2016.2

Processamento de Linguagem Natural em Game of Thrones

TRABALHO DA DISCIPLINA TÓPICOS ESPECIAIS EM SISTEMAS INTELIGENTES 2

DANIEL LIMA DOS SANTOS & ANNANDA DANDI DE FREITAS SOUSA

Table of Contents

Proposta

O objetivo do trabalho é aplicar técnicas de processamento de linguagem natural para tentar achar entidades nomeadas, seus tipos e as relações envolvendo essas entidades em um serie de textos sobre episódios da série Game of Thrones.

Deveríamos então gerar uma série de arquivos com essas informações em formato CSV e uma versão final dos textos onde as entidades estariam marcadas.

Também deveríamos utilizar o algoritmo TF-IDF para processar os documentos de forma a responder por buscas, apresentando como resposta os documentos em ordem de relevância.

Limpando o texto

O texto de cada episódio possui várias informações sobre a série, algumas já estruturadas como por exemplo quais mortes acontecem e os personagens presentes. Como consideramos que usar essas informações prontas tornariam o trabalho menos interessante, resolvemos apenas lidar com as informações sobre os acontecimentos nos episódios. Nos arquivos, essas informações são as contidas nas seções “Plot” e “Summary”.

Para começar, removemos todas as linhas em branco. Depois andamos linha por linha, pegando somente as linhas que vem depois de “PlotEdit” e antes de “SummaryEdit”. Em seguida pegamos as linhas entre “SummaryEdit” e “AppearancesEdit” ou “RecapEdit”, já que a seção que vem após o Summary é uma das duas, dependendo do episódio.

Finalmente, notamos que alguns caracteres do texto não eram ASCII, como “…”, “—” e “\u200B” (um caractere invisível). As vezes eram usados três pontos (...) e as vezes (…). Visualmente são idênticos, mas na prática são caracteres diferentes. Para manter o sentido do texto, substituímos eles por seus equivalentes ASCII ou retiramos quando apropriado.

A lista de substituição foi:

UTF-8	**ASCII**
—	-
…	...
’	'
’	'
–	-
—	-
“	"
”	"
\u200B (caractere não visível)	(nada)

Encontrando entidades

A ideia central das regras usadas foi que o nome de entidades geralmente está escrito com letras maiúsculas no meio do texto. Para facilitar a busca, realizamos um tokenização do texto, através da biblioteca nltk.

Um exemplo, com as entidades marcadas com negrito:

*"A short time later, **Littlefinger** meets with **Sansa**, offering her a place on his ship that will take him from the capital (...). "*

Além de entidades de uma palavra, algumas possuem nomes compostos. Ex:

*"**Jon Snow** learns about wargs from the **Free Folk**"*

Para reconhece-las, consideramos uma sequência de letras maiúsculas como uma só entidade.

Outro caso é a presença de palavras com letras minúsculas no meio do nome da entidade, como "of" e "the":

*"**Lord Commander Jeor Mormont of the Night's Watch** leads the few survivors of the slaughter at the **Battle of the Fist of the First Men** south in hope of reaching the **Wall**."*

Consideramos então as palavras "of", "the", " 's " e " ' " como parte do nome da entidade caso elas estejam presentes entre duas palavras com maiúsculas. O token " 's " é aceito para pegar casos como:

*"At **Craster's Keep**, **Locke** scouts the keep for the party of the **Night's Watch**".*

A aspas simples " ' " só é aceita para pegar casos em que houve erro ortográfico na escrita do " 's ".

Para realizar um filtro, usamos um POS Tag para adicionar as classes gramaticais a cada token. Então analisamos as entidades e recusamos qualquer uma que não possua pelo menos um token com as classes 'NN', 'NNP', 'NNS' e 'NNPS', que representam substantivos.

