Processo Seletivo Turing USP - 2021

Case Técnico

Pontuação Total: 100 pontos

Olá, candidate! Seja muito bem-vinde ao processo seletivo Turing USP 2021! Nessa fase do nosso processo seletivo, você terá que resolver um case técnico. São várias perguntas, incluindo um texto em formato de Turing Talks e dois desafios. Envios incompletos serão considerados, mas encorajamos que tente fazer o máximo possível. Os **desafios valem uma pontuação extra** e você pode escolher se quer fazer ou não, mas **deve escolher apenas um dos dois**. Se fizer os dois vamos corrigir apenas um, portanto veja qual faz mais sentido para você, se achar que vale a pena fazer o desafio. Algumas questões são propositalmente desafiadoras, então saiba o que atacar, padawan. O intuito aqui é entender o seu raciocínio e qualidade de código, bem como o seu desenvolvimento no processo.

Lembrete: isso não é uma prova da USP, sinta-se livre para procurar guias e vídeos na internet para te ajudar.

Caso tenha alguma dúvida, é só perguntar para a gente no nosso Discord, onde temos canais específicos para cada parte do case. Se preferir, você pode também entrar em contato pelo nosso e-mail turing.usp@gmail.com, pela nossa página facebook.com/turing.usp ou mandar uma mensagem para algum dos membros que estão listados em cada parte do case.

Tentaremos responder o quanto antes! Além disso, **teremos monitorias nos dias 28/05 (Sexta-feira) das 17h às 19h e 02/06 (Quarta-feira) das 19h às 21h**, que vão ocorrer **online**, no nosso **Discord**.

O case é dividido em três partes, uma de lógica de programação e algoritmos, outra parte de análise de dados e inteligência artificial e uma terceira composta pelo texto no formato de um Turing Talks. Na primeira parte, **lógica de programação e algoritmos**, serão aceitas resoluções em **Python, C, C++ e Java**. Na segunda parte, **análise de dados e inteligência artificial**, serão aceitos os **formatos jupyter notebook (em Python e R).** A última parte, **redação do Turing Talks**, deve ser

enviada em formato **PDF**. Para a parte de lógica e algoritmos, utilizaremos **correção automática** (ou seja, uma primeira correção — antes da correção humana — será feita por um programa de computador). Portanto, é **proibida a utilização de acentos nas palavras na hora de escrever no seu código**, isso porque pode gerar problemas durante a correção automática.

Também é importante lembrar que quanto mais bem comentado o código melhor, pois facilitará o entendimento e a lógica por trás do mesmo. Neste sentido, códigos mal documentados podem resultar em perda de pontuação.

Formato de Submissão

Quando terminar seu case, envie-o nesse forms. Não aceitaremos múltiplas submissões.

Você deve seguir o seguinte formato:

- As respostas dos exercícios de lógica de programação e algoritmos podem ser entregues em Python (.py), C (.c), C++ (.cpp) e Java (.java). Note que **esperamos um arquivo por questão/item**, com a questão devidamente identificada no nome do arquivo
- As respostas dos exercícios de análise de dados e IA devem ser entregues em Jupyter Notebook (.ipynb).
- O seu texto (Turing Talks) deve ser entregue em pdf.

Os nomes dos arquivos e pastas devem identificar as questões que eles resolvem, **conforme a estrutura abaixo** (só inclua as partes que você resolver, não precisam estar completas para mandá-las). A submissão deve ser feita por meio de um único arquivo zip com o nome "`nusp_nome_sobrenome.zip`".

```
12345678_alan_turing.zip
├ 1_logica
| ├ q1a.py (ou demais formas)
| ├ q1b.py (ou demais formas)
| ├ q2a.py (ou demais formas)
| ├ q2b.py (ou demais formas)
| └ desafio.py (ou demais formas)
├ 2_analise-e-ia
| └ analise_e_modelos.ipynb
└ 3_turing-talks
  └ turing-talks.pdf
```

Observações:

- Recomendamos dar uma passada em todas as partes antes de começar a fazer o case para escolher o que você se sente mais confortável em fazer! E não fique assustado com o tamanho dos enunciados, a maioria do conteúdo foi colocado justamente para ajudar na realização da tarefa.
- Como nas outras etapas, utilize nusp 0 (zero) se você for membro da USP, mas não tiver nusp.

Parte 1: Lógica de Programação e Algoritmos

40 pontos | Nível Intermediário

Caso tenha alguma dúvida nessa parte do case, é só perguntar para a gente no nosso Discord (no canal **#duvidas-intermediario-logica**), no nosso e-mail turing.usp@gmail.com, na nossa página facebook.com/turing.usp ou mandar uma mensagem para algum dos membros da equipe abaixo:

Wesley Almeida	(python, c, c++, java)	+55 (11) 97027-6690
Fernando Matsumoto	(python, c, c++, java)	+55 (19) 99850-2800
Guilherme Salustiano	(python, c, c++, java)	+55 (41) 99928-7997
Rodrigo Fill	(python, c, c++)	+55 (11) 97277-0014
Noel Eliezer	(python)	+55 (16) 98126-8981
Ariel Guerreiro	(python)	+55 (11) 94331-1181

Essa parte do case tem **2 questões** (Q1, Q2) de temas diferentes, cada uma com **itens A e B**, além de uma questão **desafio**. No final de cada item e do desafio são dados exemplos de entrada e saída para que você teste os seus programas.

Algumas dicas gerais:

- Os exemplos podem ajudar na compreensão dos exercícios.
- Depois de escrever os seus programas, é importante testá-los com os exemplos dados, mas lembre-se de que você também pode escrever os seus próprios exemplos.

Observações técnicas:

- Nessa parte do case, estamos considerando as seguintes versões de cada linguagem:
  - **Python 3**: versão 3.8 ou anterior
  - **C/C++ 11** (códigos escritos em versões anteriores costumam funcionar na versão 11, mas é bom verificar)
  - **Java 15** ou anterior
- Em java, **não coloque o seu código dentro de nenhum pacote** (não coloque `package xxx;` no início do arquivo).

