Processamento de Linguagem Natural

Processamento Básico de Texto

Expressões Regulares e Normalização de Texto

Prof.: Hansenclever Bassani (Hans) hfb@cin.ufpe.br

Site da disciplina: www.cin.ufpe.br/~hfb/pln/

Baseado nos slides do curso de Stanford no Coursera
por Daniel Jurafsky e Christopher Manning.

Tradução: Ygor César Sousa
Revisão: Hansenclever Bassani

UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Processamento de Texto Básico

Expressões Regulares

Expressões Regulares

- Uma linguagem formal para especificar cadeias de caracteres
- Como podemos encontrar qualquer das opções abaixo?
  - woodchuck*
  - woodchucks
  - Woodchuck
  - Woodchucks

*Marmota (Um tipo grande de esquilo)

Expressões Regulares: Disjunções

- Letras dentro dos colchetes []

Padrão	Equivale
[wW]oodchuck	Woodchuck, woodchuck
[1234567890]	Any digit

- Conjuntos em intervalos [A-Z]

Padrão	Equivale	
[A-Z]	Uma letra maiúscula	Drenched Blossoms
[a-z]	Uma letra minúscula	my beans were impatient
[0-9]	Um único dígito	Chapter 1: Down the Rabbit Hole

Expressões Regulares: Negação em Disjunções

- Negações [^Ss]
  - ^ significa negação apenas quando vem primeiro nos []

Padrão	Equivale	
[^A-Z]	Uma letra “not” maiúscula	Oyfn pripetchik
[^Ss]	Não ‘S’ e não ‘s’	I have no exquisite reason
[^e^]	Não ‘e’ e não ‘^’	Look here
a^b	O padrão a ^ b	Look up a^b now

Expressões Regulares: Mais Disjunções

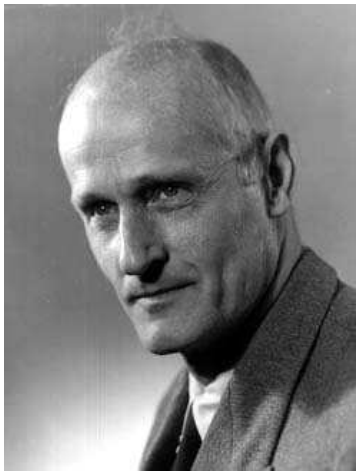
- Woodchuck e Groundhog são sinônimos para Marmota!
- Barra vertical | para disjunção

Padrão	Equivale
groundhog|woodchuck	
yours|mine	yours mine
a|b|c	= [abc]
[gG]roundhog|[Ww]oodchuck	

Expressões Regulares: ? * + .

Padrão	Equivale	
colou?r	Caractere anterior opcional	color colour
oo*h!	0 ou mais do caractere anterior	oh! ooh! oooh! ooooh!
o+h!	1 ou mais do caractere anterior	oh! ooh! oooh! ooooh!
baa+	``	baa baaa baaaa baaaaa
beg.n	Qualquer* caractere	begin begun begun beg3n

*exceto quebra de linha



Stephen C Kleene

Kleene *, Kleene +

Expressões Regulares: Ancoras: ^ $

Padrão	Equivale	
^[A-Z]	Inicia com letra maiúscula	Palo Alto
^[^A-Za-z]	Inicia com algo que não seja letra maiúscula ou minúscula	1 “Hello”
\.$	Finaliza com ponto	The end.
.$	Finaliza com algum caractere	The end? The end!

Exemplo

- Encontrar todas as instâncias da palavra “the” em um texto.

  the

  Perde exemplos com letras maiúsculas

  [tT]he

  Retorna incorretamente other ou theology

  [^a-zA-Z][tT]he[^a-zA-Z]

Tipos de Erros

- O processo que acabamos de vicenciar foi baseado na resolução de dois tipos de erros
  - Encontrar instâncias que não deveriam ser encontradas (there, then, other)
    - Falsos positivos (Tipo I)
  - Não encontrar instâncias que deveriam ser encontradas (The)
    - Falso negativo (Tipo II)

Tipos de Erros

- Em PLN nós estamos sempre lidando com esses tipos de erros.
- Reduzir a taxa de erro para uma aplicação sempre envolve dois esforços antagônicos:
  - Melhorar acurácia ou precisão (minimizando falsos positivos)
  - Melhorar cobertura (minimizando falsos negativos).

