

kNarrator

Adriano L. Kerber

kerberpro@gmail.com



Referências

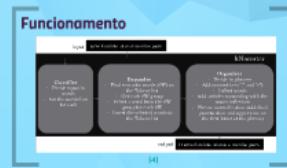
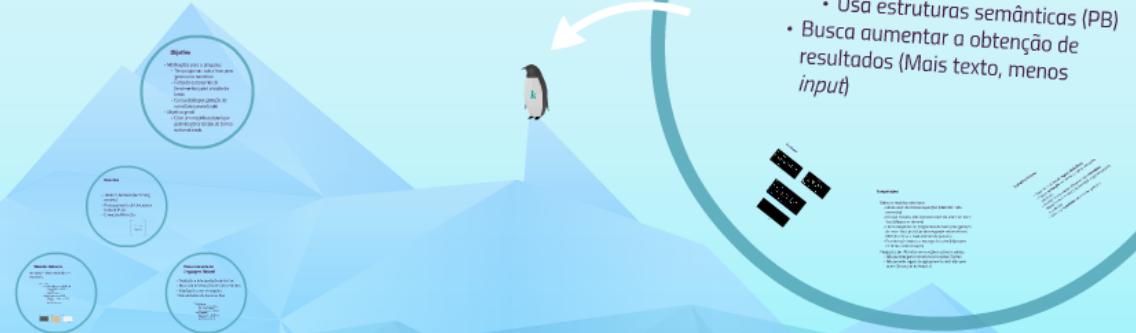
- [1] - Obtida do software Twine que pode ser baixado em <http://twinery.org/>
- [2] - Obtida do site: <http://brightspiral.com/tracery/>
- [3] - Obtidas através do software Wide Ruled 2: <https://games.soe.ucsc.edu/project/wide-ruled>
- [4] a [9] - Imagens do kNarrator

kerberpro@gmail.com



kNarrator

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



Objetivo

- Motivações para a pesquisa:
 - Tempo que um autor leva para gerar uma narrativa
 - Falta de autonomia de ferramentas para criação de texto
 - Curiosidade por geração de narrativas procedurais
- Objetivo geral:
 - Criar um modelo autoral que permita gerar textos de forma automatizada.

Assuntos

- Modelos Autorais (Authoring models)
- Processamento de Linguagem Natural (PLN)
- O modelo kNarrator

Áreas de estudo

- Linguística (PLN)
- Narrativa (Autoral)

Áreas de estudo

- Linguística (PLN)
- Narrativa (Autoral)

Modelos Autorais

Buscam facilitar o trabalho dos escritores.

Classificação proposta:

- **Não baseados em enredo (N-PB)**
 - Não voltados a estrutura narrativa
 - Regras simples
- **Baseados em enredo (PB)**
 - Modelos voltados a estrutura narrativa
 - Regras complexas