Q1. Vacas magras no Turing

Formato: .py, .c, .cpp, .java

Com a entrada dos membros ao Turing, a área de estratégia precisa enviar os e-mails de todos os membros para o professor Bufo-Chan cadastrá-los no Datacamp. Bufo-Chan é um professor muito antigo da USP e por isso não utiliza e-mail e a única forma de enviar a lista é por correio.

Infelizmente, como o processo de arrecadação de fundos do grupo ainda não começou, foi decidido utilizar o mínimo possível de tinta para imprimir a lista com emails. Um dos membros sugeriu a seguinte solução: alinhamos os e-mails à direita. A partir do segundo e-mail impresso, **os caracteres iniciais** do próximo e-mail a ser impresso que coincidirem com os do e-mail acima são omitidos, ficando apenas um espaço em branco.

Por exemplo para os emails `camila.lobianco@usp.br`, `camilalala.topp@usp.br` e `azank.pistolado@usp.br` a impressão ficará da seguinte forma:

```
c a m i l a . l o b i a n c o @ u s p . b r
          l a l a . t o p p @ u s p . b r
a z a n k . p i s t o l a d o @ u s p . b r
```

Pelo exemplo acima é possível perceber que foram economizadas 6 letras. Os membros cogitaram remover também os caracteres finais que se repetissem, mas entenderam que isso poderia tornar muito difícil a leitura para o professor.

Item A (10 pontos)

Submissão: 1_logica/q1a.py (.c, .cpp, .java)

Você, entusiasmado por ter entrado no grupo, se propôs a fazer um programa que recebe a lista de emails e indica quantas letras foram economizadas. Neste item, você deve determinar quantas letras podem ser economizadas **mantendo a ordem** dos emails.

Referências do Item A:

- Vetores - IME-USP
- Ordenação de vetores

Formato de Entrada e Saída

Entrada

Sua entrada é composta de N, seguido de N emails (um por linha).

Limites:

- $1 < N < 100$

Saída

Seu programa deve *printar* um número indicando quantas letras podem ser economizadas para o conjunto N de e-mails.

Exemplos

Exemplo 1

Entrada do exemplo 1:

```
3
camila.lobianco@usp.br
camilalala.topp@usp.br
azank.pistolado@usp.br
```

Saída do exemplo 1:

```
6
```

Exemplo 2

Entrada do exemplo 2:

```
5
nelson.turingusp@usp.br
noelson.turingusp@usp.br
noelson.estudausp@usp.br
matsumoto.membrot@usp.br
mateus.fakemembro@usp.br
```

Saída do exemplo 2:

```
12
```

Item B (10 pontos)

Submissão: 1_logica/q1b.py (.c, .cpp, .java)

Enquanto os membros do Turing estavam imprimindo a lista de e-mails, eles perceberam que a impressora foi infectada por um vírus e está imprimindo de forma incorreta. Depois de olhar para várias páginas impressas por um tempo, você percebe que ele está imprimindo cada linha de dentro para fora. Em outras palavras, a metade esquerda de cada linha está sendo impressa a partir do meio da página até a margem esquerda. De maneira similar, a metade direita de cada linha está sendo impressa a partir da margem direita e prosseguindo em direção ao centro da página.

Por exemplo, o e-mail `azank.pistolado@usp.br` está sendo impresso como `otsip.knazarb.psu@odal`. Isso ocorre pelo seguinte processo:

`azank.pistolado@usp.br`	original
`azank.pisto | lado@usp.br`	orig dividido
`otsip.knaza | rb.psu@odal`	div invertido
`otsip.knazarb.psu@odal`	invertido

Além disso, você percebeu que alguns e-mails também tiveram seus domínios embaralhados/errados após a impressão, inviabilizando o envio de tais e-mails ao professor. Um dos exemplos é o e-mail do membro Noel, que foi impresso como `irut.leonrb.pus@gn` e após a desinverte-lo temos `noel.turing@sup.br`. Perceba que nesse caso o domínio `@usp.br` está errado.

Sua tarefa nesta questão é printar os e-mails corretamente formatados a partir da lista impressa ou ERRO caso o domínio esteja errado.

Formato de Entrada e Saída

Entrada

Sua entrada é composta de um inteiro N que representa a quantidade de e-mails impressos. Em seguida são fornecidos N e-mails, um em cada linha.

Limites:

- 1 < N < 10000

Saída

Seu programa deve printar 1 e-mail por linha corretamente formatado ou ERRO caso o domínio esteja errado.

Exemplos

Exemplo 1

Entrada do exemplo 1:

```
3
ibol.alimacrb.psu@ocna
t.alalalimacrb.repsu@ppo
.orbmem_ovonrb.psu@gnirut
```

Saída do exemplo 1:

```
camila.lobianco@usp.br
ERRO
novo_membro.turing@usp.br
```

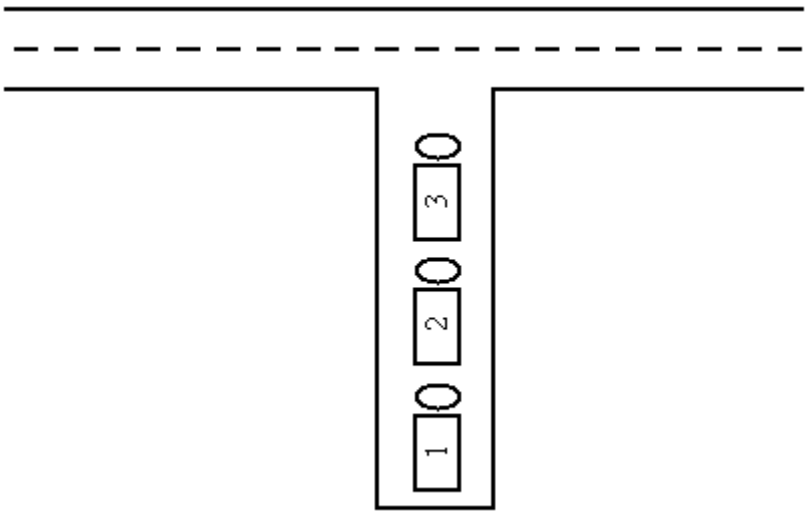
Q2. Estacionamento do Nelnelson

Formato: .py, .c, .cpp, .java

Depois de muito tempo juntando dinheiro, Nelnelson, um dos membros mais exemplares do Turing, conseguiu juntar dinheiro para comprar seu primeiro carro (parcelado, é claro). Chega de pegar ônibus, agora sua vida será mais fácil. Pelo menos isso é o que ele pensava até lembrar do estacionamento do seu condomínio.