Sumário

- Expressões Regulares desempenham uma função surpreendentemente ampla
  - Sequencias sofisticadas de expressões regulares são sempre o primeiro modelo para qualquer processamento de texto
- Para muitas tarefas difíceis, nós usamos classificadores da aprendizagem de maquina
  - Expressões regulares são usadas como características nos classificadores
  - Pode ser bem útil na identificação de generalizações

Processamento de Texto Básico

Expressões Regulares

Processamento de Texto Básico

Separação de Tokens

Normalização de Texto

- Todas atividades de PLN precisam fazer normalização de texto:
  1. Segmentação/tokenizing de palavras em texto em corrido
  2. Normalização do formato das palavras
  3. Segmentação de sentenças no texto em execução

Quantas palavras?

- I do uh main- mainly business data processing
  - Fragmentos, pausas preenchidas
- Seuss's cat in the hat is different from other cats!
  - **Lema**: palavras com mesmo tronco (forma canônica)
    - cat and cats = mesmo lema
  - **Forma**: Flexões de um lema
    - cat and cats = mesmo lema, formas diferentes
  - **Morfema**: Elemento da linguagem que carrega significado
    - Desumidificar: três morphemas "des", "umidi", "ficar"

Quantas palavras?

they lay back on the San Francisco grass and looked at the stars and their

- **Tipo**: um elemento do vocabulário.
- **Token**: uma instância daquele tipo no texto em execução.
- Quantos?
  - 15 tokens (ou 14)
  - 13 tipos (ou 12) (ou 11?)

Quantas palavras?

N = número de tokens

V = vocabulário = conjunto de tipos

$|V|$ é o tamanho do vocabulário

Church and Gale (1990): $|V| < O(N^{½})$

	Tokens = N	Type= |V|
Switchboard phone conversations	2.4 milhões	20 mil
Shakespeare	884,000	31 mil
Google N-grams	1 trilhões	13 mil

Separação de Tokens Simples no UNIX

- (Inspirado em “UNIX para Poetas” de Ken Church.)
- Dado um arquivo de texto, devolva os tokens e suas frequências

```
tr -sc 'A-Za-z' '\n' < shakes.txt
    | sort
    | uniq –c
```

Mudar cada não-alfa para nova linha

Ordenar em ordem alfabética

Juntar e contar cada tipo

```
1945 A
  72 AARON
  19 ABBESS
   5 ABBOT
... ...
```

```
25 Aaron
 6 Abate
 1 Abates
 5 Abbess
 6 Abbey
 3 Abbot
....  ...
```

Primeiro Passo: Separação em Tokens

tr -sc ’A-Za-z’ ’\n’ < shakes.txt | head

THE
SONNETS
by
William
Shakespeare
From
fairest
creatures
We
...

Segundo Passo: Ordenação

```
tr -sc ’A-Za-z’ ’\n’ < shakes.txt | sort | head

A
A
A
A
A
A
A
A
A
A
...
```

Por fim: Contagem

- Unir maiúsculas e minúsculas

```
tr ‘A-Z’ ‘a-z’ < shakes.txt | tr –sc ‘A-Za-z’ ‘\n’ | sort | uniq –c
```

- Ordenar as contagens

```
tr ‘A-Z’ ‘a-z’ < shakes.txt | tr –sc ‘A-Za-z’ ‘\n’ | sort | uniq –c | sort –n –r
```

```
23243 the
22225 i
18618 and
16339 to
15687 of
12780 a
12163 you
10839 my
10005 in
8954  d
```

O que aconteceu aqui?

