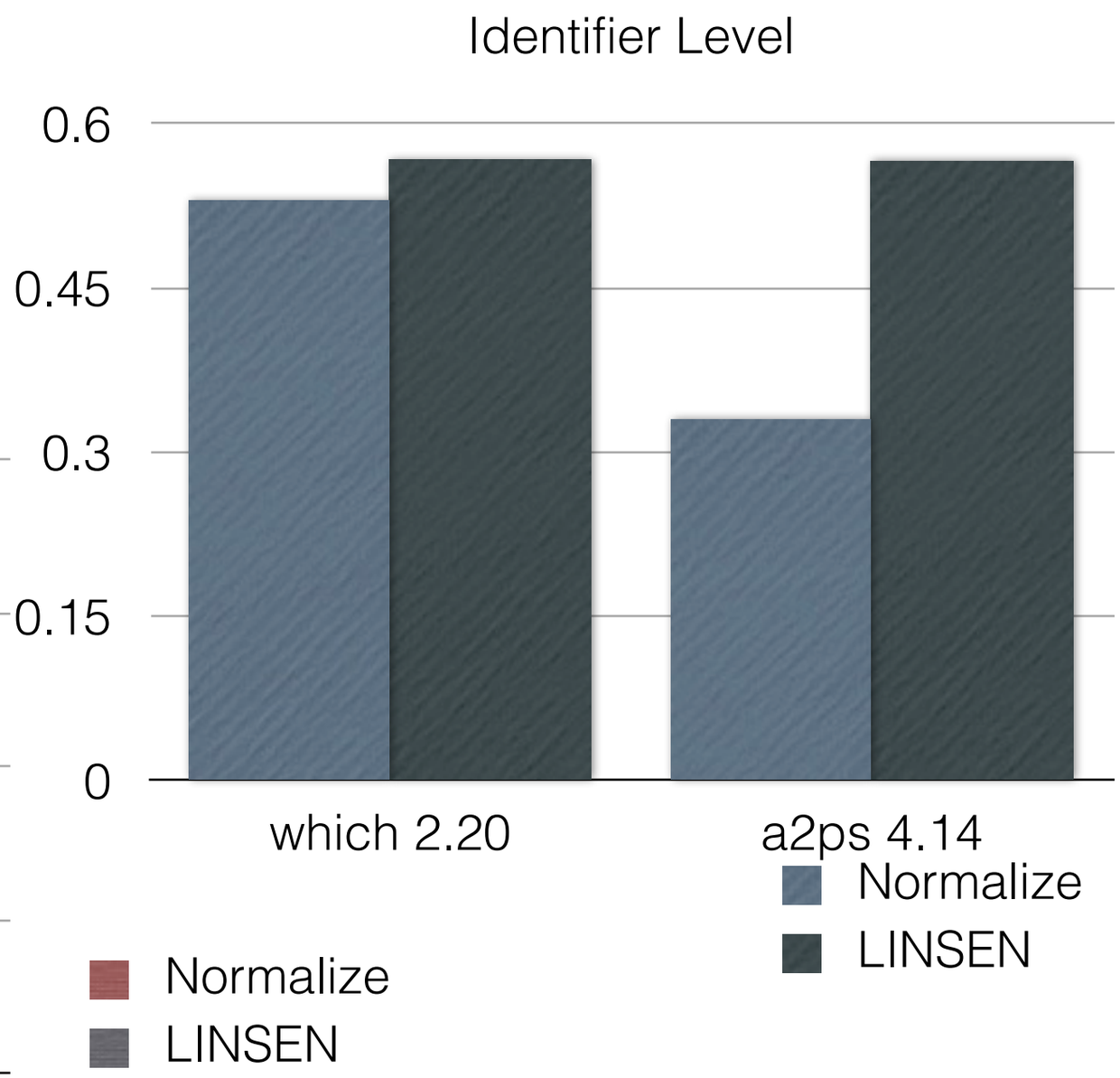


RESULTS
RQ2

RQ2: MAPPING

Accuracy Rates for the comparison with **Normalize** (Lawrie and Binkley, 2011)



