kNarrator

Adriano L. Kerber

kerberpro@gmail.com

References

[1] - Obtida do software Twine que pode ser baixado em http://twinery.org/
[2] - Obtida do site: http://brightspiral.com/tracery/
[3] - Obtidas através do *software* Wide Ruled 2: https://games.soe.ucsc.edu/project/wide-ruled
[4] a [9] - Imagens do kNarrator

kerberpro@gmail.com

SBGames 2017 Curitiba PR

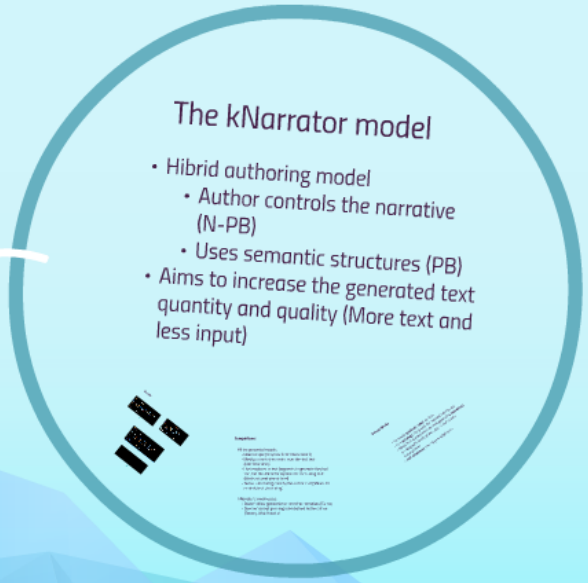
UNISINOS

kNarrator

Adriano L. Kerber, Daniel Camozzato, Rossana B. Queiroz, Vinicius J. Cassol - kerberpro@gmail.com

SBGames 2017 Curitiba PR

UNISINOS



Prezi

Objective

- Research motive:
  - Time spent by an author generating a narrative
  - Lack of autonomy from narrative generation tools
  - Curiosity about procedural generation of narratives
- General objective:
  - Develop an authoring model that allows text generation in a automatized way.

Subjects

- Authoring models
- Natural Language Processing (NLP)
- The kNarrator model

Study fields

- Linguistics (NLP)
- Narrative (Authoring)

Study fields

- Linguistics (NLP)
- Narrative (Authoring)

Authoring Models

Aim to facilitate the writers' job

Proposed Taxonomy:

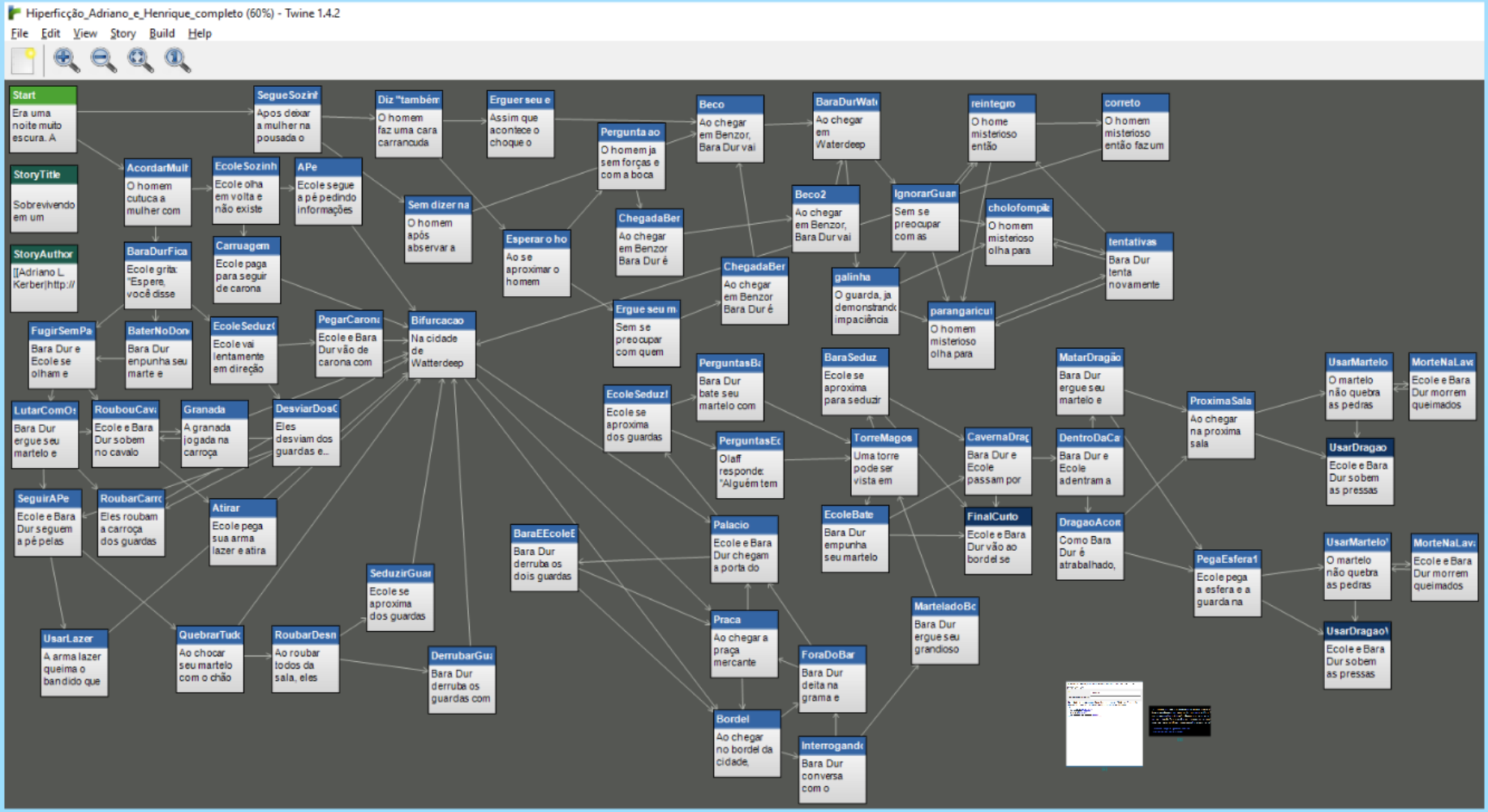
- **Not-Plot based (N-PB)**
  - Not focused on the narrative structure
  - Simple rules
- **Plot based (PB)**
  - Models focused on the narrative structure
  - Complex rules

