EP 1– Etiquetagem Morfossintática

Entrega: 26/10/2025

Este EP pode ser feito em duplas

O objetivo deste exercício programa (EP) é exercitar os conhecimentos aprendidos em sala de aula desenvolvendo programas para etiquetagem morfossintática. No inglês, esta atividade se chama *Part-of-Speech Tagging*, ou simplesmente *PoS tagging*. Vamos exercitar tanto a parte de treinamento quanto a etiquetagem propriamente dita.

Nesse contexto, é fundamental termos um conjunto de dados.

Dados

Para desenvolver este trabalho, utilizaremos o dataset Porttinari (PORTuguese Treebank), o qual está disponível em:

`https://github.com/UniversalDependencies/UD_Portuguese-Porttinari/tree/master` .

Os arquivos que temos interesse são os do formato CoNLL-U (Universal Dependencies) que estão disponíveis na raiz do repositório[1]. Atualmente há três arquivos:

- `pt_porttinari-ud-dev.conllu`;
- `pt_porttinari-ud-test.conllu`;
- `pt_porttinari-ud-train.conllu`.

Como os nomes sugerem, eles já são um conjunto de dados separados em treinamento (train), validação (dev), e teste (test). A tarefa é aprender as regras com o conjunto de treinamento e avaliar no conjunto de teste, utilizando o conjunto de validação para ajustar hiperparâmetros (mais abaixo).

Existem três tipos de linha em um arquivo CoNLL-U:

1. *Linhas de Comentário:* Linhas que começam com o símbolo de cerquilha (ou jogo-da-velha) (#). Elas contêm metadados (informações sobre a sentença). Por exemplo:

   `# sent_id =` Um identificador único para a sentença.

   `# text =` O texto original da sentença, antes de ser dividido em palavras (tokens).

2. *Linhas de Palavra (Token)*: Cada linha descreve uma única palavra ou símbolo de pontuação e é dividida em 10 colunas separadas por tabulação (TAB).

[1] O Formato CoNLL-U está definido em `https://universaldependencies.org/format.html`.

3. *Linhas em Branco:* Uma linha vazia separa uma sentença da outra.

Para facilitar a leitura sugere-se que o arquivo conllu seja visualizado no formato de planilha, como se fosse um arquivo no formato tsv (ou seja, um csv com tab como separador). Para ilustrar, veja a segunda sentença do conjunto de validação:

35	# sent_id = FOLHA_DOC000097_SENT054								
36	# text = Viver a vida real, em vez de ficar com o celular na mão.								
37	1	Viver	viver	VERB	_	VerbForm=Inf	0	root	0:root	_
38	2	a	o	DET	_	Definite=Def|Gender=Fem|Number=Sing|PronType=Art	3	det	3:det	_
39	3	vida	vida	NOUN	_	Gender=Fem|Number=Sing	1	obj	1:obj	_
40	4	real	real	ADJ	_	Number=Sing	3	amod	3:amod	SpaceAfter=No
41	5	,	,	PUNCT	_	_	7	punct	7:punct	_
42	6	em	em	ADP	_	_	7	case	7:case	_
43	7	vez	vez	NOUN	_	Gender=Fem|Number=Sing	1	obl	1:obl:em	_
44	8	de	de	ADP	_	_	9	mark	9:mark	_
45	9	ficar	ficar	VERB	_	VerbForm=Inf	7	acl	7:acl:de	_
46	10	com	com	ADP	_	_	12	case	12:case	_
47	11	o	o	DET	_	Definite=Def|Gender=Masc|Number=Sing|PronType=Art	12	det	12:det	_
48	12	celular	celular	NOUN	_	Gender=Masc|Number=Sing	9	obl	9:obl:com	_
49	13-14	na	_	_	_	_	_	_	_	_
50	13	em	em	ADP	_	_	15	case	15:case	_
51	14	a	o	DET	_	Definite=Def|Gender=Fem|Number=Sing|PronType=Art	15	det	15:det	_
52	15	mão	mão	NOUN	_	Gender=Fem|Number=Sing	12	nmod	12:nmod:em	SpaceAfter=No
53	16	.	.	PUNCT	_	_	1	punct	1:punct	_

Notamos que as duas primeiras linhas, que são as linhas 35 e 36 do documento original, são linhas de comentários, contendo o identificador da sentença e o texto. Agora vamos analizar as linhas de tokens, a partir da terceira linha (linha 39 no doc original).

Agora vamos analisar a terceira linha (linha 39 no documento original), em detalhe:

```
1 Viver viver VERB _ VerbForm=Inf 0 root 0:root _
```

As linhas de tokens são divididas em 10 colunas separadas por caracteres de tabulação, definidas desta maneira.