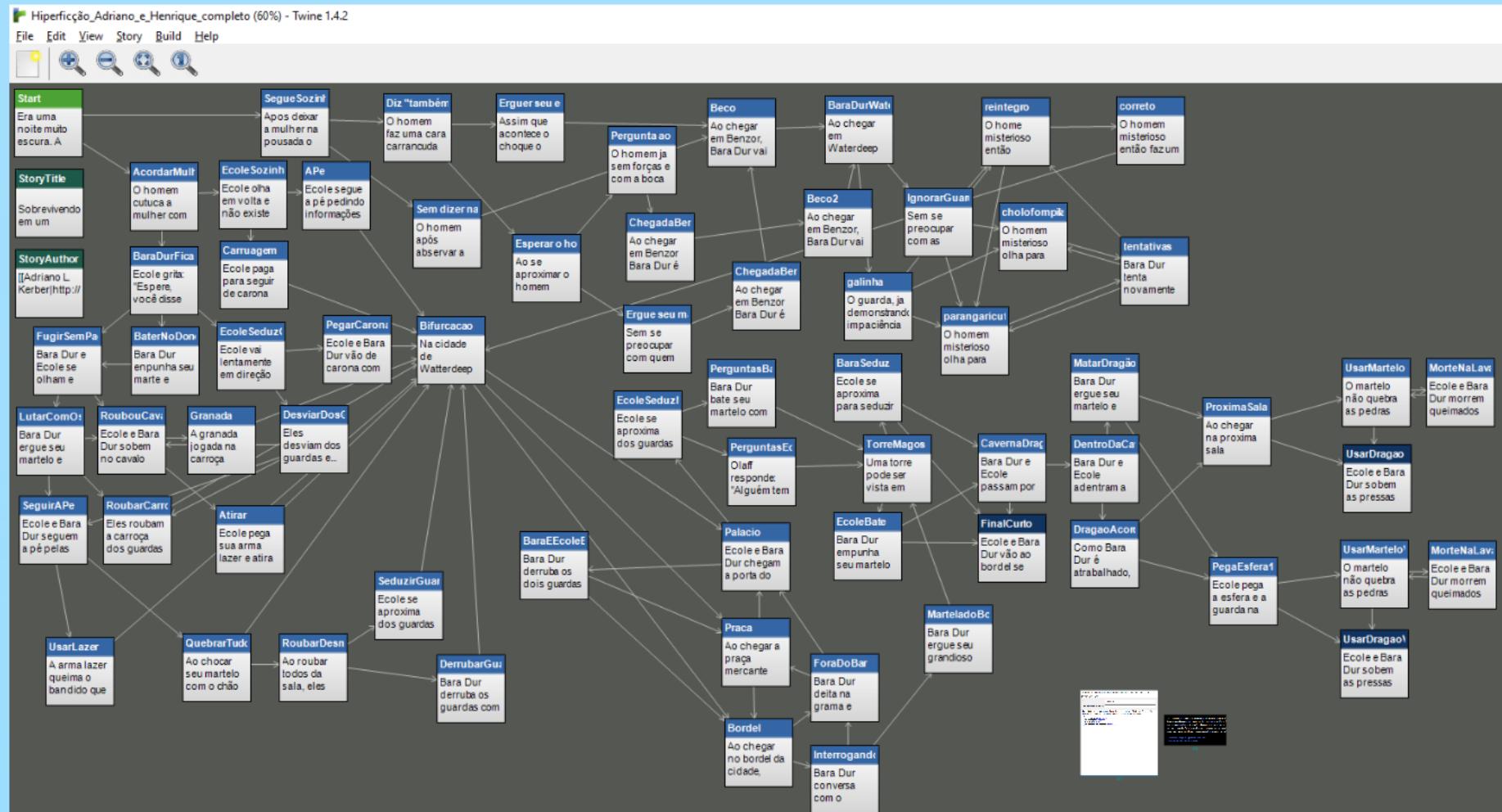


Classificação proposta:

- **Não baseados em enredo (N-PB)**
 - Não voltados a estrutura narrativa
 - Regras simples
- **Baseados em enredo (PB)**
 - Modelos voltados a estrutura narrativa
 - Regras complexas

Twine

Não baseado em enredo



[1]



Tracery

Não baseado em enredo

The screenshot shows the Tracery application interface. On the left, there is a grammar editor window titled "tinygrammar". It contains three sections: "name" with tokens "Bertram", "Arabella", and "Cecil" followed by a plus sign; "occupation" with tokens "baker", "wizard", and "soldier" followed by a plus sign; and "origin" with the template "#name# the #occupation#" followed by a plus sign. Below these sections is a "new symbol" button. At the top of the editor are buttons for "tinygrammar 17/03/2016 16:58:30", "JSON", "undo", and "show colors". To the right of the editor is a "reroll" panel containing a list of generated strings: "Cecil the baker", "Bertram the wizard", "Cecil the baker", "Bertram the baker", and "Cecil the baker".



Wide Ruled 2

Basedo em enredo

MurderMysteryInteractive.wr2 - Wide Ruled 2.0, Release 3.61

File

Characters

Environments

Plot Point Types

Goals and Plot Fragments

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

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

Crime Info
Murderer
Murder Solved

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

New Edit ... Delete

New Edit ... Delete

New ... Edit ... Delete

Interactivity Actions ...