Depois disso, ainda tínhamos muitas entidades erradas devido ao fato de que no início de frases as palavras têm letras maiúsculas mesmo que não sejam nomes:

*"**Enduring Lord Walder Frey** 's insults directed at him (…) "*

Isso também acontece caso ocorra um diálogo no meio da frase. No começo do diálogo, a palavra geralmente também está com letra maiúscula independentemente de ser ou não uma entidade nomeada:

*“**Daenerys** turns (...) and bluntly says "**A** dragon is not a slave," (...)”*

Para resolver isso, nos marcamos todas as entidades que estão no começo de diálogos ou frases e revisamos elas antes de considerar elas realmente entidades válidas. Essa revisão é feita olhando a primeira palavra de cada entidade. A ideia é que caso ela seja uma palavra comum da língua, ela provavelmente apareceu sem a letra maiúscula em outro lugar do texto. Caso isso seja verdade, nos eliminamos a entidade.
Depois de fazer isso, vimos que várias entidades que começavam com “The” ou títulos como “Lord” tinham sido excluídas. Nós então passamos a aceitar entidades que começavam com títulos e “The” independentemente dessa verificação.

Finalmente, nós recusamos qualquer entidade com menos de 2 caracteres.

Agrupando e Classificando

Tendo então uma lista de entidades, temos que uma mesma entidade pode ser referenciada de muitas maneiras. Nós decidimos que o nome “canônico” (ou “real”) de uma entidade pessoa será sempre o nome mais a família dela, se houver. Ou seja, o nome canônico de “Arya” é “Arya Stark”.

Executamos um passo a passo para aplicar tanto classificação quanto agrupamento simultaneamente. Ele depende de conhecimento prévio de títulos, casas e formas de se referir a lugares. Para obtê-lo mapeamos títulos usados geralmente como “Princess”, “Lady”, “Commander”, “Lord Commander of”, identificamos que casas sempre possuem só um nome e são referenciadas por “House” seguido desse nome e vimos que lugares são geralmente precedidos de “return from”, “arrive at”, “back to”, “visit to”, e outras expressões.

Dado uma lista de títulos, casas e lista de expressões que indicam lugar, realizamos os seguintes passos:

Se uma entidade começa com um título, o nome canônico é sem o título e a entidade é uma pessoa.

Ex: “King Robert”, “Lord Snow”, “Lady Catelyn”

Procura por “House” mais algo. Esses “algo” são consideradas casas.

Ex: “House Stark”, “House Targaryen”, “House Reed” geram as casas “Stark”, “Targaryen” e “Reed”.

Se houver uma entidade que só tenha duas palavras e termina com uma casa, é uma pessoa.

Ex: “Eddard Stark”, “Daenerys Targaryen”, “Howland Reed”

Para cada uma dessas entidades, se existe outra entidade que possui só uma palavra e é igual a primeira palavra dela, elas são a mesma entidade.

Ex: As entidades anteriores permitem supor que “Eddard”, “Daenerys” e “Howland” encontrados sozinhos se referem a “Eddard Stark”, “Daenerys Targaryen”, “Howland Reed”.
O nome canônico deles é então o nome com duas palavras.

Para cada entidade ainda não marcada, procuramos se as palavras anteriores a ela são uma expressão que indica lugar.
Se ela estiver depois de pelo menos 5 expressões de lugar, então é classificada como lugar.

Qualquer entidade não marcada é classificada como “other”, que é o equivalente a desconhecido.

Resultados

Encontramos 13877 entidades, sendo 1273 únicas.

Analisando manualmente, em torno de 150 estão erradas.

Agrupando, temos 1073.

Das 13877 entidades, obtivemos quantas de cada tipo foram classificadas:

Classe	Número de Classificados	Porcentagem
other	5277	39%
person	6977	50%
place	1529	11%

Se considerarmos as entidades únicas,

Classe	Número de Classificados	Porcentagem
other	891	73%
person	329	25%
place	31	2%

Analisando os números, podemos ver que mais da metade das entidades foram classificadas. Porém, ao considerar as entidades únicas, a maioria não foi classificada.

Isso é um reflexo do fato de que existe um pequeno número de entidades nomeadas únicas que aparece muito no texto e um grande número que aparece apenas uma vez no texto.
Como a maior parte dessas entidades que aparecem muito são pessoas, elas geram um número absoluto grande de entidades marcadas.

Encontrando relações

A nossa busca por relações é bem simples. Se baseia na ideia de que uma relação aparecerá no texto geralmente quando existir um verbo entre duas entidades.

Com o POS Tag, é possível obter informações de tokens que podem ser verbos. Para cada entidade, verificamos se existe um token das classes "VB", "VBG", "VBZ", "VBN". Se sim, criamos uma tripla em que a relação é o verbo e as entidades são as que estão em volta do verbo.