O estacionamento do condomínio é composto de apenas 1 corredor que comporta K carros um atrás do outro. Como este estacionamento só tem um portão, só é possível entrar e sair por ele.

Ao entrar com o primeiro carro, ele ocupa a posição próxima a parede (carro 1). O próximo carro fica à frente deste (carro 2) e assim por diante. A imagem a seguir demonstra um exemplo:



Obviamente, não é possível que um carro passe por cima de outro, portanto só é possível que um carro saia do estacionamento se ele for o último da fila (nesse exemplo o carro 3).

Referências gerais para a questão 2:

- Vetores - IME-USP
- Pilha - IME-USP
- Pilha em Python - Pandas IME USP
- Pilhas em Python com listas

Item A (10 pontos)

Submissão: 1_logica/q2a.py (.c, .cpp, .java)

Você, como um bom programador, quer ajudar Nelnelson a saber se todos os moradores do condomínio conseguiram estacionar seus carros. Dessa forma, você decide escrever um programa que verifica se é possível que todos os carros consigam estacionar e sair sem problemas.

Dica: Utilize uma pilha para fazer a inserção e remoção dos carros no estacionamento.

Formato de Entrada e Saída

Entrada

Sua entrada é composta de K e N (tamanho do estacionamento e número de instantes que serão monitorados). Em seguida, cada linha representa um instante, ou seja, a primeira linha é o primeiro instante, a segunda, o segundo instante e assim por diante. Para cada linha será dado um valor C. Se C for positivo, o carro indicado pelo número está entrando no estacionamento. Se C for negativo, o carro indicado pelo número está saindo do estacionamento. **Veja o exemplo 1 para maiores explicações**.

Limites:

- $1 \leq K < 1000$
- $1 \leq N < 100000$
- $1 \leq |C| < 1000$

Obs: $|x|$ representa o módulo ou valor absoluto de x, isto é, se $x \geq 0$, então $|x| = x$. Se $x < 0$, então $|x| = -x$. Por exemplo: $|-3| = 3$ e $|3| = 3$. Acima estamos dizendo que o valor absoluto da variável C estará sempre entre os limites especificados.

Saída

Você deve *printar* `sim`, caso seja possível que todos os carros estacionem e saiam sem problemas e `nao`, caso contrário. Caso um carro não consiga entrar no estacionamento por ele já estar lotado, seu programa deve printar `nao`.

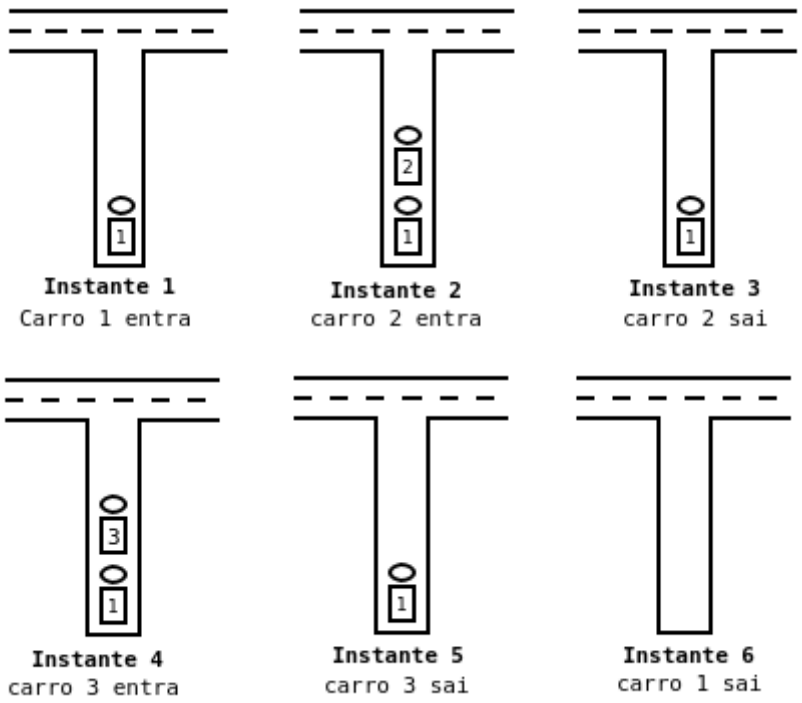
Exemplos

Exemplo 1

Entrada do exemplo 1:

```
3 6
1
2
-2
3
-3
-1
```

No exemplo acima, quando temos -3, por exemplo, significa que o carro 3 está saindo do estacionamento. Veja:



Saída do exemplo 1:

```
sim
```

Exemplo 2

Entrada do exemplo 2:

```
3 6
1
2
-2
3
5
-3
-1
-5
```

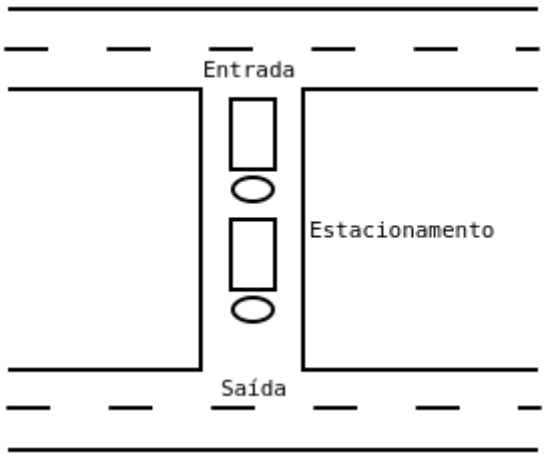
Saída do exemplo 2:

```
nao
```

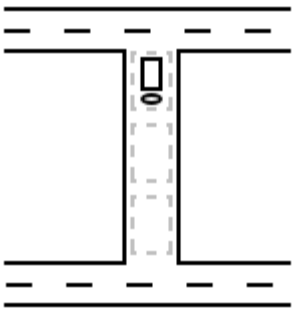
Item B (10 pontos)