Generate Story

Show Status Messages

[3]



Processamento de Linguagem Natural

- Tradução e interpretação de textos
- Busca de informações em documentos
- Interface homem-máquina
- Sumarização de documentos

Classificação proposta:

- **Normalização**
 - Algoritmos para identificação de grupos de palavras (Stemming ou Lemmatization)
- **Desambiguação**
 - Algoritmos para classificação e etiquetagem de palavras (Part-Of-Speech tagging)

Classificação proposta:

- **Normalização**
 - Algoritmos para identificação de grupos de palavras
(Stemming ou Lemmatization)
- **Desambiguação**
 - Algoritmos para classificação e etiquetagem de palavras
(Part-Of-Speech tagging)

O modelo kNarrator

- Modelo autoral híbrido
 - Autor controla a narrativa (N-PB)
 - Usa estruturas semânticas (PB)
- Busca aumentar a obtenção de resultados (Mais texto, menos *input*)



Comparações

Todos os modelos anteriores:

- Necessitam de sintaxe específica (kNarrator não necessita)
- Em sua maioria, não oferecem controle direto no texto [final] (não autorável)
- Usam templates ou fragmentos de texto para geração do texto [final]. já o kNarrator expande seu texto com PUI (Funciona no nível abstrato de palavra).

Estruturação textual é encargo do autor (kNarrator controla o estruturação)

Próteicos do kNarrator em relação a outros modelos:

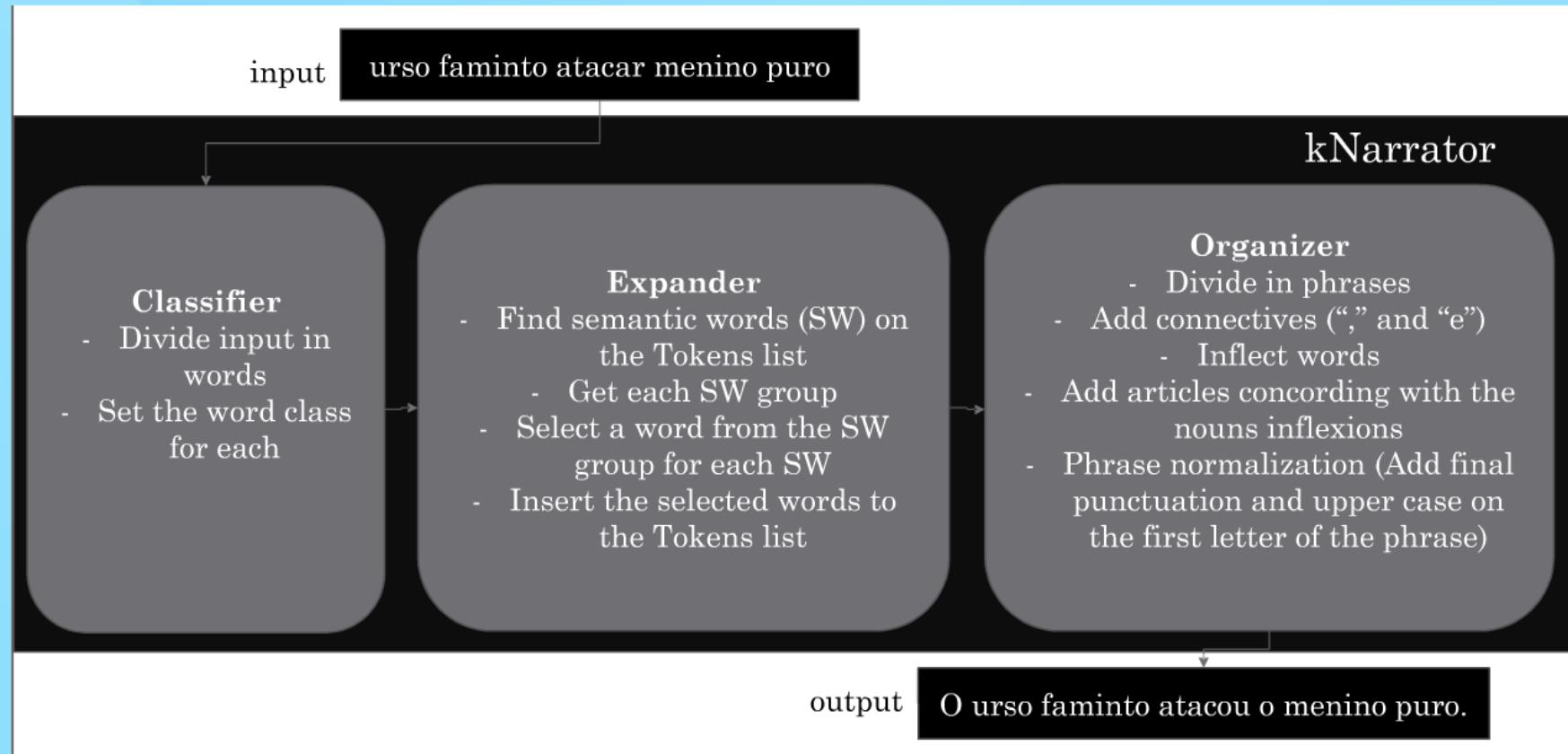
- Não permite gerar normativas bifuncionais (twain)
- Não permite regas de argumentação definidas pelo autor [Troceny, Wide Ruled 2]