Proposed Taxonomy:

- **Not-Plot based (N-PB)**
  - Not focused on the narrative structure
  - Simple rules
- **Plot based (PB)**
  - Models focused on the narrative structure
  - Complex rules

Twine

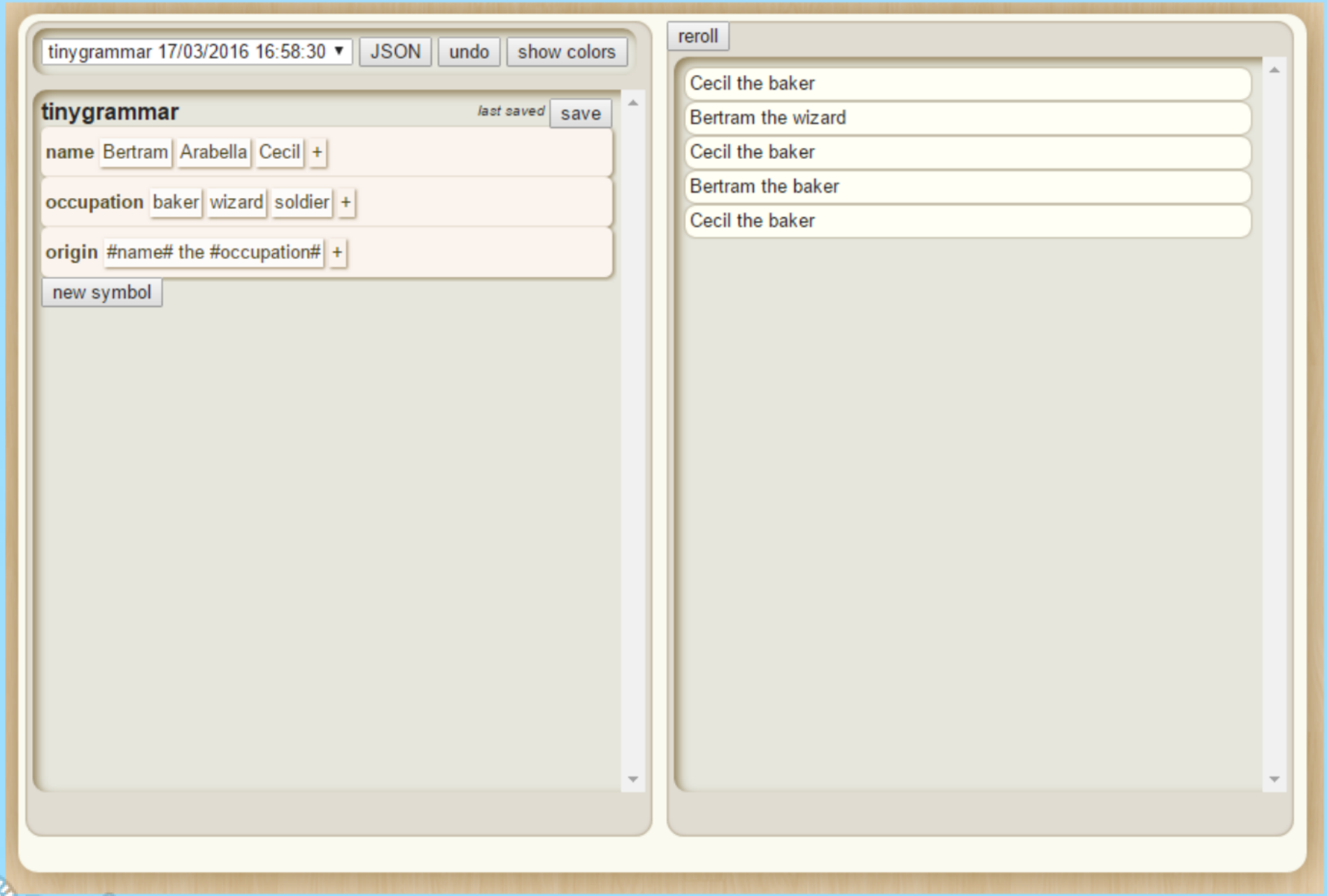
Not-Plot based



[1]

Tracery

Not-Plot based



Wide Ruled 2

Plot based

MurderMysteryInteractive.wr2 - Wide Ruled 2.0, Release 3.61

File

Characters

John Smith
Jerry Fontana
Cindy Rollins
Gene Franks
Rachel Delores
Dick Tracy
Sherlock Holmes

New | Edit ... | Delete

Environments

The Docks
The Alleyway
The City Park
The Shady Bar
The Abandoned Shack

New | Edit ... | Delete

Plot Point Types

Crime Info
Murderer
Murder Solved

New | Edit ... | Delete

Goals and Plot Fragments

- **Do Murder Mystery**
  - Do Story
    - *Do the Crime*
    - *Find a Detective*
    - *Investigate*
    - *Solve the Crime*
    - *Capture the Murderer*
    - *Restart Story*
  - No more alive victims
- **Do the Crime (Text Victim Name)**
  - Random Murderer
    - *Attack the Victim*
  - Murderer is Enemy
    - *Attack the Victim*
- **Attack the Victim (Text victim name)**
  - Gunshot
  - Stabbing
- **Find a Detective (Text Detective Name)**
  - Detective sleeping
  - Detective drunk
- **Investigate**
  - Find crime scene evidence
    - *Investigate*