1. ID=1: A palavra Viver é a primeira da sentença.

2. FORM=Viver: A palavra exata encontrada no texto.

3. LEMMA=viver: O lema, ou seja, a forma base ou de dicionário da palavra. No caso de verbos é infinitivo.

4. UPOS=VERB: Viver é classificado como verbo na etiquetagem segundo o Universal Part-of-Speech[2]

5. XPOS=_: Etiqueta morfossintática mais específica para o idioma em questão. É deixada em branco (_) quando se usa o conjunto universal.

6. FEATS=VerboForm=Inf: Uma lista (separada por |) de características morfológicas da palavra (gênero, número, tempo verbal, dentre outros). Neste caso VerboForm=Inf indica que o verbo aparece no infinitivo.

7. HEAD=0: ID da palavra da qual esta palavra depende sintaticamente (seu núcleo). O valor 0 indica que a palavra Viver é a raiz (root) da árvore de dependências, ou seja, o núcleo principal da sentença. A palavra vida (ID=3) tem HEAD=1, indicando que ela depende de Viver (ID=1).

8. DEPREL=root: A relação de dependência sintática (Dependency Relation) entre a palavra atual e sua HEAD. Viver é a raiz (root), já a palavra vida, que possui a palavra Viver como seu HEAD, tem uma relação obj, significando que é o objeto do verbo.

[2]A definição deste conjunto de 17 etiquetas (*tagset*) que será usado neste EP pode ser encontrata em: `https://universaldependencies.org/u/pos/index.html`

9. DEPS=0:root: Um grafo de dependências aprimorado. Geralmente não é utilizado e é deixado em branco (_)

10. MISC=_: qualquer informação extra. A palavra real (ID=4), por exemplo, tem MISC=SpaceAfter=No, indicando que não há um espaço entre a palavra real e o próximo caractere (a vírgula ,).

Para este EP, estaremos interessados apenas nas primeiras quatro colunas.

Note que, na linha 49, ao invés de termos um número temos um intervalo 13-14 que indica um "multi-word token", isto é, um token que é dividido em mais de uma palavra. Neste caso o token `na`, é dividido nos tokens `em` (linha ID=13) e `a` (linha ID=14). Note que, no treinamento e na etiquetagem, estas linhas com tokens multi-palavras deverão ser filtradas.[3]

Agora que vocês sabem como parsear o arquivo, vem a parte bacana: implementar alguns modelos!

Parte A: Previsão de completação

O primeiro moPrevisãodelo que vocês devem implementar é um bi-grama. Com base na palavra atual, o modelo deverá dizer qual é a probabilidade da próxima palavra.

Para isso, você deve filtrar o córpus de treino, mantendo apenas a coluna 2, ou seja, a coluna FORM, que contem o texto, com todas as suas maiúsculas. Você deve extrair esta coluna, precedê-la com uma palavra especial de início de texto (do tipo <`start`> ou <`s`>) e uma palavra de fim (do tipo </`start`> ou </`s`>).

Treinamento. Por se tratar de bigramas do tipo w_1w_2, você deve computar durante a fase de treinamento

$$P(w_2|w_1) = \frac{P(w_1w_2)}{P(w_1)} \approx \frac{C(w_1w_2)}{C(w_1)}$$

onde $C(\texttt{texto})$ é a contagem de ocorrências de `texto` no córpus de treinamento.

Suavização. Teoricamente, o número n_2 de bigramas deve ser igual ao número de unigramas n_1 ao quadrado: $n_s = (n_1)^2$, mas nem todos os bigramas possíveis ocorrem no córpus de teste, então você deve usar algum tipo de suavização para os omo visto em sala de aula, recomenda-se usar alguma variante da Lei de Lidstone, como a suavização de Laplace, ou alguma outra forma de alterar a inicialização das contagens que você julguem mais apropriada.

Execução. Após o treinamento, você deve usar o córpus de teste para prever a palavra seguinte. Para isso você deve escolher a palavra w_2 que maximiza a probabilidade $P(w_2|w_1)$, onde w_1 é a palavra recém vista na sentença, sendo que a primeira palavra sempre deve seguir a palavra especial de início (e.g. <`s`>). Formalmente,

$$w_2 = \arg\max_w P(w|w_1)$$

[3]Se você está achando que boa parte do tempo você vai passar filtrando os dados em vez de treinando os modelos saiba que esta é a ***regra do processamento de dados***, já era verdade antes dos métodos de aprendizado automático, nos aplicativos de Bancos de Dados, e deve continuar valendo.

Você deve tentar prever todas as palavras do córpus de texto, da seguinte maneira. Você deve produzir as sentenças do texto, e baseado na última palavra, usando o método acima, você deve prever a próxima palavra. IIniciando com a palavra especial de início, prever a primeira palavra. Em seguida, usando a palavra correta do texto de teste, você deve prever a seguinte, até o final da sentença (inclusive prevendo a palavra especial de fim de sentença).