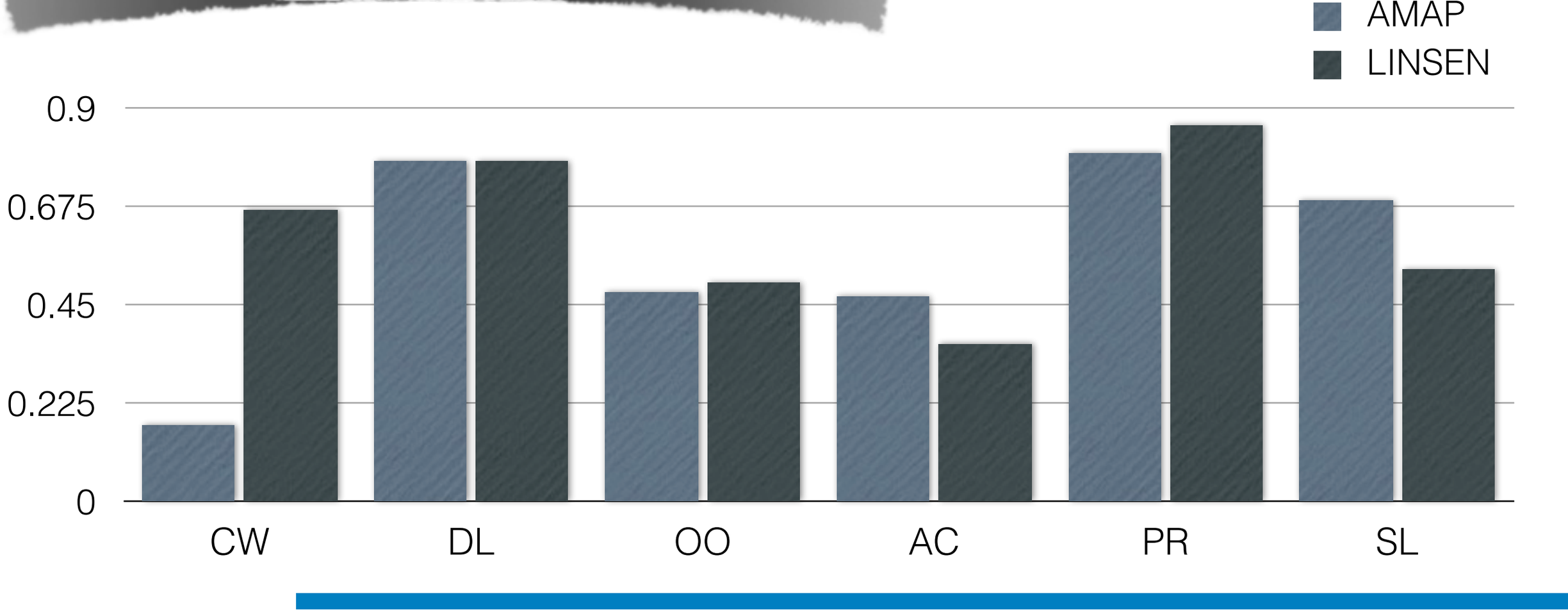


RESULTS
RQ 3

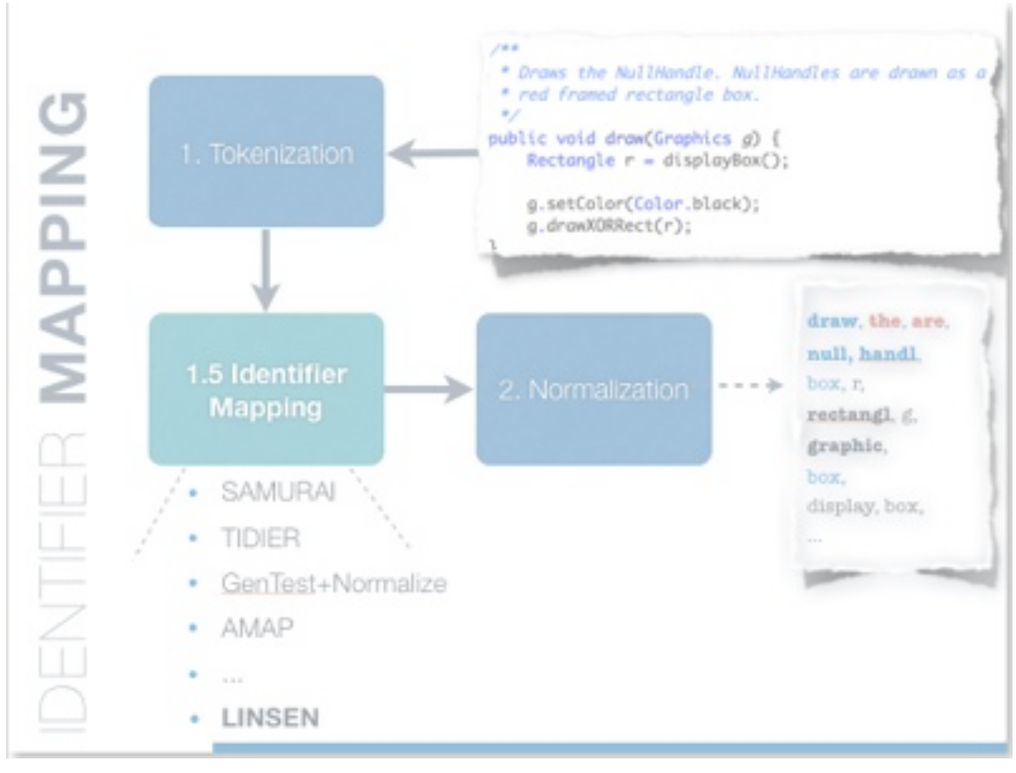
RQ3: EXPANSION

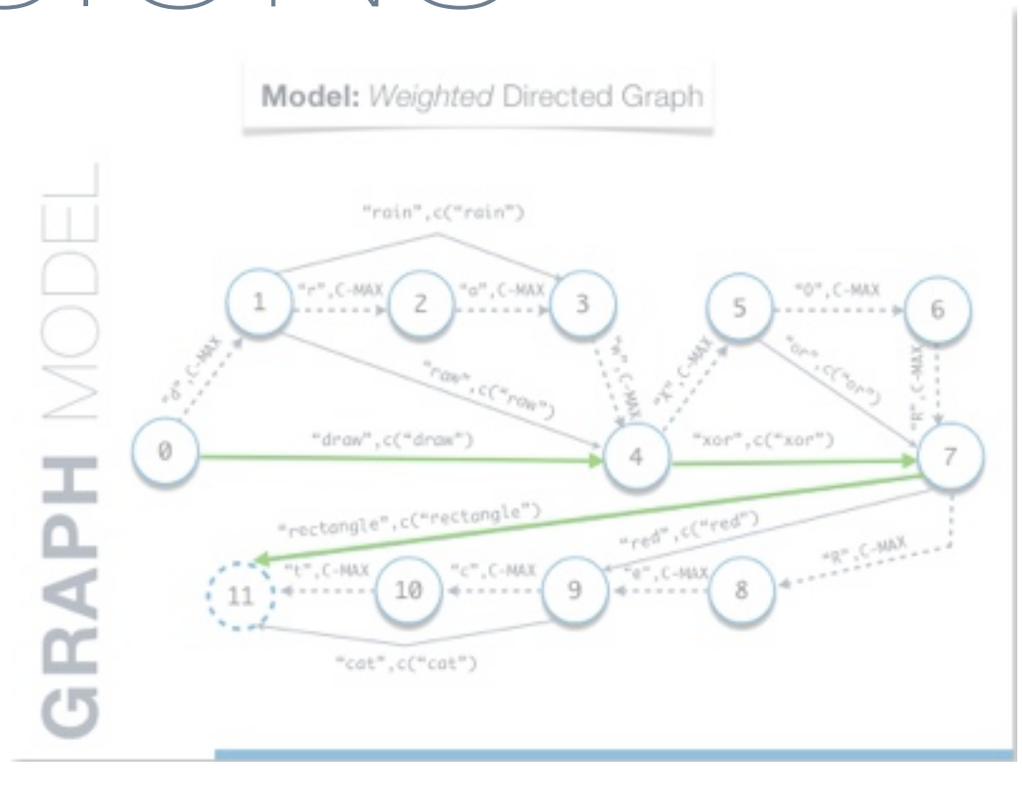
Accuracy Rates for the comparison with **AMAP** (Hill and Pollock, 2008)

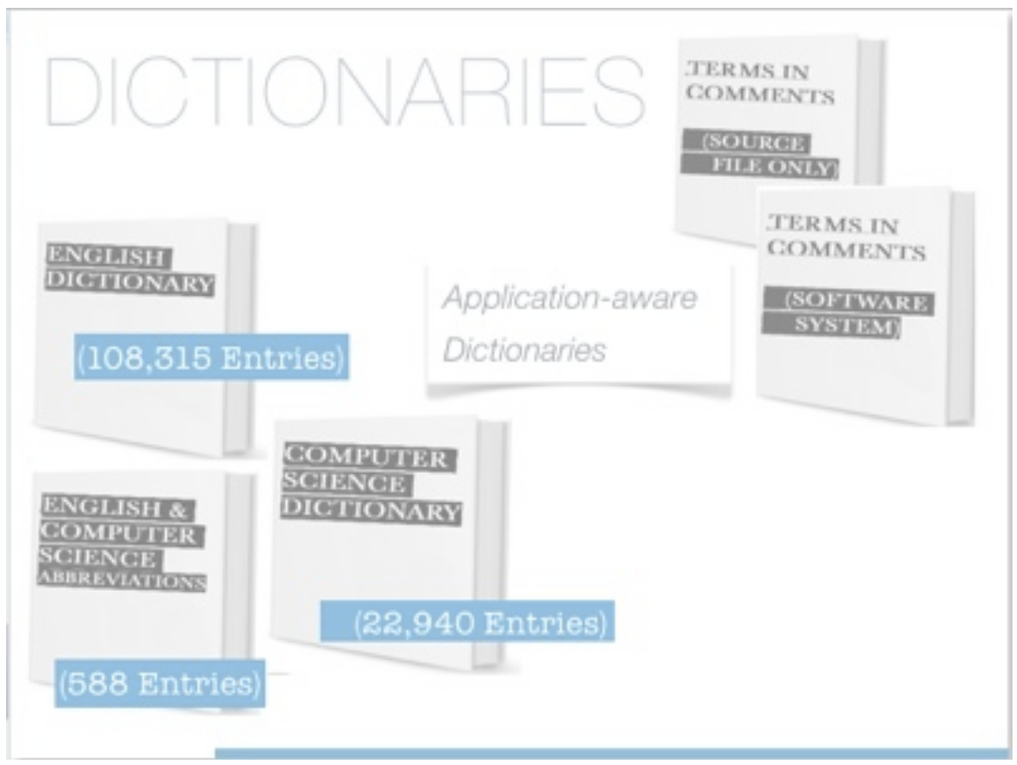
CW: Combination Words
DL: Dropped Letters
OO: Others
AC: Acronyms
PR: Prefix
SL: Single Letters