Questões na Separação de Tokens

- Finland's capital → Finland, Finlands, Finland's *?*
- what're, I'm, isn't → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?
- San Francisco → one token or two?
- m.p.h., PhD. → ??

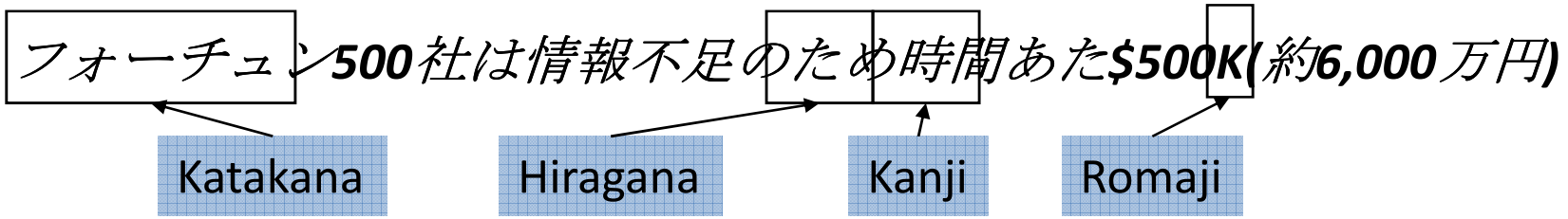
Separação de Tokens: Questões de Idioma

- Francês
  - ***L'ensemble*** → um token ou dois?
    - ***L*** ? ***L'*** ? ***Le*** ?
    - Queremos que ***l'ensemble*** seja equivalente a ***un ensemble***

- Substantivos compostos em Alemão não são segmentados
  - ***Lebensversicherungsgesellschaftsangestellter***
  - 'Funcionário de uma companhia de seguros de vida'
  - Recuperação de informação em Alemão precisa de um **separador de compostos**

Separação de Tokens: Questões de Idioma

- Chinês e Japonês não têm espaços entre palavras:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
  - Sharapova now lives in US southeastern Florida
- Ainda mais complicado em Japonês, por ter múltiplos alfabetos misturados
  - Datas/Quantidades em múltiplos formatos



Usuário final pode expressar uma pergunta inteiramente em hiragana!

Separação de Tokens em Chinês

- Também chamado de **Separação em Segmentos**
- Palavras em Chinês são compostas de caracteres
  - Caracteres geralmente representam 1 sílaba e 1 morfema.
  - Uma palavra tem em média 2.4 caracteres.
- Algoritmo inicial padrão de segmentação:
  - Maximum Matching (também conhecido como Greedy)

Maximum Matching
Algoritmo de Separação em Segmentos

- Dada uma lista de palavras de Chinês e uma cadeia de caracteres.

1) Comece com um apontador no começo da cadeia de caracteres
2) Encontre a palavra mais longa no dicionário que corresponde a sequencia de caracteres a partir do apontador
3) Mova o ponteiro através da palavra
4) Volte ao passo 2

Segmentação Max-match: Ilustração

- Thecatinthehat — the cat in the hat
- Thetabledownthere — the table down there / theta bled own there
- Geralmente não funciona em Inglês!
- Mas funciona surpreendentemente bem em Chinês
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
- Algoritmos modernos de segmentação probabilística são ainda melhores

Processamento de Texto Básico

Separação de Tokens

Processamento de Texto Básico

Normalização de Palavras e Stemming

Normalização

- Necessidade por normalizar termos
  - Recuperação de Informação: texto indexado & termos de busca precisam ter a mesma forma.
    - Queremos combinar ***U.S.A.*** e ***USA***
- Nós implicitamente definimos classes equivalentes de termos
  - e.g., deletando pontos em um termo
- Alternativa: expansão assimétrica:
  - Enter: ***window*** Search: ***window, windows***
  - Enter: ***windows*** Search: ***Windows, windows, window***
  - Enter: ***Windows*** Search: ***Windows***
- Potencialmente mais poderoso, mas menos eficiente