Trabalhos Futuros

- Criar um sistema de **regras semânticas**
- Utilizar **ontologia** em conjunto com o dicionário
- Implementar que o modelo Expender criá **sentenças** relacionadas com nível de discussão e generalização de argumentos.
- Adicionar **coerência** entre textos gerados



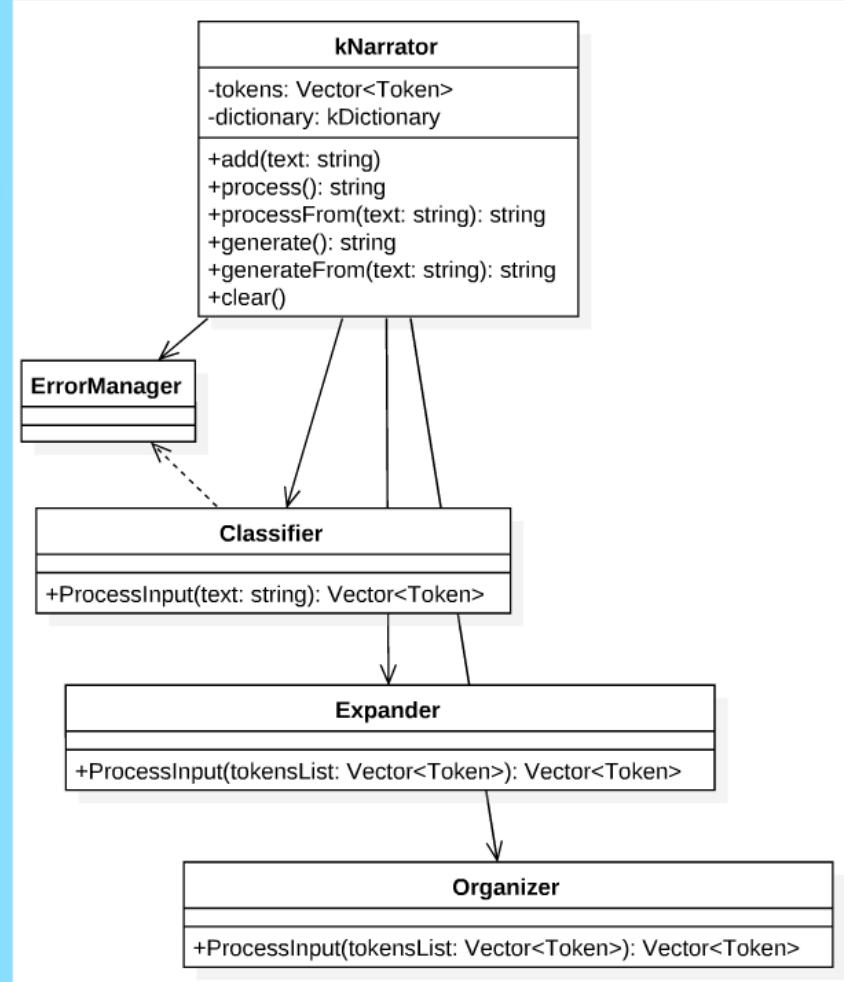
Funcionamento



[4]