  - Interview friend
    - *Investigate*
  - Interview coworker
    - *Investigate*
  - Find clue at murderer hideout
    - *Investigate*
  - Put together clues and finish investigation
- **Solve the Crime**
  - Solve with celebration
  - Solve with smugness
- **Capture the Murderer**
  - Capture alive
  - Capture alive after fight
- **Restart Story**
  - Clean up plot points and restart
    - *Do Murder Mystery*
- **Change the Murderer**
  - Murderer is random other person

New ... | Edit ... | Delete

Interactivity Actions ...

Generate Story

Show Status Messages

[3]

Natural Language Processing

- Translation and text interpretation
- Data retrieval in documents
- Interface man-machine
- Summarization of documents

Proposed Classification:

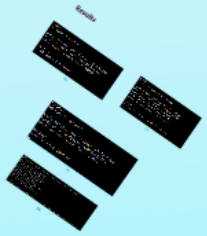
- **Normalization**
  - Stemming and Lemmatization algorithms
- **Disambiguation**
  - Algorithms of Part-Of-Speech tagging

Proposed Classification:

- **Normalization**
  - Stemming and Lemmatization algorithms
- **Disambiguation**
  - Algorithms of Part-Of-Speech tagging

The kNarrator model

- Hibrid authoring model
  - Author controls the narrative (N-PB)
  - Uses semantic structures (PB)
- Aims to increase the generated text quantity and quality (More text and less input)