Submissão: 1_logica/q2b.py (.c, .cpp, .java)

A rua em que Nelson irá guardar seu carro, que antes era sem saída em uma das extremidades, agora possui uma entrada e uma saída conforme a seguinte imagem mostra:

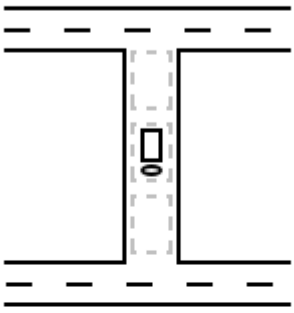


Este estacionamento funciona de uma maneira muito peculiar, pois a cada novo instante o carro avança uma posição dentro dele. Veja o exemplo para um estacionamento de tamanho K = 3 e um determinado carro:



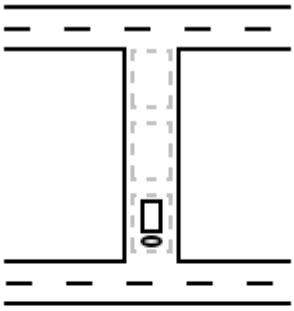
Instante 1:

O carro está no estacionamento



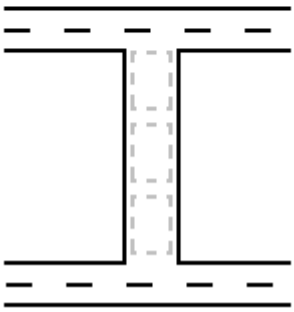
Instante 2:

O carro avança 1 posição no estacionamento



Instante 3:

O carro avança mais 1 posição no estacionamento



Instante 4:

O carro sai do estacionamento

Pelo exemplo, acima percebemos que uma vez que o carro entra no estacionamento, a cada instante ele avança uma posição até sair. Dito isso, você consegue ajudar Nelson a descobrir como está o estacionamento em um determinado instante?

Formato de Entrada e Saída

Entrada

Sua entrada é composta por dois números K e N (tamanho do estacionamento e números de carros que entram no estacionamento), em seguida, nas próximas N linhas é dado o horário de entrada (E_n) de cada carro (a primeira linha representa o carro 1, a segunda linha o carro 2 e assim por diante). Por fim é dado um instante T, no qual queremos saber como está o estacionamento.

Limites:

- $1 < N, K < 10000$
- $1 \leq E_n < 1000$
- $1 \leq T < 10000$

Saída

Sua saída deve ser composta de K números (tamanho do estacionamento) **separados por 1 espaço entre cada número e em uma única linha**, que indicam como está o estacionamento no instante T. Caso uma posição do estacionamento não possua nenhum carro, você deverá *printar* `0`. Além disso, o primeiro número impresso representa o carro na primeira posição do estacionamento, isto é, logo depois da entrada. Veja o exemplo a seguir para maiores explicações.