Cuidado que podem haver palavras desconhecidas, ou seja, palavras no córpus de teste. Para evitar este problema, verificar se há palavras desconhecidas no córpus de validação. Trocar as palavras desconhecidas por uma etiqueta (<DESC> ou <UNKNOWN>) e medir as probabilidades $P(w|\texttt{<DESC>})$ para cada w no córpus de validação. Esta probabilidade pode ser usada para prever a palavra que segue uma palavra desconhecida no córpus de teste, da mesma forma que as outras palavras são previstas. Isto quer dizer, também, quue sempre erraremos a previsão de uma palavra desconhecida, mas que podemos acertar a previsão da palavra seguite.

Ao final da previsão, teremos dois textos paralelos, o do córpus de teste (padrão ouro) e o das palavras geradas pelo método de bigramas, e iremos avaliar o resultado.

Avaliação. Para a valiação de bigramas, vamos usar três medidas: Precisão, Cobertura e medida-F e acurácia.

As medidas de precisão e cobertura e medida-F são medidas micro, ou seja, são medidas feitas no nível de palavras. Lembrando que, ao final da previsão teremos dois textos, o córpus de teste, padrão ouro, e o córpus de previsões. Para cada palavra w do córpus de teste, teremos as seguintes medida, por palavra:

$$\textit{Precisão(w)} = \frac{\text{N. de vezes que } w \text{ aparece corretamente}}{\text{N. de vezes que } w \text{ aparece no córpus de previsão}}$$

$$\textit{Cobertura(w)} = \frac{\text{N. de vezes que } w \text{ aparece corretamente}}{\text{N. de vezes que } w \text{ aparece no padrão ouro}}$$

JKá a medida-F é computada a partir da precisão e cobertura, como a média narmônica entre eles (ou seja, o produto pela média):

$$\textit{Medida-F(w)} = \frac{2 \cdot \text{Precisão}(w) \cdot \text{Cobertura}(w)}{\text{Precisão}(w) + \text{Cobertura}(w)}$$

Globalmente, podemos apresentar a média das medidas, somando os valores acima para todas as palavras e dividindo pelo número de palavras. Também é interessante mostrara a 10 palavras de melhor medida-F, como as mais fáceis de prever. As piores serão, com certeza, as palavras desconhecidas.

A medida de acurácia é uma medida global, definida por:

$$\textit{Precisão} = \frac{\text{N. de palavras previstas corretamente}}{\text{N. de palavras no córpus de teste}}$$

Lembrando que o córpus de teste e de previsão tem o mesmo tamanho.

Parte B: Treinamento e avaliação de Etiquetagem Morfosintática

O segundo modelo deverá efetivamente etiquetar a sentença, isto é, para cada palavra do córpus de teste, o modelo deverá gerar a etiqueta (UPOS) mais provável. Como visto

em aula, o algoritmo para gerar estas etiquetas é o Algoritmo de Viterbi que leva em consideração a sentença como um todo.

O tagset a ser utilizado é o UPOS, que possui 17 etiquetas[4]. As colunas que iremos usar dos córpus dados são apenas as colunas 2 (texto) e a coluna 4 (etiquets UPOS).

Treinamento. O modelo a ser treinado é da forma $\mu = (A, B, \pi)$. O treinamento deverá ser feito com uma cadeia de Markov de ordem 2, ou seja, para todos os pares XY onde X e Y são etiquetas. No total o número de linhas da matriz de transição de estados, matriz A, é de $17^2 = 289$. No entanto, com apenas as transições da forma XY-YZ são pertidas (i.e., têm probabilidades maiores que 0), precisamos considerar apenas 17 colunas, pois a transição XY-Z pode ser implicitamente considerada como XY-YZ.

Nesse caso, a matriz A seria uma matriz com 289 linhas e 17 colunas, um total de $17^3 = 4913$ células. As linhas da matriz A devem somar 1.

Já a matriz B é a matriz de emissões, que possui uma linha por estado (17 linhas) e uma coluna por palavra do vocabulários, que deverá ser obtido a partir do córpus de treinamento. Portanto, teremos vários milhares de palavras, e esse número de colunas na matriz B. As linhas da matriz B devem somar 1.

Por fim, o vetor π indica a probabilidade de nos encontrarmos num determinado estado, logo será um vetor com apenas 17 posições. As posições do vetor π devem somar 1.

O algoritmo de treinamento será o algoritmo Baum-Welsh, ou backward forward. Para a inicialização, o algoritmo sugere uma inicialização aleatória, respeitando as somas iguais a 1. No entanto, para uma convergência mais rápida, sugerimos as seguintes inicializações:

- Os elementos a_{ij}, onde i agora representa um par de estados $S_iS'_i$, são obtidos pela contagem de ocorrências de $S_iS_{i'}S_j$, dividido pelo número de ocorrências de $S_iS'_i$.
- Os elementos de b_{ik} devem ser inicializados pela contagem do número de vezes que a palavra w_k ocorre com a etiqueta S_i, dividido pelo número de vezes que a etiqueta S_i ocorre no córpus de treinamento.
- Os elementos π_i são obtidos pelo número de vezes que ocorre a etiqueta S_i dividido pelo númeoro total de etiquetas (igual ao número total de palavras) que ocorrem no córpus de treinamento.