Comparisons

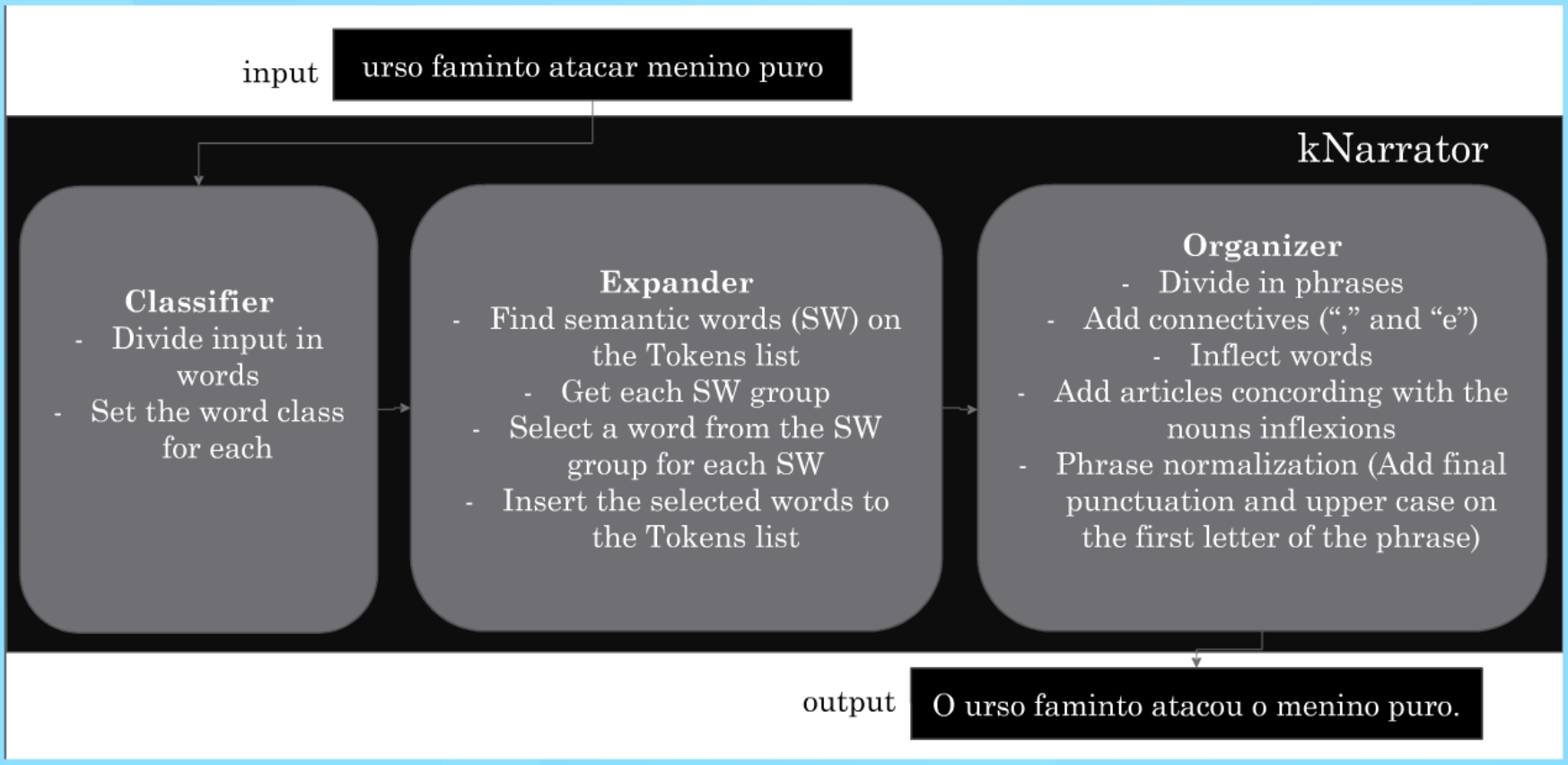
All the presented models
- Needs a specific syntax (kNarrator doesn't)
- Mostly, do not offer control over the final text (kNarrator does)
- Use templates or text fragments to generate the final text, but the kNarrator expands its texts using NLP (Works at word atomic level)
- Textual structuring done by the author itself (kNarrator controls text structuring)

kNarrator's weaknesses:
- Doesn't allow generation of branched narratives (Twine)
- Does not accept grouping rules defined by the author (Tracery, Wilde Ruled 2)