Case Folding

- Aplicações como IR: transformam todas as letras em minúsculas
  - Já que usuários tendem a usar minúsculas
  - Possível Exceção: maiúsculas no meio da sentença?
    - e.g., ***General Motors***, ***Fed*** vs. ***fed***
    - ***SAIL*** vs. ***sail***
- Para análise de sentimentos, MT, Extração de informação
  - Ter letras maiúsculas e minúsculas é útil(***US*** versus ***us*** é importante)

Lemmatização

- Reduzir inflexões ou formas variantes para a forma base
  - *am, are, is → be*
  - *car, cars, car's, cars' → car*
- *the boy's cars are different colors → the boy car be different color*
- Lemmatização: tem que encontrar forma base correta no dicionário
- Tradução automática
  - Espanhol quiero ('I want'), quieres ('you want') mesmo lemma que querer 'want'

Morfologia

- **Morfemas**:
  - A menores unidades de significado que fazem palavras
  - **Stems**: Unidades de significado base
  - **Afixos**: Pedaços que aderem à stems
    - Sempre com funções gramaticais

Stemming

- Reduz termos à seus “caules” em recuperação de informação
- *Stemming* é o corte de afixos
  - Dependente de linguagem
  - e.g., ***automate(s), automatic, automation*** todos reduzidos para ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

for exampl compress and compress ar both accept as equival to compress

Algoritmo de Porter
O mais comum Stemmer de Inglês

Step 1a

sses	→ ss	caresses	→	caress
ies	→ i	ponies	→	poni
ss	→ ss	caress	→	caress
s	→ ø	cats	→	cat

Step 1b

(*v*)ing	→ ø	walking	→	walk
		sing	→	sing
(*v*)ed	→ ø	plastered	→	plaster
…				

Step 2 (para stems longos)

ational	→ ate	relational	→	relate
lzer	→ ize	digitizer	→	digitize
Ator	→ ate	operator	→	operate
…				

Step 3 (para stems mais longos)

al	→ ø	revival	→	reviv
able	→ ø	adjustable	→	adjust
ate	→ ø	activate	→	activ
…				

Visualizando Morfologia como um corpo
Por quê só retirar –ing se tiver uma vogal?

(*v*)ing → ø walking → walk
sing → sing

Visualizando Morfologia como um corpo
Por quê só retirar –ing se tiver uma vogal?

(*v*)ing → ø walking → walk
sing → sing

tr -sc 'A-Za-z' '\n' < shakes.txt | grep ’ing$' | sort | uniq -c | sort –nr

1312 King
548 being
541 nothing
388 king
375 bring
358 thing
307 ring
152 something
145 coming
130 morning

548 being
541 nothing
152 something
145 coming
130 morning
122 having
120 living
117 loving
116 Being
102 going

tr -sc 'A-Za-z' '\n' < shakes.txt | grep '[aeiou].*ing$' | sort | uniq -c | sort –nr

Lidar com morfologia complexa as vezes é necessário

- Alguns idiomas necessitam de segmentação complexa de morfemas
  - Turco
  - Uygarlastiramadiklarimizdanmissinizcasina
  - `(comportar-se) como se você estivesse entre aqueles que nós não pudemos civilizar'
  - Uygar `civilized' + las `become'
    + tir `cause' + ama `not able'
    + dik `past' + lar 'plural'
    + imiz 'p1pl' + dan 'abl'
    + mis 'past' + siniz '2pl' + casina 'as if'