Módulos

Três módulos principais e um auxiliar



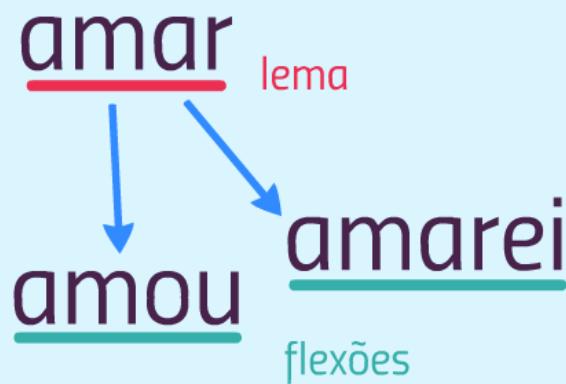
[5]

Classifier

Classifica as palavras do *input* em classes de palavras através de um dicionário

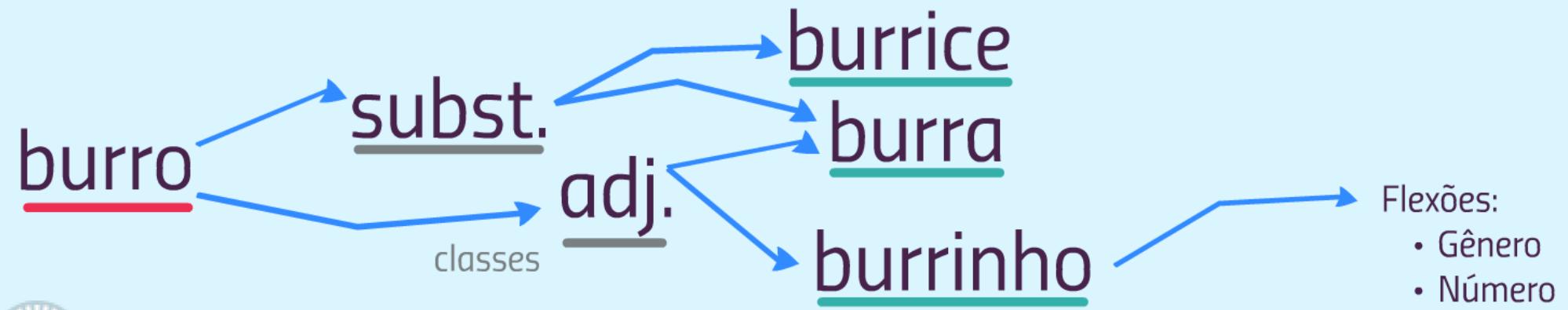
- Recebe input (pseudo-texto)
- Compara as palavras do input com as palavras do **dicionário**
- Cada palavra se torna um Token

Dicionário



O que é salvo no dicionário é o lema da palavra com todas as palavras pertencentes ao grupo (Flexões do lema).

O dicionário usa um banco de dados SQL.



- Gênero
- Número
- Grau

ErrorManager

Palavras não
identificadas são
enviados para este
módulo

(nº linha + id da palavra)

Expander

Insere novas palavras para expandir o texto de *input* com mais detalhes (Módulo opcional)

"Urso atacar homem floresta" gera o resultado:
"Urso grande forte atacar homem floresta denso silencioso".

Onde as palavras "Urso" e "floresta" recebem uma adição de palavras em sua descrição.