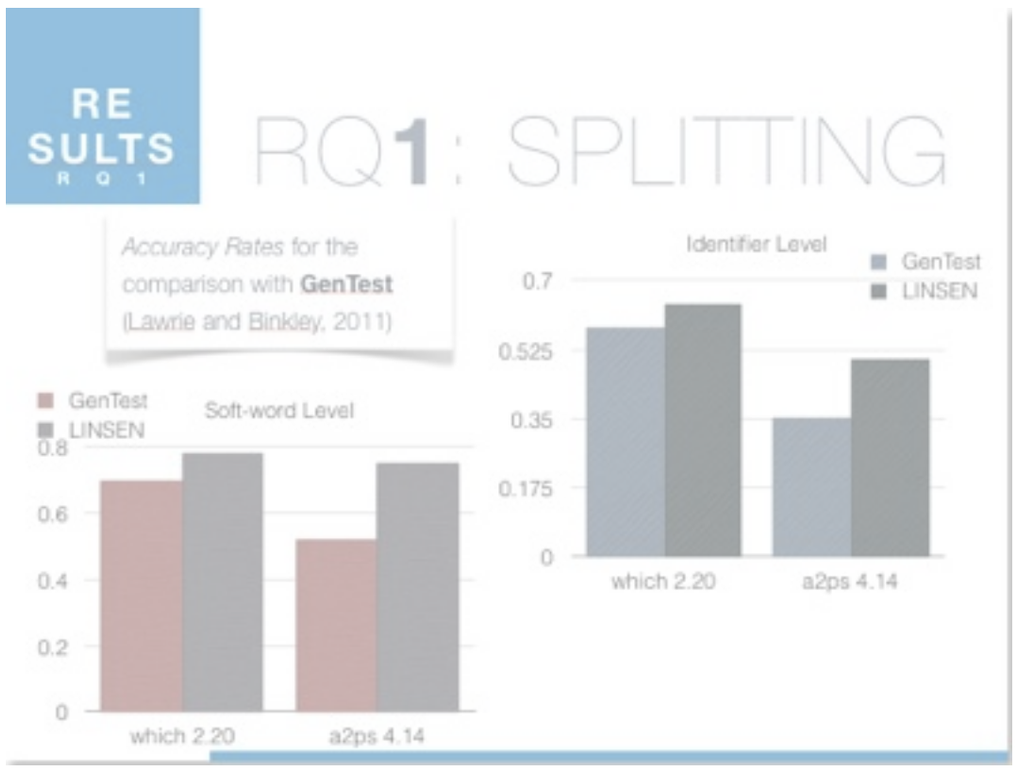


CONCLUSIONS

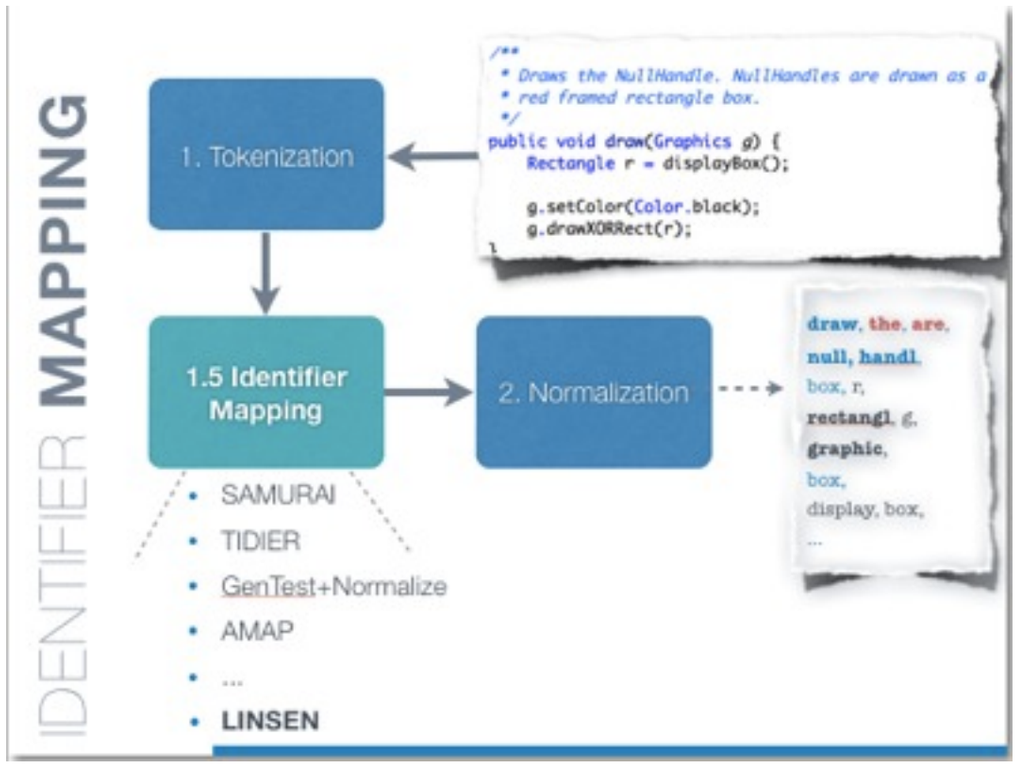


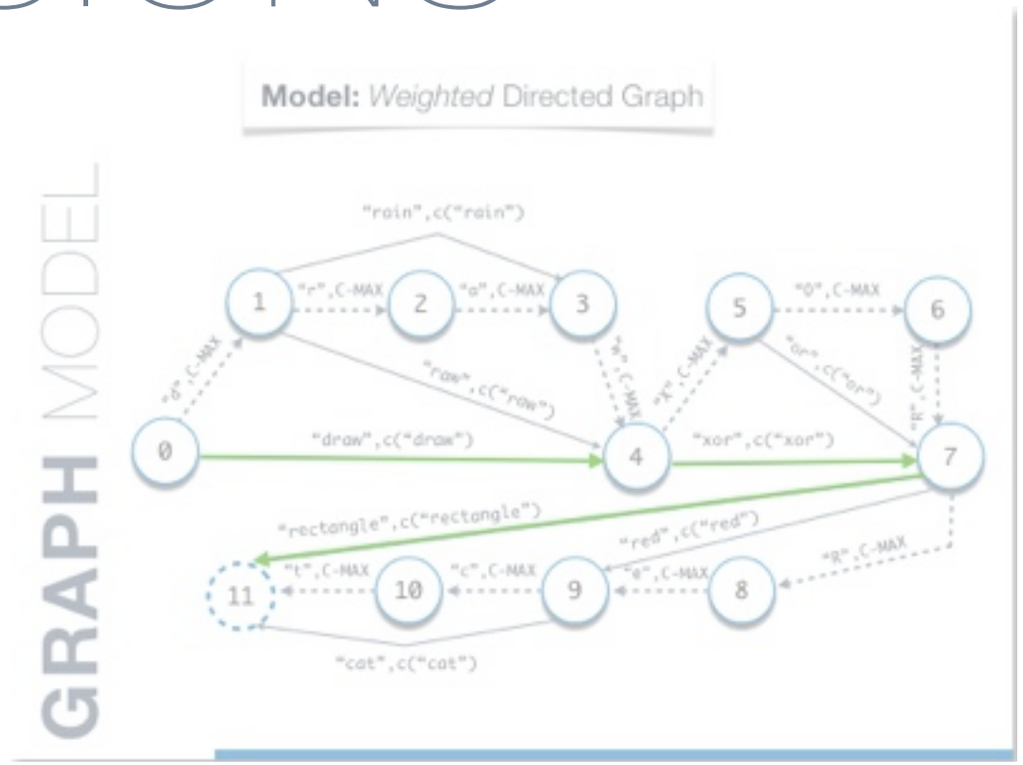


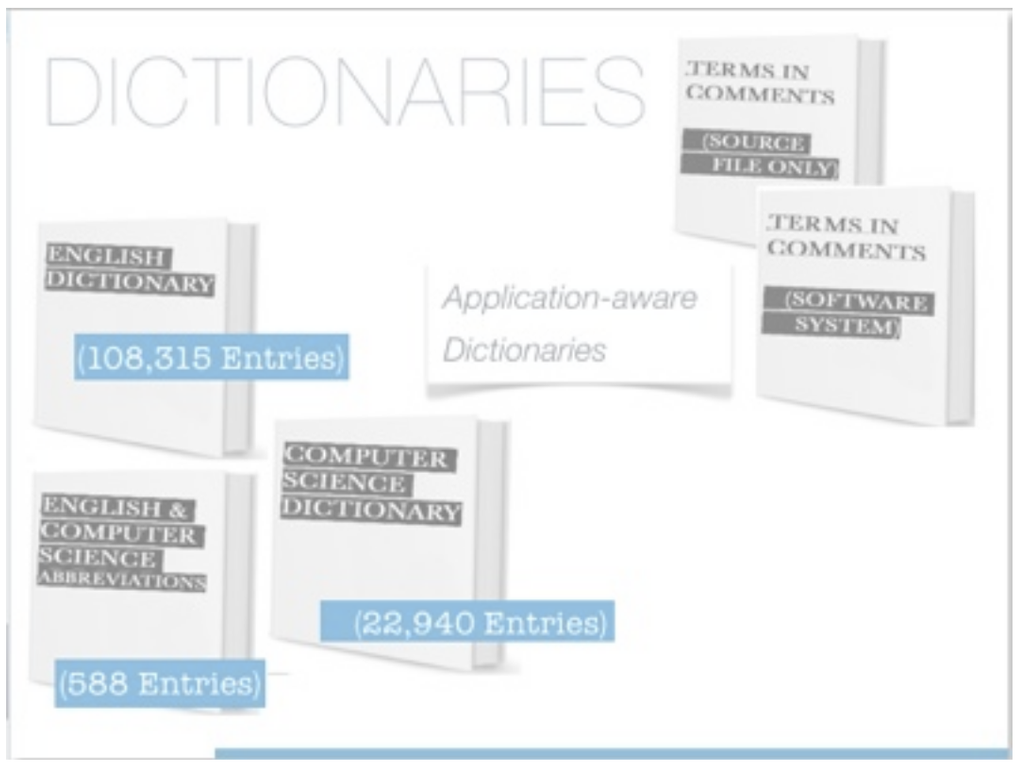


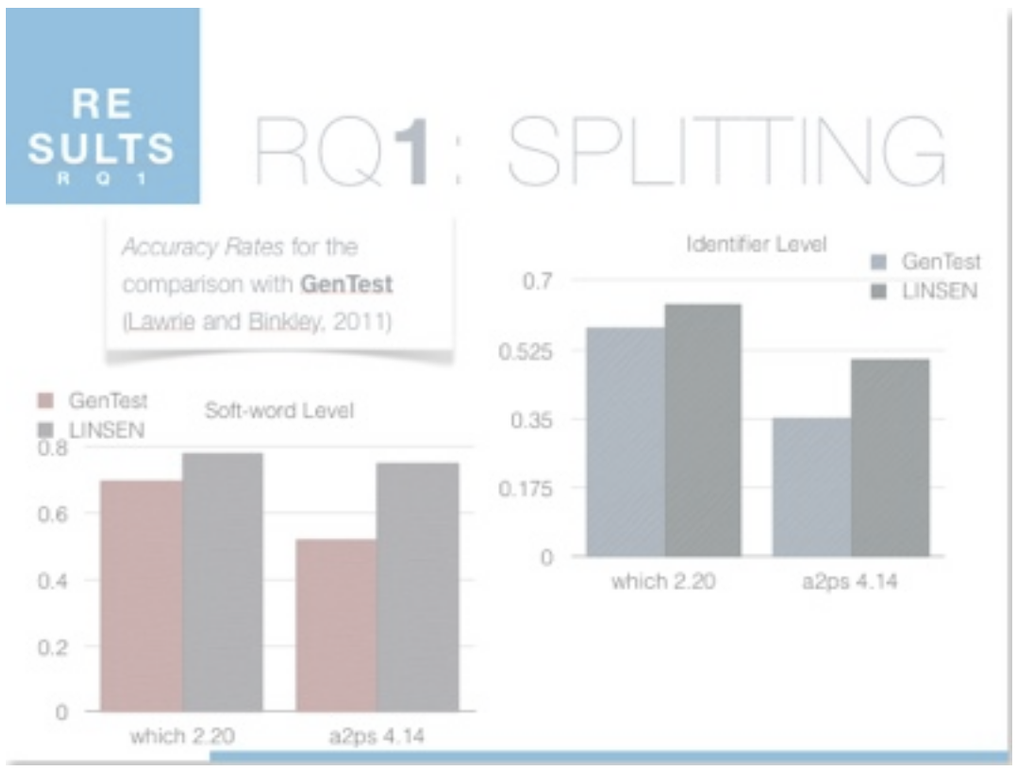