Future Works

- Generate **symbolic rules** system
- Use **ontology** along with the semantic dictionary
- Allow that the expander module generates **narratives** working with levels of description and events management
- Add **coherence** among generated texts

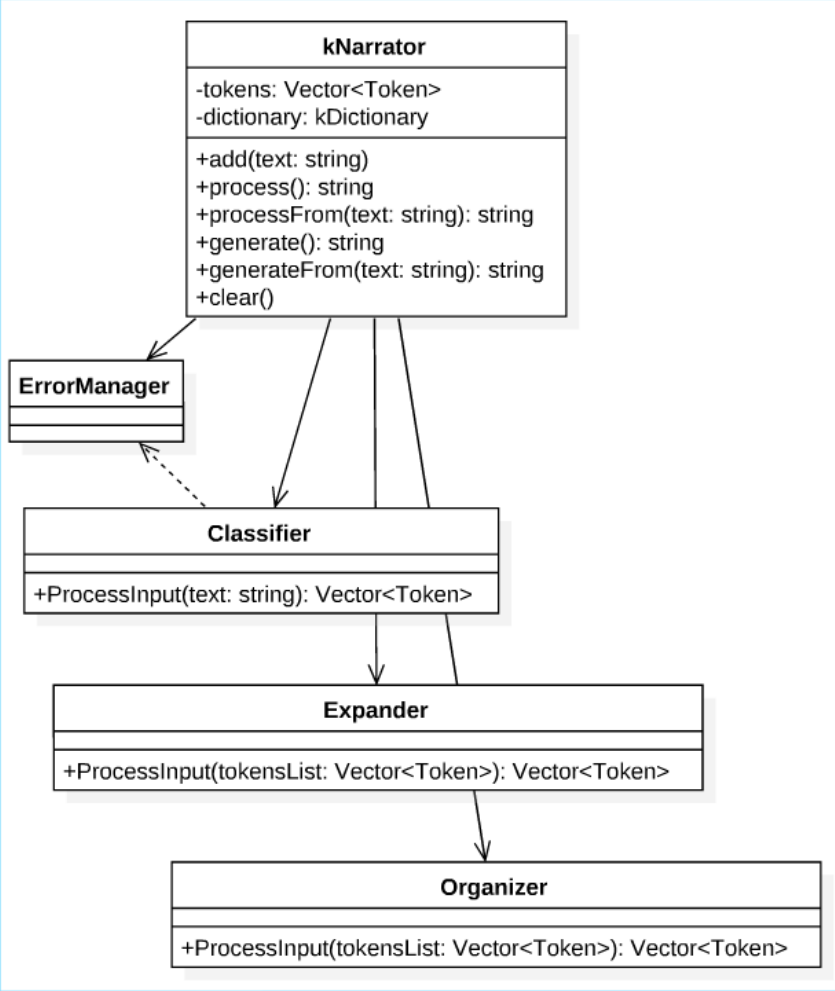
Pipeline



[4]

Modules

Three main modules and an auxiliary



[5]

(line number + word ID)

Classifier

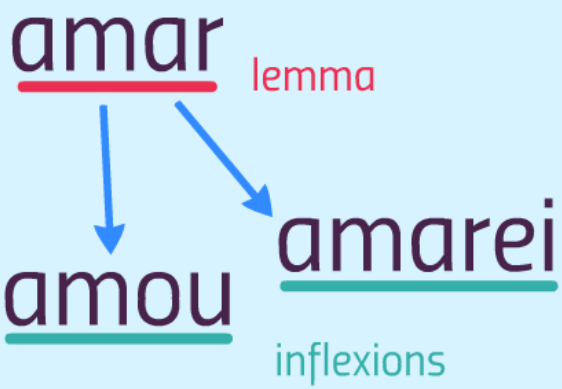
Classify the input words in Word Classes using a dictionary