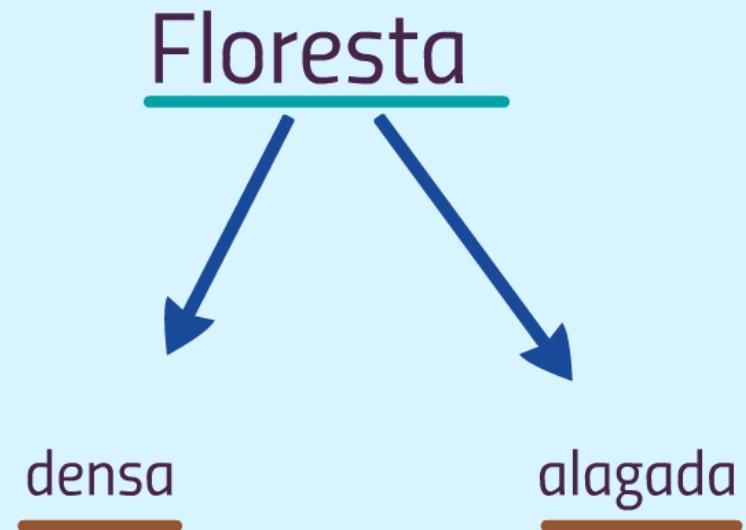
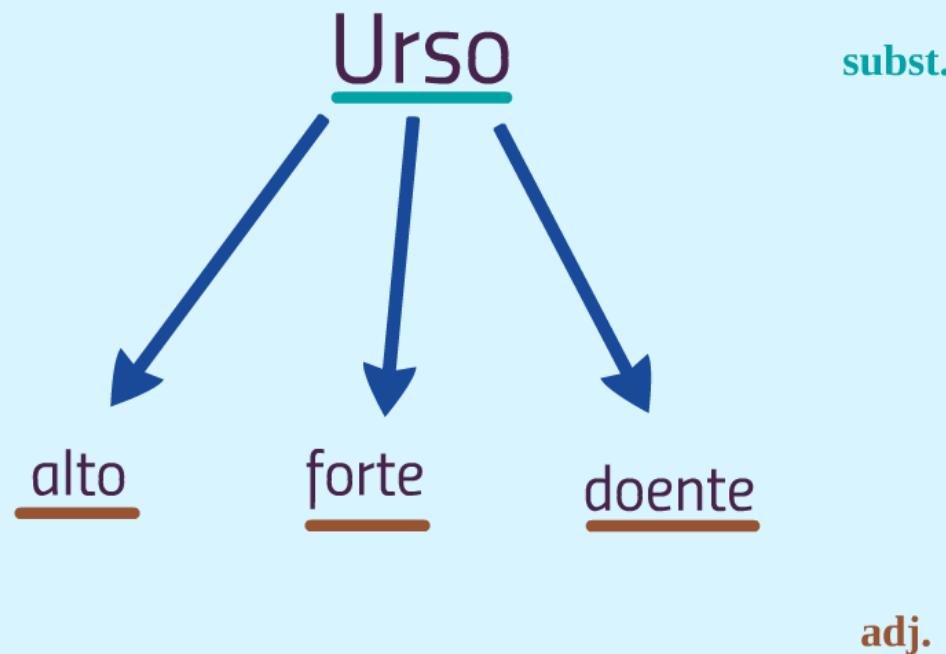
- Recebe a lista de Tokens
- Compara as palavras com as palavras do **dicionário semântico**
- Insere novas palavras para descrever os Tokens selecionados

Dicionário Semântico

Uma estrutura que tem palavras com múltiplos resultados (Chave - Valores)

Estruturas de classificação:

- Personagens
- Lugares



Organizer

Recebe a lista de Tokens e a transforma em texto fluido

- Frases: divide os tokens em frases verbais
- Pontuação: adiciona pontuação ao final das frases
- Concordancia: flexiona Tokens para concordarem entre si
- Conectivos: adiciona ";" e "e" para ligar as palavras
- Artigos: adiciona artigos antecedendo substantivos
- Realização: adiciona letras maiúsculas as palavras que iniciam frases

Parâmetros de controle

Flexiona o texto conforme os parâmetros definidos pelo autor

São quatro parâmetros de controle para forçar as flexões do texto final:

- Gênero
- Número
- Pessoa
- Tempo

Resultados

```
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]