CONCLUSIONS

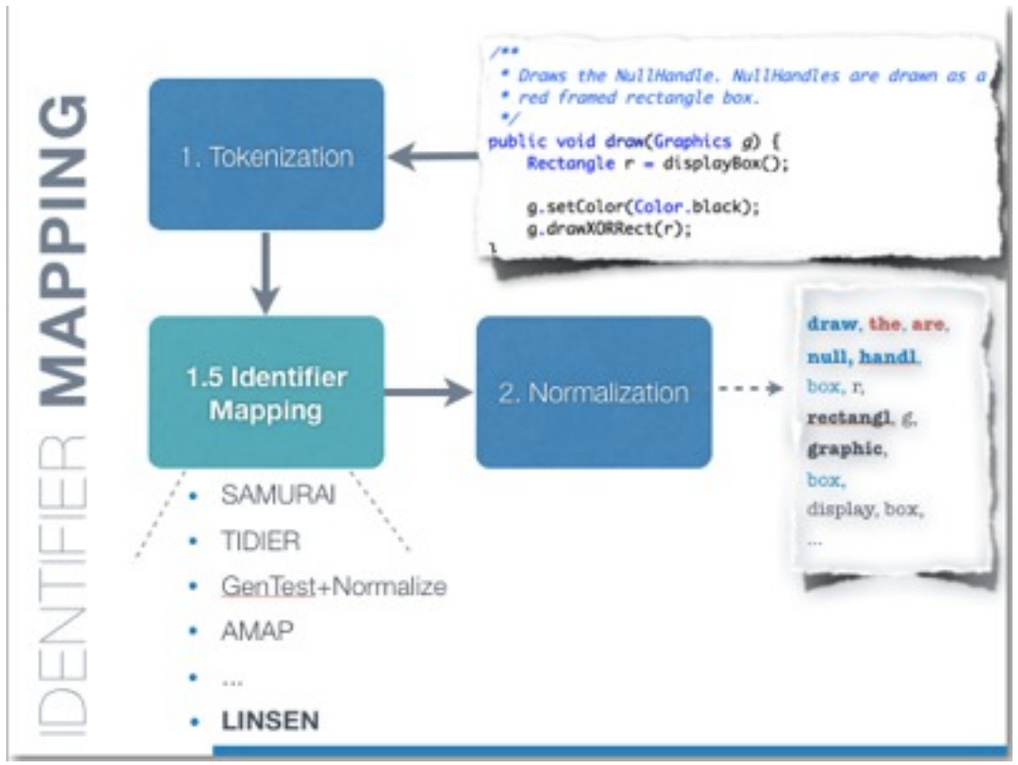


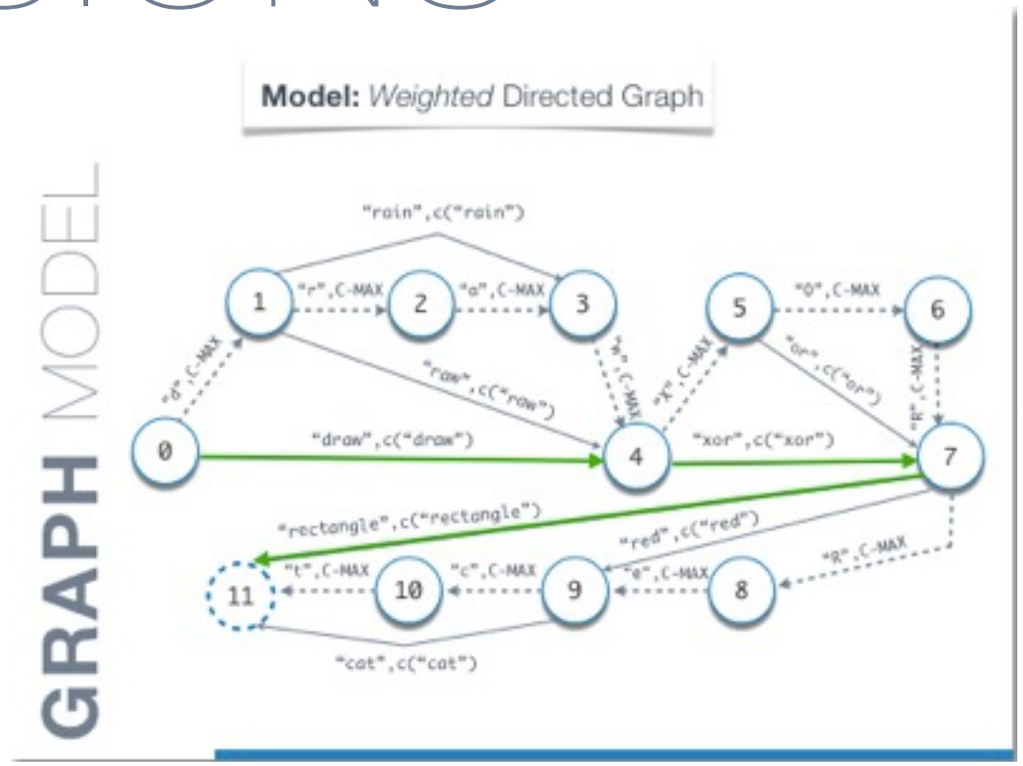


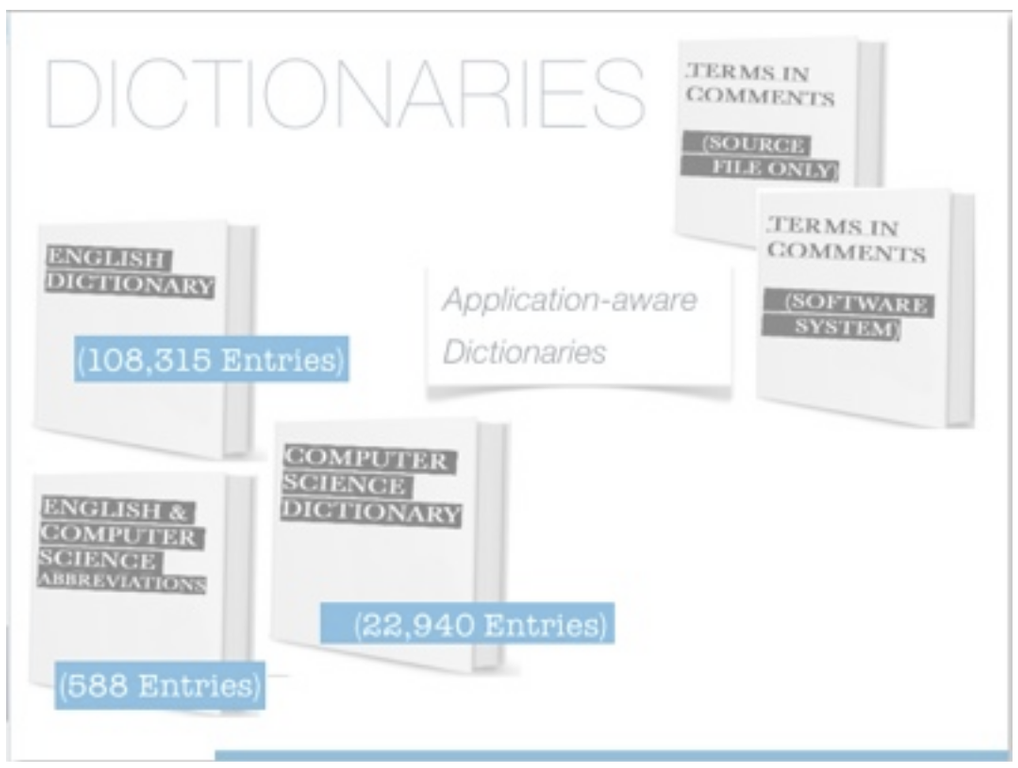


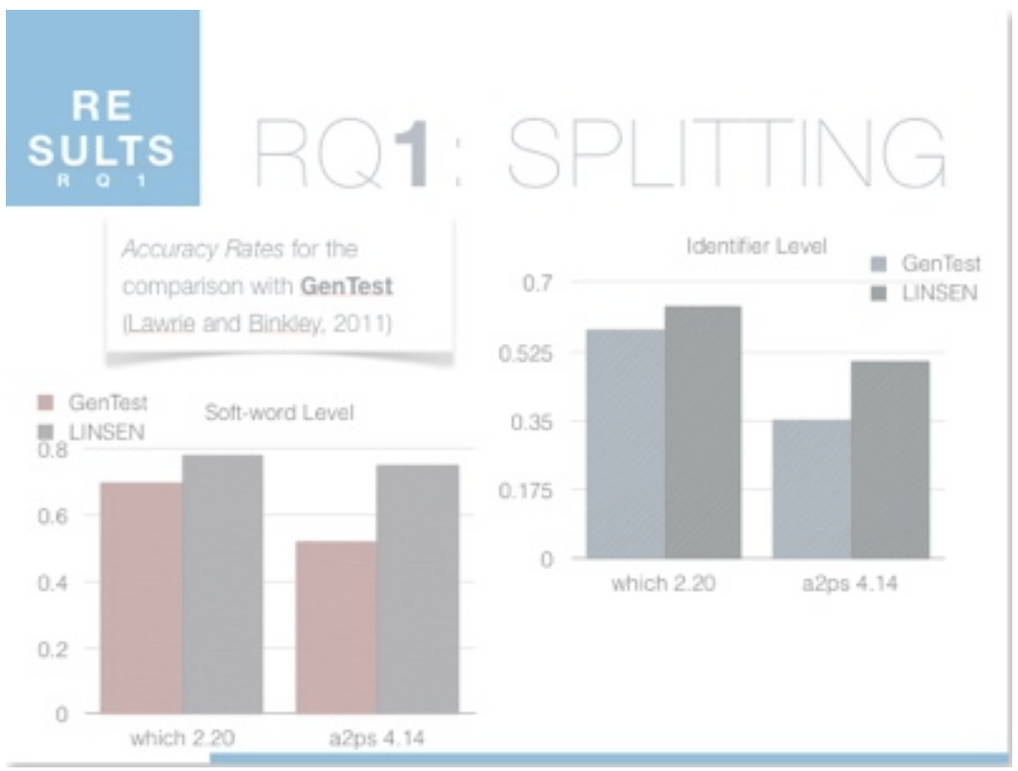


CONCLUSIONS