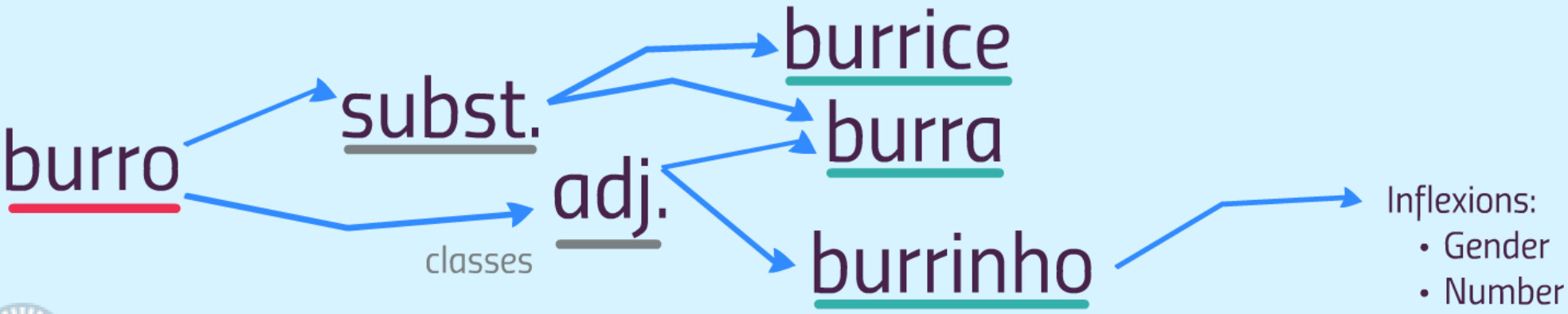
- It receives input (pseudo-text)
- It compares the input words with the **dicionary** words
- Each word becomes a Token

Dictionary

What is saved on the dictionary is the word's lemma along with all the inflexions of this word.

The dictionary uses a SQL database.

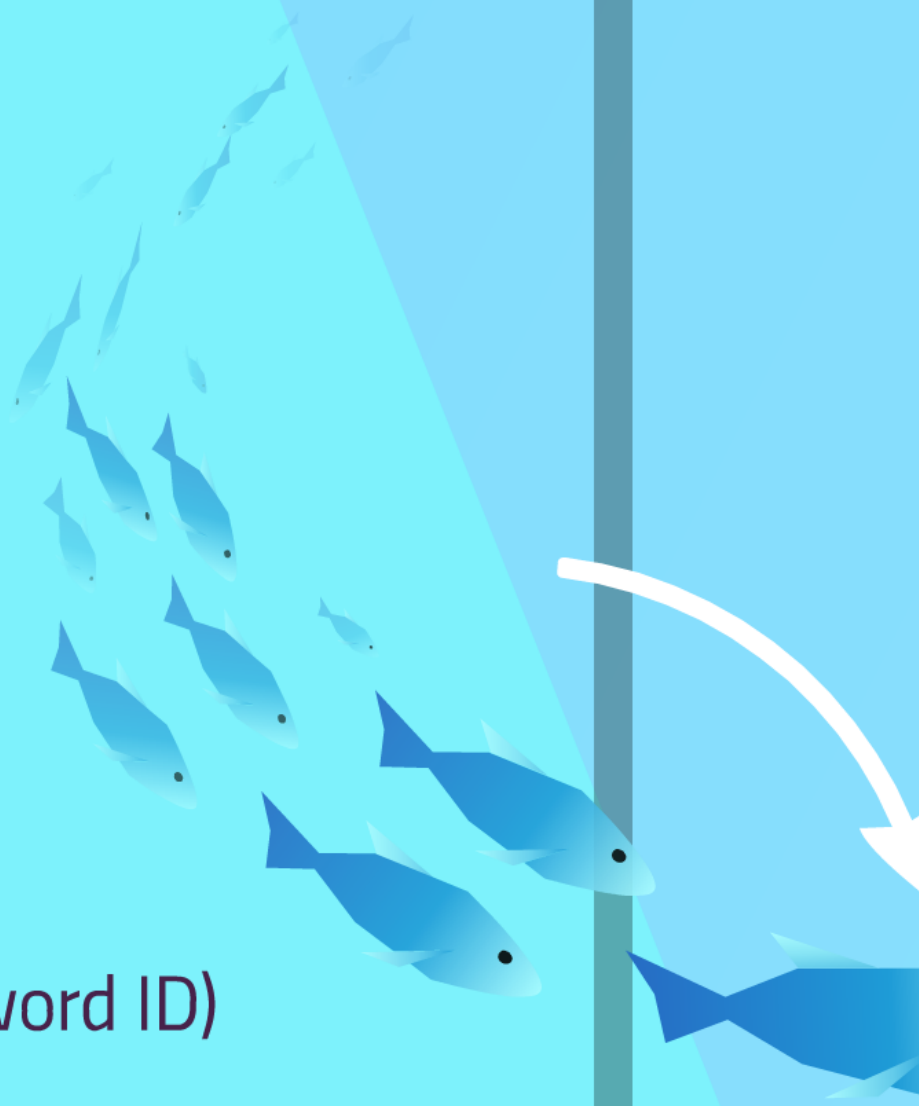




ErrorManager

Not identified
words are sent to
this module

(line number + word ID)



Expander

Insert new words in order to expand the input text with more details (Optional module)

"Urso atacar homem floresta" generates the result: "Urso grande forte atacar homem floresta denso silencioso".
The words "Urso" and "floresta" receive an increment of words to its descriptions.