Com isso temos o modelo no passo 0, $\mu^0 = (A^0, B^0, \pi^0)$e a partir daí iremos tomar o modelo no passo s, $\mu^s = (A^s, B^s, \pi^s)$, e geraro modelo no passo $s+1$, $\mu^{s+1} = (A^{s+1}, B^{s+1}, \pi^{s+1})$.

Conforme o algoritmo de treinamento visto em sala de aula, a partir dos valores de μ^s, computamos as grandezas (matrizes) $\alpha_i(t), \beta_i(t), \gamma_i(t), p_t(i, j)$, para então usar estes valores para computar μ^{s+1}.

Uma descrição detalhada deste algoritmo pode ser vista no Apêndice A do livro de Jurafsky e Martin, 3a edição, disponível na internet.

Para o encerramento do algoritmo, precisamos calcular a distância de dois modelos, $dist(\mu^s, \mu^{s+1})$, dada por

$$dist(\mu^s, \mu^{s+1}) = \sum_{i,j}(a_{ij}^{s+1} - a_{ij}^s)^2 + \sum_{i,k}(b_{ik}^{s+1} - b_{ik}^s)^2 + \sum_i(\pi_i^{s+1} - \pi_i^s)^2$$

Idealmente, calculamos esta distância no primeiro passo e teremosinamos quando ela fica pequena o suficiente, que deve ser, no mínimo, duas ordens de grandeza a menos que no primeiro passo.

[4] `https://universaldependencies.org/u/pos/index.html`

$$\frac{dist(\mu^s, \mu^{s+1})}{dist(\mu^0, \mu^1)} < \varepsilon$$

e sugerimos algum $\varepsilon < 10^{-2}$.

Suavizações . Como no caso anterior, a inicialização pode conter valores não vistos no córpus de treinamento, e o processo de inicialização deve tratar de não fornecer valores negativos para nenhuma das posições, possivelmente utilizado a Lei de Lidstone.

Palavras desconhecidas. Para tratar de palavras desconhecidas, sugerimos inserir uma nova coluna na matriz B. Utilizando o córpus de validação, substituímos cada palavra desconhecida por <DESC>, e computamos $b_{i<\texttt{DESC}>}$ da mesma forma como foi proposta para a inicialização de B. No final, corrigimos proporcionalmente as probabilidades das demais posições de B para mantermos a soma de suas linhas em 1.

Na hora de execução, antes de rodarmos o algoritmo de Viterbi, trocamos todas as palavras desconhecidas da sentença a ser etiquetada por <DESC>.

Etiquetagem. Uma vez que computamos o modelo $\mu = (A, B, \pi)$, passamos a usar o córpus de treinamento. Para cada sentença do córpus de treinamento, etiquetamos a sentença usando o Algoritmo de Viterbi, conforme visto em sala de aula (O livro de Jurafsky e Martin, no apêndica A, também possui uma descrição deste algoritmo).

O nosso padrão outro será o córpus de teste. O nosso córpus gerado será o mesmo texto, com as sentenças geradas pelo Algoritmo de Viterbi.

Avaliações. Iremos computar as seguintes medidas:

- Para cada etiqueta, vamos computar a precisão, cobertura e medida-f para cada etiqueta, comparando o padrão ouro com as etiquetas computadas.
- Vamos computar as médias de valores de precisão e cobertura para todas as etiquetas. Em seguida, vamos computar uma medida-f para estas medidas médias.
- Computar a acurácia da etiquetagem, calculando o número de etiquetas corretas dividido pelo número de palavras do córpus de teste.
- Listara as 10 papavras mais difíveis de etiquetar.

Instruções para entrega

Você deve submeter via eDisciplinas um zip[5] o seguintes conteúdos:

a) Um diretório com todos os programas desenvolvidos. Espera-se que a linguagem usada seja Python. Incluir aí os córpus filtrados utilizados.

b) Um relatório, em pdf, tanto para a Parte A quanto para a Parte B, com as seguintes informações:

[5]Não entregue arquivos compactados no formato RAR!!! Aceitamos zip ou tgz, apenas.

i. Descrição da implementação, do pré-processamento dos dados e qualquer hipótese relevante assumida na implementação.

ii. Descrição da ativação do programa, ou seja, como rodar os seus programas.

iii. Os seus resultados, com as métricas globais solicitadas.

iv. Comentários e conclusões sobre o exercício.