Processamento de Texto Básico

Normalização de Palavras e Stemming

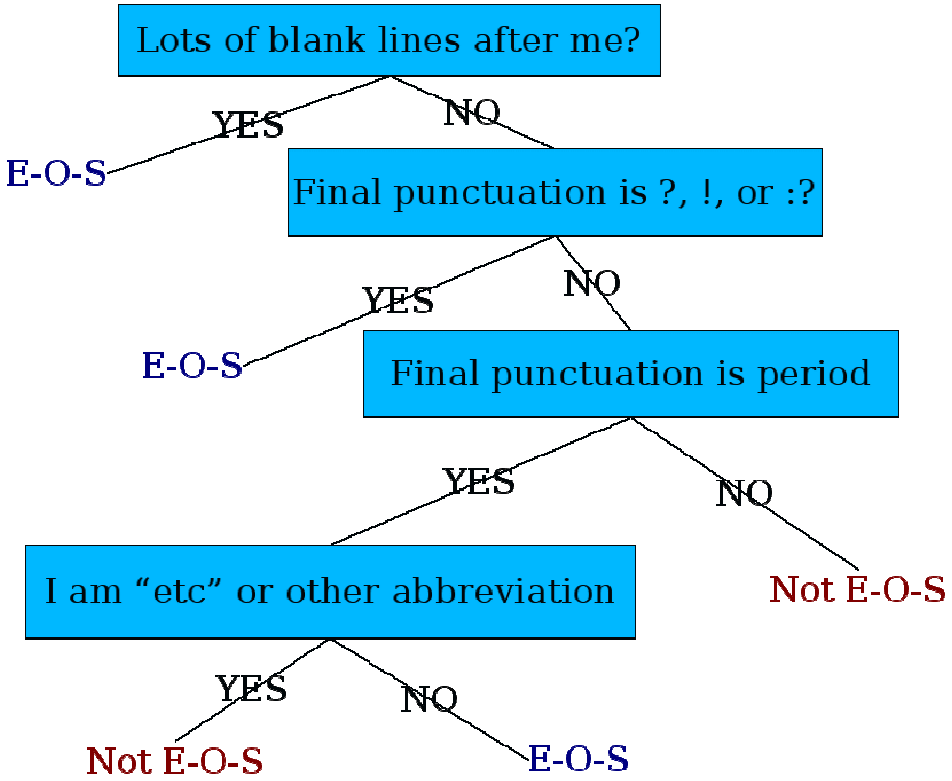
Processamento de Texto Básico

Segmentação de Sentenças e Árvores de Decisão

Segmentação de Sentenças

- !, ? São relativamente não ambíguos
- "." É bastante ambíguo
  - Delimitador de sentença
  - Abreviações como Inc. ou Dr.
  - Números como .02% ou 4.3
- Construir um classificador binário
  - Observa o "."
  - Decide entre FimDeSentença/NãoFimDeSentença
  - Classificadores: baseado em regras escritas à mão, expressões regulares, ou aprendizagem de máquina

Determinar se uma palavra é fim de sentença: a Decision Tree



Características de Árvores de Decisão mais Sofisticadas

- Caso de palavra com “.”: Upper, Lower, Cap, Number
- Caso de palavra depois de “.”: Upper, Lower, Cap, Number
- Características Numéricas
  - Tamanho da palavra terminando com “.”
  - Probabilidade (palavra com “.” ocorrer no fim da sentença)
  - Probabilidade (palavra depois de “.” ocorrer no começo da sentença)

Implementando Árvores de Decisão

- Uma árvore de decisão é baseada em declarações *if-then-else*
- A parte interessante é escolher as características
- Configurar a estrutura é sempre muito difícil de fazer manualmente
  - Configuração manual é apenas viável para características de domínios muito simples
    - Para características numéricas é muito difícil escolher cada limiar
  - Ao invés disso, a estrutura é normalmente aprendida por aprendizagem de máquina com uma base de treinamento

Ávores de Decisão e Outros classificadores

- Nós podemos configurar as características em uma árvore de decisão
- As características podem ser exploradas por qualquer tipo de classificador
  - Regressão logística
  - SVM
  - Redes Neurais
  - etc.

Processamento de Texto Básico

Segmentação de Sentenças e Árvores de Decisão