Exemplos

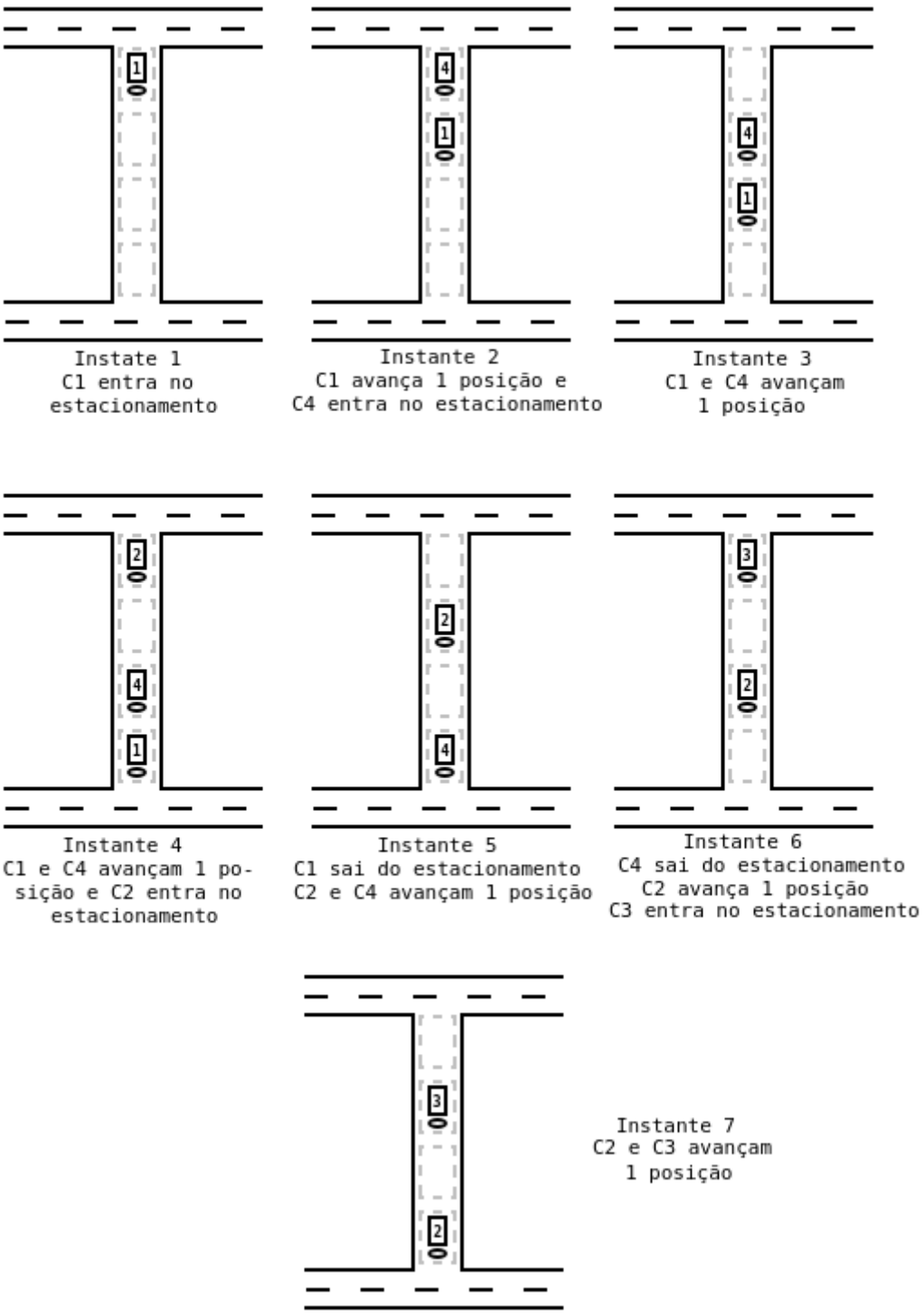
Exemplo 1

Entrada do exemplo 1:

```
4 4
1
4
6
2
7
```

No exemplo acima perceba que K = 4 (tamanho do estacionamento) e N = 4 (número de carros que entram no estacionamento). Além disso, sabemos que: o carro 1 entrou

no instante 1, o carro 2 entrou no instante 4, o carro 3 entrou no instante 6 e o carro 4 entrou no instante 2. Por fim, o instante em que queremos saber como está o estacionamento é o 7.



Saída do exemplo 1:

```
0 3 0 2
```

Exemplo 2

Entrada do exemplo 2:

```
5 7
3
6
12
9
4
15
16
7
```

Saída do exemplo 2:

```
0 2 0 5 1
```

Observações para o item B:

- O usuário pode considerar que dois carros nunca entrarão no mesmo instante no estacionamento
- Se o instante de interesse é X e um carro **entra** no estacionamento no mesmo instante X, considera-se que o carro **já está** no estacionamento. De maneira análoga, se um carro **sai** do estacionamento no mesmo instante X de interesse, considera-se que o carro **não está** no estacionamento.

Desafio. Amizades do Turing

Formato: .py, .c, .cpp, .java

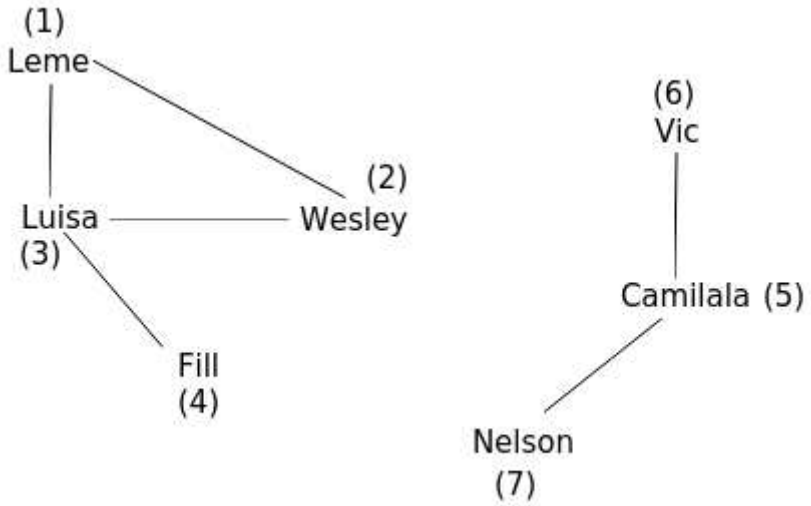
Atenção: esse case tem dois desafios: um de *análise de dados e IA* e outro de *lógica de programação e algoritmos*. Lembre-se de que:

- eles valem uma **pontuação extra**, i.e., eles não podem elevar a sua nota no case técnico acima dos 100 pontos; e
- Se optar por fazê-los, você deve escolher **no máximo um deles**. Em outras palavras, **não entregue ambos desafios**.

Item Único (10 pontos)

Submissão: 1_logica/desafio.py (.c, .cpp, .java)

O Turing é formado por membros de vários institutos da USP, como IME, FFLCH, FEA e POLI. Os membros da área de RH perceberam que micro grupos estão se formando dentro do Turing devido a proximidade de membros do mesmo instituto, o que atrapalha a integração geral do grupo. Você, a fim de ajudar a área de RH do grupo a entender tal fenômeno, se comprometeu a criar um programa que dada a relação de amizade entre os participantes do Turing, devolve quantos micro grupos existem atualmente no grupo. Por simplicidade, os relacionamentos podem ser representados como um grafo, como o abaixo.



No exemplo acima, cada nó é um membro, e cada aresta (linha) representa um relacionamentos, demonstrando que as duas pessoas envolvidas são amigas. Analisando o grafo, percebemos que o número de micro grupos é **2**.

Referências gerais para o desafio:

- Grafos
- Introdução à representação de grafos em Python

Formato de Entrada e Saída

Entrada

Sua entrada é composta de dois inteiros N e K, que representam o número de membros e de relacionamentos existentes entre os membros do Turing. Cada uma das próximas N linhas contém dois inteiros A e B, correspondendo a um relacionamento entre os membros A e B.

Limites:

- $1 < N, K < 50$
- $1 < A, B < 100$

Saída

Seu programa deve imprimir o número de micro grupos existentes no Turing.

Exemplos

Exemplo 1

Nesse exemplo, que corresponde ao grafo mostrado no enunciado, o primeiro valor da entrada é o número de membros (7) e o segundo é a quantidade de relacionamentos (6) no grupo. Em seguida, cada linha representa um relacionamento. O grafo correspondente é o mesmo que foi mostrado no início dessa questão.