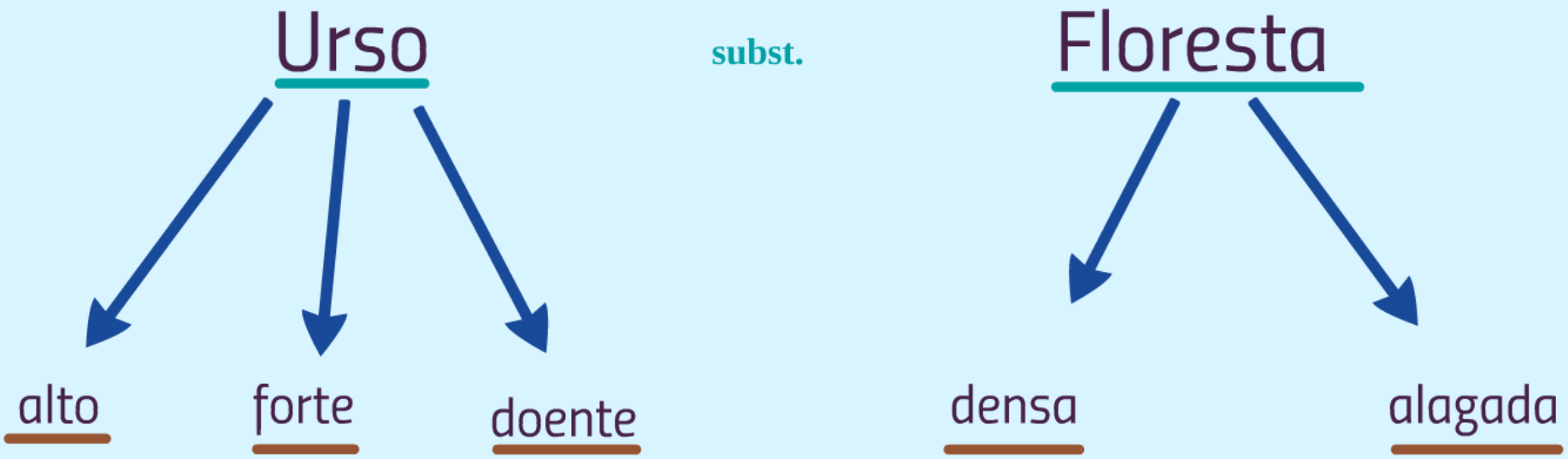
- It receives the Tokens' list
- Match the words with the **semantic dictonary**
- Insert new words to describe the selected tokens

Semantic Dictonary

Structure that has words with multiple results (word - set of words)

Classification structure:

- Characters
- Places



Organizer

It receives a Tokens' list and transforms it in fluid text

- Phrases: devide the Tokens in verbal phrases
- Punctuation: adds punctuation to the phrases generated
- Concordance: inflect the Tokens to concord among themselves
- Conectives: adds "," and "e" to connect the words
- Articles: adds articles preceding nouns
- Realization: adds capital letters to the beginning of the phrases

Control parameters

Inflects the text following the parameters defined by the author

There are four control parameters to force the inflections to the final text:

- Gender
- Number
- Person
- Time

Results

```
Input:
Adriano ir aprovar

Word: Adriano. Word Class: SUBSTANTIVO
Word: ir. Word Class: VERBOAUXILIAR
Word: aprovar. Word Class: VERBO

Output:
O Adriano irá aprovar.
```

[6]

```
Input:
Paulo ir beber comer dormir

Word: Paulo. Word Class: SUBSTANTIVO
Word: ir. Word Class: VERBOAUXILIAR
Word: beber. Word Class: VERBO
Word: comer. Word Class: VERBO
Word: dormir. Word Class: VERBO

Output:
O Paulo foi beber, comer e dormir.
```