```
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 hurro andará e atacará a criatura.
```

[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]



Input:

Adriano ir aprovar menina ir comer beber dormir Guerreiro andar atacar criatura

[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]

Comparações

Todos os modelos anteriores

- Necessitam de sintaxe específica (kNarrator não necessita)
- Em sua maioria, não oferecem controle direto no texto final (kNarrator oferece)
- Usam *templates* ou fragmentos de texto para geração do texto final, já o kNarrator expande seu texto com PLN (Funciona a nível atômico de palavra)
- Estruturação textual a encargo do autor (kNarrator controla a estruturação)

Fraquezas do kNarrator em relação a outros modelos:

- Não permite gerar narrativas bifurcadas (Twine)
- Não permite regras de agrupamento definidas pelo autor (Tracery, Wide Ruled 2)

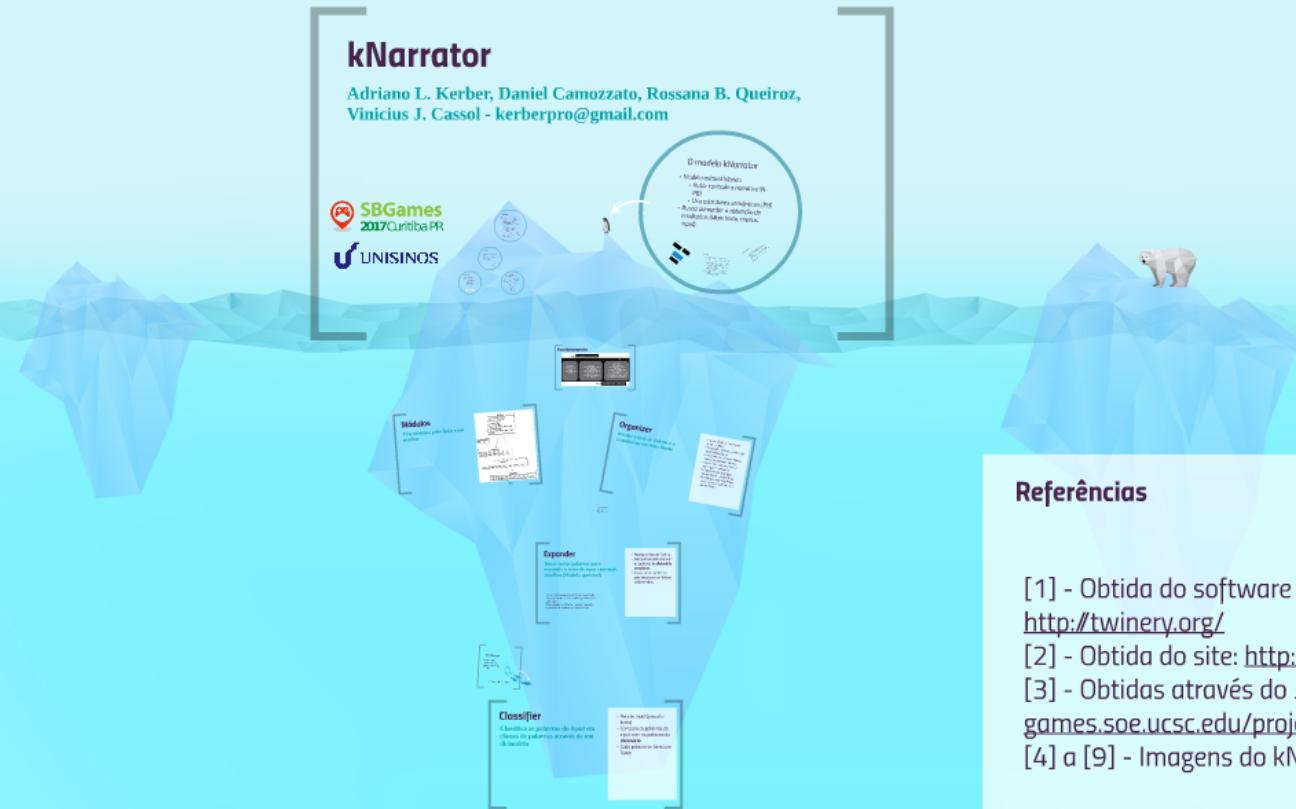
Trabalhos Futuros

- Criar um sistema de **regras simbólicas**
- Utilizar **ontologia** em conjunto com o dicionário semântico
- Permitir que o módulo Expander crie **narrativas** trabalhando com níveis de descrição e gerenciamento de eventos
- Adicionar **coerência** entre textos gerados

kNarrator

Adriano L. Kerber

kerberpro@gmail.com



Referências

- [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



Referências

- [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
2017Curitiba PR

 **UNISINOS**