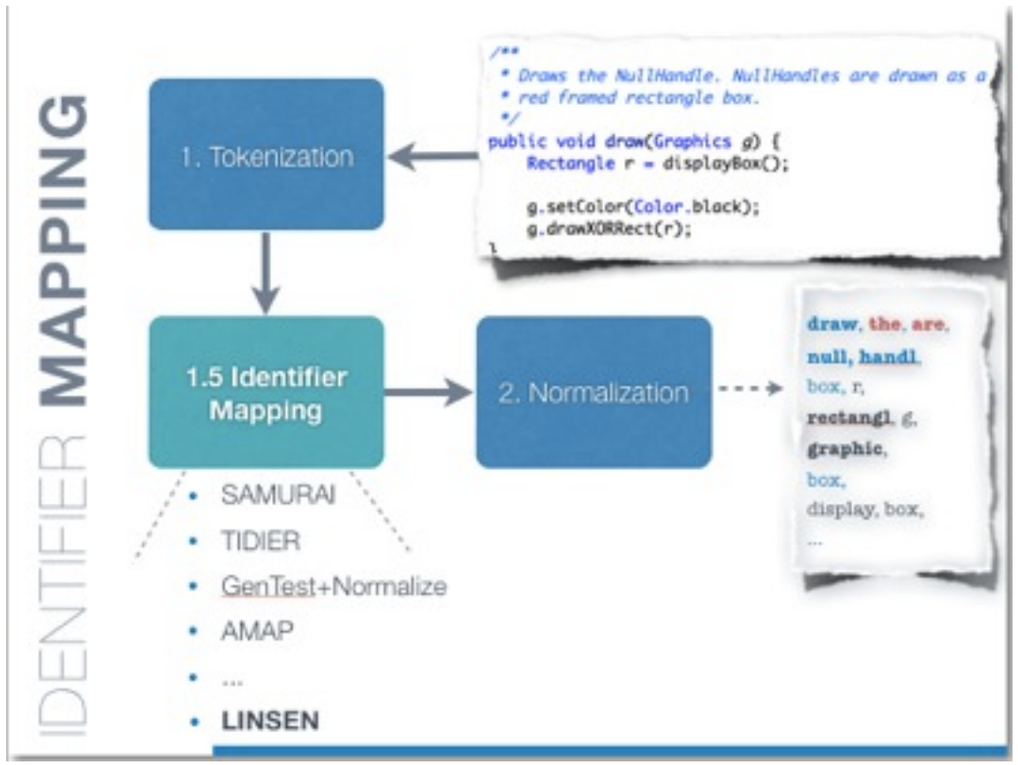


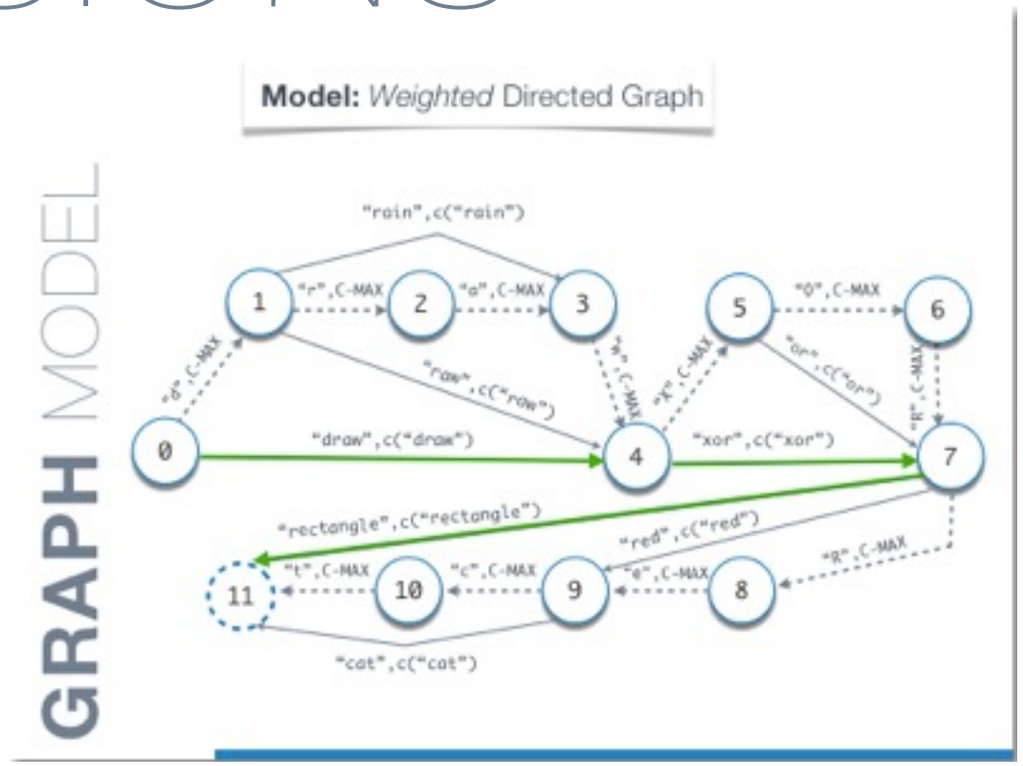


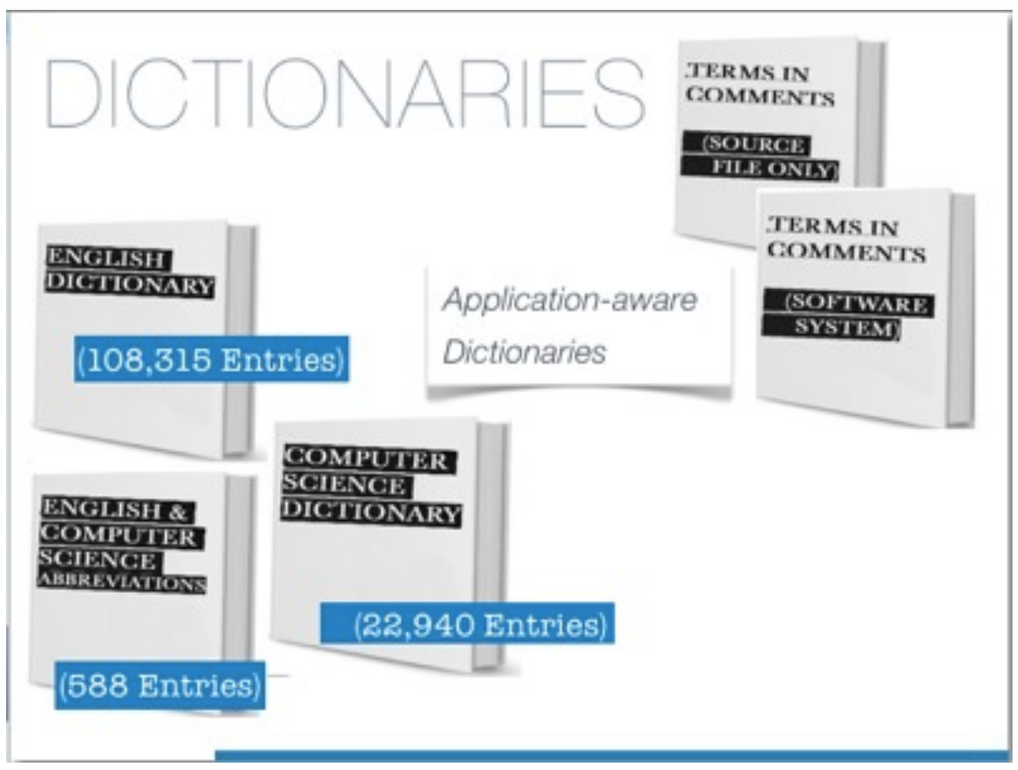


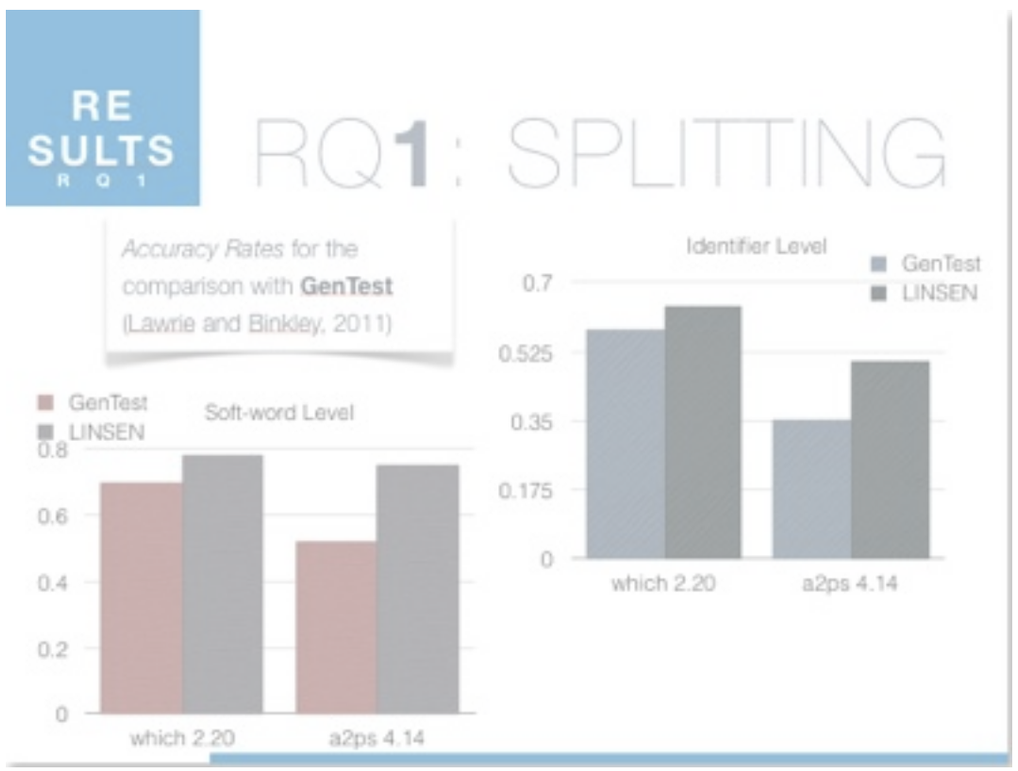


CONCLUSIONS