Entrada do exemplo 1:

```
7 6
1 2
2 3
3 1
3 4
6 5
5 7
```

Saída do exemplo 1:

```
2
```

Parte 2

Análise de Dados e Inteligência Artificial

40 pontos | Nível Avançado

Ao trabalhar com dados, podemos encontrar duas principais categorias: os dados estruturados e os não-estruturados.

Os dados estruturados são aqueles que têm uma estrutura rígida, por exemplo uma planilha contendo dados sobre a venda de algumas lojas: uma coluna poderia conter o nome da loja, outra o número vendido de um dos itens disponíveis ali, outra o preço desse item, etc.

Os dados não estruturados, por outro lado, não possuem estruturas bem definidas. Aqui entram imagens, textos e áudio, por exemplo.

A quantidade de dados não estruturados tende a crescer cada vez mais rápido e representam a maior parte dos dados gerados hoje, sejam em mensagens de whatsapp, tweets, músicas ou vídeos de youtube.

Ao trabalhar com IA ou Machine Learning, você, com certeza, em algum momento se deparará com dados não estruturados.

Por isso, nesse case, nós iremos trabalhar com textos, mais especificamente com um dataset de **reviews de videogames da Amazon**. Esse conjunto de dados advém do dataset Amazon product data. Não se preocupe em baixar os dados da fonte original, utilize **esse link**, alí já separamos o que deve utilizar como dados de treino,validação e teste e fizemos um sample da base de dados. Aqui constam mais informações sobre o dataset original.

Se você recebeu esse case, esperamos que você já saiba o que são dados de treino, validação e teste!

Em suma, esse dataset contém reviews para jogos vendidos na Amazon. Ele contém o nome do usuário, a nota dada, o timestamp da review, o título dado para a review, o número de upvotes e down-votes e, claro, o texto da review em si. **A sua tarefa será**

fazer um modelo que dada uma review, prevê sua polaridade (positivo se a nota for maior que três e negativo se a nota for menor ou igual a três).

Caso tenha alguma dúvida nessa parte do case, é só perguntar para a gente no nosso Discord (no canal **#duvidas-avancado-analise**), no nosso e-mail turing.usp@gmail.com, na nossa página facebook.com/turing.usp ou mandar uma mensagem para algum dos membros da equipe abaixo:

Luísa Mendes Heise	(python, R)	+55 (11) 99218-7534
Vitoria Rodrigues	(python)	+55 (11) 98748-8154
Fernando Matsumoto	(python)	+55 (19) 99850-2800
Camilla Fonseca	(python, R)	+55 (11) 96163-7978

O que é Processamento de Linguagem Natural?

NLP (Natural Language Processing) é o estudo relacionado à interação entre computadores e linguagens naturais dos seres humanos. Sendo assim, o foco dessa área de estudo é, de certa forma, ensinar o computador a entender e interpretar as linguagens humanas.

Ao conversarmos com alguém, somos capazes de determinar como a outra parte da conversa está se sentindo, analisando suas palavras e expressões, conseguimos expressar emoções através da maneira como falamos e temos total capacidade de entender o assunto sendo discutido. Tudo isso subconscientemente, não precisamos nos esforçar muito para entender tons negativos e positivos durante uma discussão. Além disso, gírias e abreviações utilizadas tão amplamente em redes sociais não são desafiadoras para você, e seu cérebro consegue interpretar de forma automática um “vc” como "você" e um “tb” como “também”. Entretanto, essas tarefas, tão simples para nós, são desafiadoras para máquinas.

Como você pode explicar para um computador que a frase “Não estou me sentindo feliz hoje” tem uma conotação negativa, ou que “eu tb ñ gosto de ficar em casa no fds” significa “Eu também não gosto de ficar em casa no fim de semana”? Seria um computador capaz de interpretar frases humanas, com suas singularidades? E seria ele capaz de, em algum nível, até reproduzir a nossa linguagem? Em NLP, estudamos essas relações e queremos que seja possível um computador interpretar, entender e até mesmo reproduzir essas características.

Ferramentas úteis/recomendadas

Recomendamos que a resolução seja feita em Python, mas você pode usar R, se sentir mais confortável. Dentro das linguagens de programação, **recomendamos** os seguintes pacotes/ bibliotecas (lembrando que **você pode usar outras libs** além das especificadas aqui):

Python

- Para ler e manipular os arquivos de dados:
  - Pandas
- Para criar visualizações:
  - Seaborn
  - Matplotlib
  - Plotly
- Para pré-processamento e criação de modelos de machine learning:
  - Spacy
  - NLTK
  - Sklearn
  - Pytorch ou Keras (para o desafio)
- Se você quiser fazer uma wordcloud ;)
  - Wordcloud

R

- Para manipular os arquivos de dados e criar visualizações:
  - Tidyverse
  - Plotly
- Para pré-processamento e criação de modelos de ML:
  - Spacy
  - Quanteda
  - tidytext
  - caret
  - keras r (para o desafio)
- Se você quiser fazer uma wordcloud ;)
  - Wordcloud ou Quanteda

Google Colab

Essa parte do case deve ser feita **utilizando jupyter notebook.** Se você não possui o jupyter instalado no seu computador, ou se preferir, pode usar o **Google Colab.** O Google Colab é uma ferramenta do Google que permite rodar jupyter notebooks na nuvem.

Para criar um notebook no Colab, acesse o link https://colab.research.google.com/notebook#create=true.

Para criar o notebook em R, utilize o link: https://colab.research.google.com/notebook#create=true&language=r.