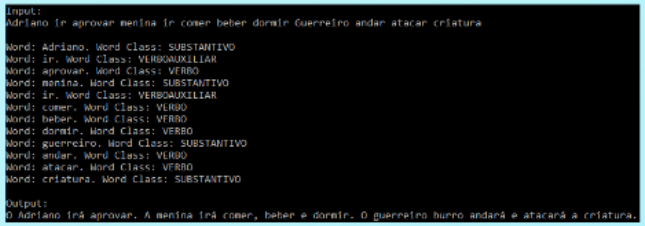
[7]

```
Input:
Adriano ir aprovar

Word: Adriano. Word Class: SUBSTANTIVO
Word: ir. Word Class: VERBOAUXILIAR
Word: aprovar. Word Class: VERBO

Output:
O Adriano vai aprovar.
```

[8]



[9]

```
Input:
Adriano ir aprovar

Word: Adriano. Word Class: SUBSTANTIVO
Word: ir. Word Class: VERBOAUXILIAR
Word: aprovar. Word Class: VERBO

Output:
O Adriano irá aprovar.
```

[6]

```
Input:
Paulo ir beber comer dormir

Word: Paulo. Word Class: SUBSTANTIVO
Word: ir. Word Class: VERBOAUXILIAR
Word: beber. Word Class: VERBO
Word: comer. Word Class: VERBO
Word: dormir. Word Class: VERBO

Output:
O Paulo foi beber, comer e dormir.
```

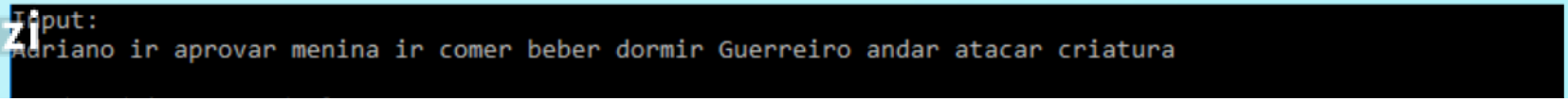
[7]

```
Input:
Adriano ir aprovar

Word: Adriano. Word Class: SUBSTANTIVO
Word: ir. Word Class: VERBOAUXILIAR
Word: aprovar. Word Class: VERBO

Output:
O Adriano vai aprovar.
```

[8]



[8]

```
Input:
Adriano ir aprovar menina ir comer beber dormir Guerreiro andar atacar criatura

Word: Adriano. Word Class: SUBSTANTIVO
Word: ir. Word Class: VERBOAUXILIAR
Word: aprovar. Word Class: VERBO
Word: menina. Word Class: SUBSTANTIVO
Word: ir. Word Class: VERBOAUXILIAR
Word: comer. Word Class: VERBO
Word: beber. Word Class: VERBO
Word: dormir. Word Class: VERBO
Word: guerreiro. Word Class: SUBSTANTIVO
Word: andar. Word Class: VERBO
Word: atacar. Word Class: VERBO
Word: criatura. Word Class: SUBSTANTIVO

Output:
O Adriano irá aprovar. A menina irá comer, beber e dormir. O guerreiro burro andará e atacará a criatura.
```

[9]

Comparisons

All the presented models

- Needs a specific syntax (kNarrator doesn't)
- Mostly, do not offer control over the final text (kNarrator does)
- Use templates or text fragments to generate the final text, but the kNarrator expands its texts using NLP (Works at word atomic level)
- Textual structuring done by the author itself (kNarrator controls text structuring)

kNarrator's weaknesses:

- Doesn't allow generation of branched narratives (Twine)
- Does not accept grouping rules defined by the author (Tracery, Wide Ruled 2)

Future Works

- Generate **symbolic rules'** system
- Use **ontology** along with the semantic dictionary
- Allow that the expander module generates **narratives** working with levels of description and events management
- Add **coherence** among generated texts

kNarrator

Adriano L. Kerber

kerberpro@gmail.com

References

[1] - Obtida do software Twine que pode ser baixado em http://twinery.org/
[2] - Obtida do site: http://brightspiral.com/tracery/
[3] - Obtidas através do *software* Wide Ruled 2: https://games.soe.ucsc.edu/project/wide-ruled
[4] a [9] - Imagens do kNarrator

kerberpro@gmail.com

SBGames 2017 Curitiba PR

UNISINO

References

[1] - Obtida do software Twine que pode ser baixado em http://twinery.org/
[2] - Obtida do site: http://brightspiral.com/tracery/
[3] - Obtidas através do *software* Wide Ruled 2: https://games.soe.ucsc.edu/project/wide-ruled
[4] a [9] - Imagens do kNarrator

kerberpro@gmail.com

SBGames 2017 Curitiba PR

UNISINOS

Prezi