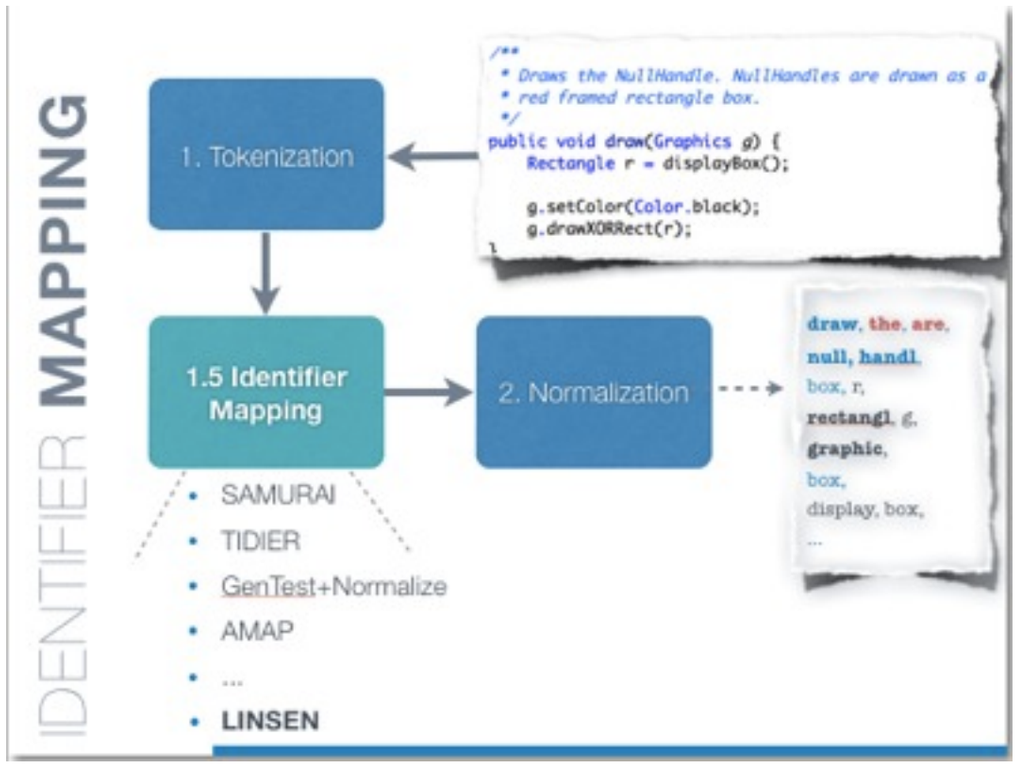


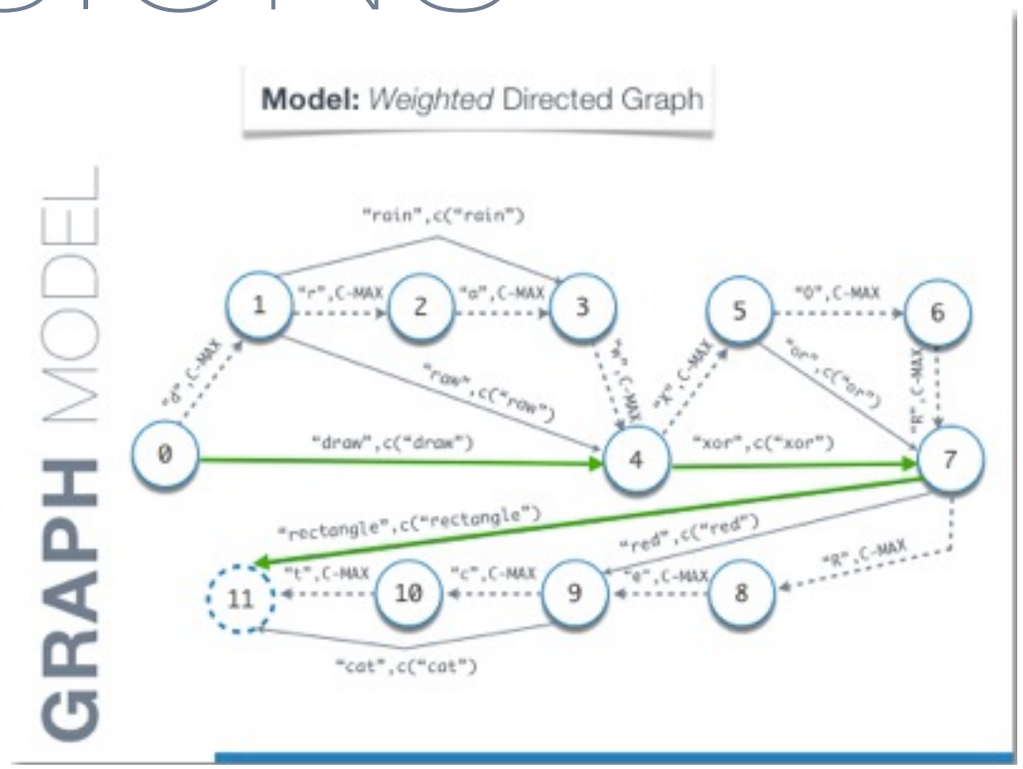


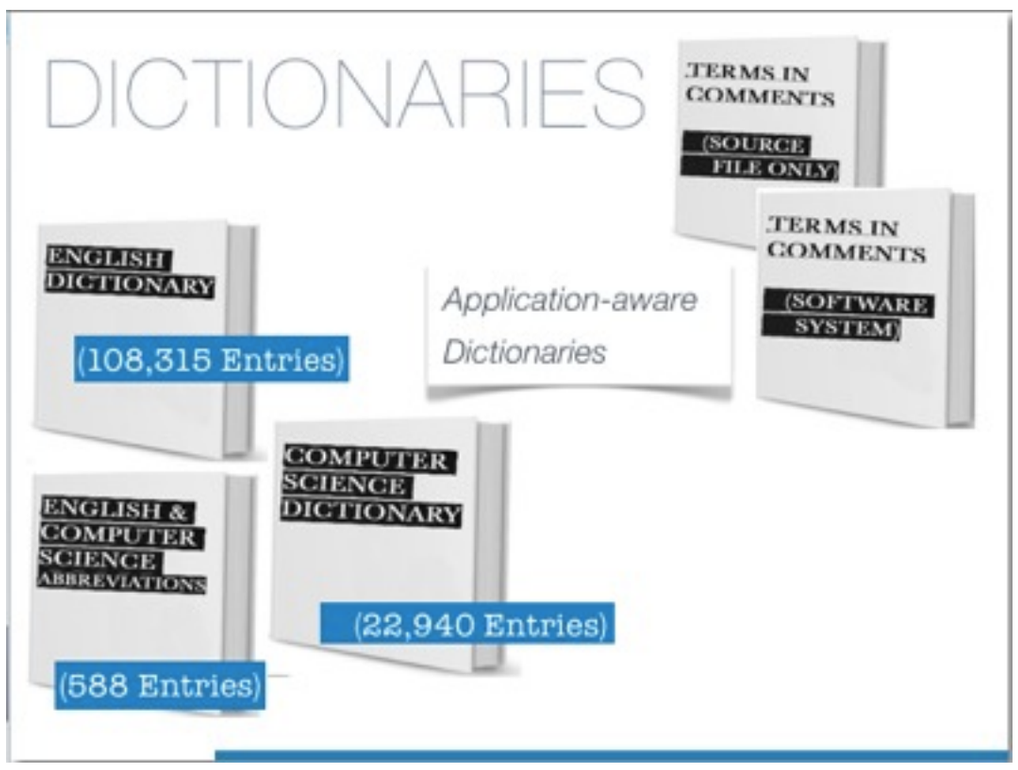


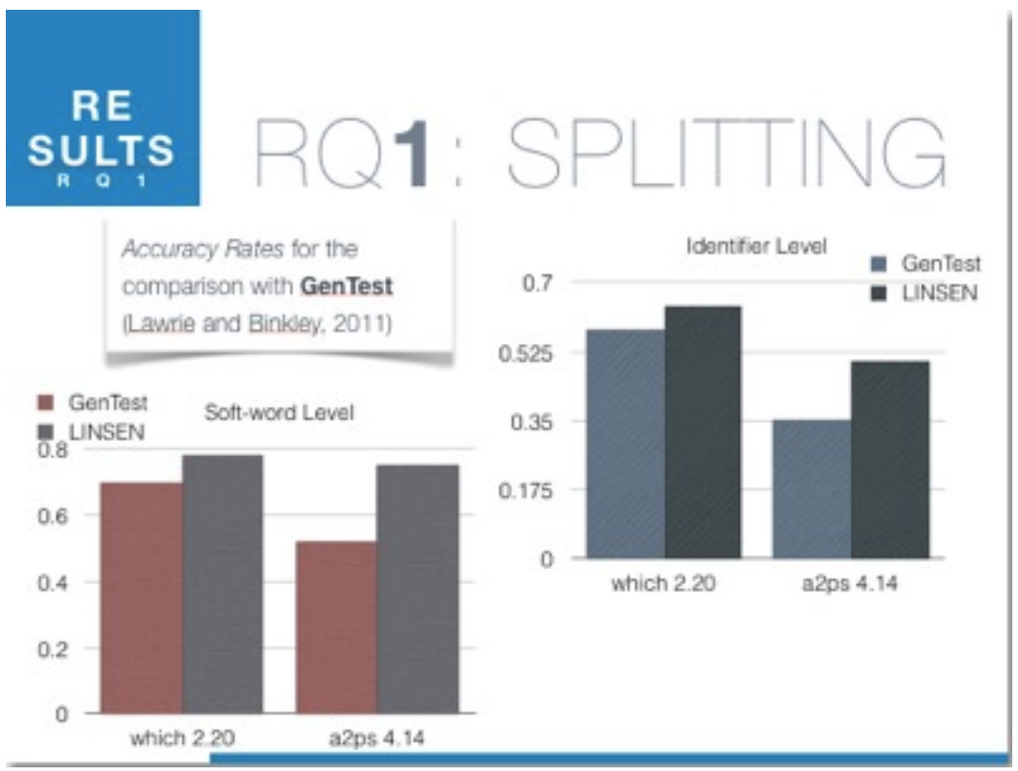


CONCLUSIONS









FUTURE WORKS

- Evaluation of the impact of each adopted **dictionary** on the performance
  - **Improve** or **change** or **add** dictionaries
- Improve the implementation of the prototype to speed up the computation
- Make use of parallel computation to process each identifier in isolation

THANK YOU

Valerio Maggio

Ph.D. Student, University of Naples *"Federico II"*

valerio.maggio@unina.it

26th Sept. 2012, ICSM2012@Riva del Garda(Trento), Italy