Caso vá fazer o desafio, recomendamos que utilize um *runtime* com GPU. Para isso clique em *Ambiente de Execução > Alterar tipo de ambiente de execução > Acelerador de hardware > GPU.*

Resumo das tarefas

Submissão: 2_analise-e-ia/analise_e_modelos.ipynb

Pré-processamento (12 pontos)

- Faça o pré-processamento do corpus do dataset com (tokenização), (remoção de stopwords) e (stemização ou lematização)

Análise do corpus (16 pontos)

- Nada fixo, algumas sugestões: entidades frequentes, classes gramaticais, frequências das palavras, tamanho das reviews

Modelagem e feature engineering (12 pontos)

- Aplique tf-idf ou bag-of-words para criar as features do seu modelo.
- Treine pelo menos dois modelos de machine learning (que não podem ser de deep learning).

- Avalie seus modelos, usando uma ou mais métricas adequadas, que você deve justificar no seu notebook.

Desafio - Deep Learning (10 pontos extra)

- Prepare seus dados: faça um mapeamento vocabulário -> id e padding das sequências
- Construa uma rede neural
- Treine sua rede e comente os resultados

Pré-processamento (12 pontos)

Especialmente quando utilizamos métodos mais antigos de classificação de texto, o comum é aplicar uma série de pré-processamentos ao corpus (conjunto de textos), para que haja uma melhor performance dos modelos.

Isso, entretanto, é uma etapa que não é ortodoxamente feita quando falamos de deep learning. Por isso, se você for fazer o desafio, tenha em mente que esse passo não será necessário.

Os passos que são comumente utilizados no pré-processamento são:

1. Tokenização
2. Remoção de stopwords
3. Lematização ou Stemização

O processo de pré-processamento começa com a tokenização. Esse processo consiste em uma forma de separar um pedaço de texto em unidades menores chamadas tokens.

Por exemplo, a frase: Oi, tudo bem?

É quebrada nos seguintes tokens: “Oi”, “,”, “tudo”, “bem”, “?”

Além disso, lembre-se de passar todas as palavras para letra minúscula. Com isso, garantimos que palavras como "oi" e "Oi" sejam interpretadas da mesma forma. Você pode usar as próprias funções do Python para isso.

Também, é importante remover as stopwords são listas de palavras muito recorrentes em línguas, e que normalmente não agregam muito na hora de analisar os textos, e cada língua tem sua própria lista de stopwords, alguns exemplos em inglês são “the”, “is”, “at”.

Uma última explicação seria o que é Stemização ou Lematização. As duas são formas de achar formas básicas de palavras, facilitando que palavras com significado similares sejam identificadas pelo seu algoritmo. Lematização procura uma palavra que existe que possa ser a raiz da palavra que você deseja lematizar, um exemplo seria as conjugações de “ter”, como “tenho”, “tinha”, “tem”, todas seriam reduzidas a “ter”. Stemização por sua vez exige bem menos poder computacional por utilizar um algoritmo mais simples, que apenas corta uma parte da palavra, o que pode resultar em uma forma não dicionarizada desta.

Para fazer isso, você pode utilizar bibliotecas do Python como Spacy e NLTK, ou no R, spacy (ou qualquer outra biblioteca que você queira). Veja as referências no final do documento.

Algumas referências:

- https://medium.com/@gon.esbuyo/get-started-with-nlp-part-ii-overview-of-an-nlp-workflow-7ba1f5948b24
- https://towardsdatascience.com/nlp-text-preprocessing-a-practical-guide-and-template-d80874676e79
- https://www.vooo.pro/insights/tutorial-sobre-expressoes-regulares-para-iniciantes-em-python/
- https://towardsdatascience.com/nlp-preprocessing-with-nltk-3c04ee00edc0
- https://towardsdatascience.com/text-preprocessing-with-nltk-9de5de891658
- https://medium.com/voice-tech-podcast/implementing-a-simple-text-preprocessing-pipeline-with-spacy-597a3568bc19
- https://towardsdatascience.com/pre-processing-should-extract-context-specific-features-4d01f6669a7e
- https://medium.com/ensina-ai/introdu%C3%A7%C3%A3o-a-processamento-de-linguagem-natural-174936c096b

Faça o pré-processamento do corpus do dataset com (tokenização), (remoção de stopwords) e (stemização ou lematização)

Análise do corpus (16 pontos)

Essa tarefa não tem um script fixo. A ideia é que você procure explorar o corpus e tirar conclusões com base em análises quantitativas. Você pode verificar a relação entre as notas e algumas características do texto da review. Tais como:

a. Frequência de termos (n-gramas)
b. Tamanho das análises (em número de caracteres
c. Entidades extraídas do texto (por meio de um NER pré treinado por exemplo)
d. Classes gramaticais presentes
e. Proporção de letras maiúsculas

Você também pode incluir as outras features (como o número de upvotes e downvotes), timestamp e etc na sua análise.

Algumas referências:

- https://spacy.io/usage/linguistic-features/
- https://programminghistorian.org/en/lessons/counting-frequencies
- https://m-clark.github.io/text-analysis-with-R
- https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk
- https://monkeylearn.com/text-analysis/
- https://blog.dominodatalab.com/natural-language-in-python-using-spacy/
- https://github.com/turing-usp/Fake-Citation/blob/main/Analysis/EDA.ipynb

Modelagem e feature engineering (12 pontos)

Feature Engineering

Você já deve saber o que é uma feature. As features nos dão informações mensuráveis acerca de um fenômeno observado, elas são uma forma estruturada de guardar informação, de tal forma que essas informações podem ser utilizadas por um modelo.

Essa também é uma etapa que não é ortodoxamente feita quando falamos de deep learning. Por isso, se você for fazer o desafio, novamente, tenha em mente que esse passo não será necessário.

Então, agora que você já tem um texto pré-processado, como transformar isso em features para alimentar um modelo?

Bag-of-Words

Existem várias formas de fazer isso. Para os modelos estatísticos ou de machine learning 'clássico' que vamos sugerir aqui, uma abordagem interessante é a chamada **'Bag of words'**. Com o termo 'bag' (do inglês 'saco' ou 'mochila'), queremos dizer que a ordem das palavras não importa, mas sim o número de ocorrências dessas palavras. Em outros tipos de abordagem, a ordem das palavras pode ser fundamental.