A seguir alguns exemplos de tuplas achadas:

Resultados bons:

(Jon Stark, leave, Night 's Watch)
(Cersei Lannister, obey, Robert Baratheon)

Resultados ruins:

(Sandor Clegane, is, Joffrey Baratheon)
(Gregor Clegane, burying, Gregor Clegane)

Resultados

2322 triplas encontradas, com 998 relações únicas.

Marcando Entidades

Para marcar a entidades encontradas no texto, a ideia inicial é fazer apenas uma substituição do nome original das entidades para uma tag xml que conteria a informação do nome canônico e do tipo da entidade.

O formato escolhido foi esse:

```
<entidade name="nome canonico" type="classificação">
        nome original
</entidade>
```

Para evitar gerar erros, tivemos que resolver dois problemas. O primeiro é que substituir “Tyrion” e depois substituir “Tyrion Lannister” no texto acabaria não substituindo nada da segunda, por que as aparições da primeira entidade na segunda fariam a segunda ter formato inválido:

Ex: Após substituir “Tyrion” no texto “Tyrion Lanninster” teríamos:

```
<entidade canonico="Tyrion Lanninster" tipo="person">
        Tyrion
</entidade>
Lanninster
```

Logo, uma substituição seguinte por “Tyrion Lanninster” iria substituir o texto do nome canônico e não o do texto. Para resolver isso, ordenamos as entidades por tamanho de tokens, da maior para a menor. Assim garantimos que “Tyrion Lanninster” seja sempre substituído primeiro.

Ex:

```
<entidade canonico="Tyrion Lanninster" tipo="person">
        Tyrion Lanninster
</entidade>
```

Mesmo assim, substituindo “Tyrion” em seguida resultaria em:

```
<entidade canonico="
        <entidade canonico="Tyrion Lanninster" tipo="person">
                Tyrion
        </entidade>
        Lanninster" tipo="person">
        <entidade canonico="Tyrion Lanninster" tipo="person">
                Tyrion
        </entidade>
        Lanninster
</entidade>
```

Para resolver esse problema, atribuímos a cada entidade um id (que na prática é a ordem em que ela vai ser substituída) e substituímos a entidade no texto por “<id>”. Assim, não corre o risco da próxima entidade substituir algo na anterior.

Após substituir todas as entidades por seus ids, substituímos todos os ids por suas entidades envoltos nas tags.

Exemplo de um parágrafo com entidades marcadas:

<en name='Jon Snow' type='other'>**Jon Snow**</en> is still separated from the <en name='Night 's Watch' type='place'>**Night 's Watch**</en> scouting group led by <en name='Qhorin Halfhand' type='other'>**Qhorin Halfhand**</en>.

TF-IDF

Aplicamos o TF-IDF padrão. Não usamos SVD para diminuir a dimensionalidade.

Pegamos todos os tokens do texto limpo e depois retiramos os seguintes tokens:
" . ", " , ", " ' ", " : ", " ! ", " ; ", " ? ", "] ", " [", " & ", " - ", " … ", ") ", " (", " `` ", " -- ", " 's ", " i.e ", " 'd ", " 'm ", " '' "," 've ", " 're ", " n't " e " 'll ".

Ao fim, obtivemos 8549 tokens. Como a quantidade de documentos é 56, a matrix TD-IDF ficou 8549 por 56.

Resultados:

Os resultados de uma pesquisa são mostrados em ordem do mais relevante para o menos relevante.

Buscando por “Eddard Stark”, temos:

1. season_1_the_wolf_and_the_lion
2. season_1_winter_is_coming
3. season_1_you_win_or_you_die
4. season_1_cripples,_bastards_and_broken_things
5. season_1_lord_snow

Os resultados fazem sentido, pois esse personagem é o personagem principal da primeira temporada e depois morre ao fim dela.

Buscando por “Battle of the Blackwater” temos:

1. season_3_valar_dohaeris
2. season_3_kissed_by_fire
3. season_3_walk_of_punishment
4. season_2_blackwater
5. season_3_and_now_his_watch_is_ended.txt

A batalha acontece no final da segunda temporada, e só depois, na terceira, é mencionada como sendo um evento importante. Nas temporadas seguintes, esse evento perde importância.