Como a ordem das palavras não é levada em conta, esse tipo de abordagem não permite capturar contexto, num sentido de quais palavras aparecem juntas.

A forma mais convencional de usar o Bag of Words é tabular as frequências com que as palavras acontecem em um dado texto, por exemplo:

> *Quero comer bolo de cenoura*
>
> *Quero beber guaraná*
>
> *Preciso comer feijão. Feijão é bom.*

Esses textos poderiam estar, pelos pré-processamentos, da seguinte forma:

`['querer', 'comer', 'bolo', 'cenoura']`

`['querer', 'beber', 'guaraná']`

`['precisar', 'comer', 'feijão', 'feijão', 'ser', 'bom']`

Gerando a seguinte tabulação:

querer	comer	bolo	cenoura	beber	guaraná	precisar	feijão	ser	bom
1	1	1	1	0	0	0	0	0	0
1	0	0	0	1	1	0	0	0	0
0	1	0	0	0	0	1	2	1	1

Uma maneira alternativa de criar features é usar um *vetor de componentes binários,* esses vetores codificam, para cada palavra no vocabulário local, a ocorrência ou não desta numa determinada observação. Nós não gravamos a frequência com os quais os termos ocorrem em novos documentos, apenas sua presença ou ausência. Por exemplo:

Meu irmão é muito muito criança, ele lava seus bonecos.

Ele lava o nariz. E o nariz é de criança.

Com os pré-processamentos poderíamos ter os vetores:

`['meu', 'irmão', 'ser', 'muito', 'muito', 'criança', 'ele', 'lavar', 'seus', 'bonecos']`

`['ele', 'lavar', 'nariz', 'nariz','ser', 'criança']`

Gerando as seguintes features para essa observação:

meu	nariz	irmão	ser	muito	criança	ele	lavar	seus	bonecos
1	0	1	1	1	1	1	1	1	1
0	1	0	1	0	1	1	1	0	0

Para alguns modelos, a segunda abordagem pode ser melhor, enquanto para outros, a primeira pode performar melhor.

TF IDF

O valor tf–idf (abreviação do inglês term *frequency–inverse document frequency,* que significa frequência do termo–inverso da frequência nos documentos), é uma medida estatística.

No caso do bag-of-words, a ideia era simplesmente tabular ocorrências. Aqui, não queremos um número de ocorrências puro, mas sim ponderar a importância de cada palavra dado o quanto ela aparece num documento em específico em relação ao corpus (conjunto de documentos) como um todo.

A parte "tf", de *term frequency*, diz respeito à frequência que um certo termo aparece num documento em específico. Essa parte da fórmula dá maior importância a um termo num documento caso ele apareça muito.

$$tf_{x,d} = \frac{nº\ de\ ocorrências\ do\ termo\ x\ no\ documento\ d}{nº\ termos\ no\ documento\ d}$$

A parte "idf", de *inverse document frequency*, diz respeito a uma medida global daquele termo. A ideia é que um termo raro deve receber mais importância do que um termo comum. A palavra "filme" deve aparecer muito num corpus de avaliações do Netflix, mas ela adiciona pouca informação nesse contexto. A fórmula para medir o "idf" considera a fração dos documentos em que um termo aparece, e faz um cálculo que penaliza isso.

$$idf_{x,D} = \log\left(\frac{nº\ de\ ocorrências\ do\ termo\ x\ no\ conjunto\ de\ documentos\ D}{nº\ documentos\ em\ que\ o\ termo\ x\ aparece\ no\ conjunto\ de\ documentos\ D}\right)$$

Aplique tf-idf ou bag-of-words para criar as features do seu modelo.

Algumas referências:

- https://www.freecodecamp.org/news/an-introduction-to-bag-of-words-and-how-to-code-it-in-python-for-nlp-282e87a9da04/
- https://towardsdatascience.com/natural-language-processing-count-vectorization-with-scikit-learn-e7804269bb5e
- https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089

Modelos de machine learning "clássico"

Um passo natural depois de extrair features de um texto é utilizá-las para treinar modelos de machine learning de forma a gerar previsões.

A ideia aqui é usar um modelo "clássico", ou seja, **não use uma rede neural**, caso queira usar deep learning, faça isso no desafio.

Sugerimos, por exemplo, o uso de modelos do tipo Naïve Bayes. Essa é uma classe de classificadores. Esses modelos são "naive" porque são a mais pura aplicação do famoso teorema de Bayes.

Você pode usar outros modelos do tipo, como boosting, árvores de decisão, regressão logística e afins.

Treine pelo menos dois modelos de machine learning (que não podem ser de deep learning).

Avalie seus modelos, usando uma ou mais métricas adequadas, que você deve justificar no seu notebook.

Algumas referências:

- https://medium.com/turing-talks/sua-primeira-an%C3%A1lise-de-sentimentos-com-scikit-learn-a47c088ea7bd
- https://medium.com/turing-talks/turing-talks-16-modelo-de-pre
- https://medium.com/turing-talks/turing-talks-17-modelos-de-predi%C3%A7%C3%A3o-decision-tree-610aa484cb05
- https://towardsdatascience.com/decision-tree-algorithm-explained-83beb6e78ef4
- https://medium.com/turing-talks/tagged/modelos-de-predi%C3%A7%C3%A3
- https://pythonprogramming.net/machine-learning-tutorial-python-introduction/

Desafio - Deep Learning (10 pontos extra)

Atenção: esse case tem dois desafios: um de *análise de dados e IA* e outro de *lógica de programação e algoritmos*. Lembre-se de que:

- eles valem uma **pontuação extra**, i.e., eles não podem elevar a sua nota no case técnico acima dos 100 pontos; e
- se optar por fazê-los, você deve escolher **no máximo um deles**. Em outras palavras, **não entregue ambos os desafios**.

Os métodos que utilizamos até agora para classificar as reviews são ainda muito utilizados. Entretanto, na última década, a quantidade de dados nos permitiu utilizar modelos com um tamanho muito grande de parâmetros: as redes neurais.

Se você nunca ouviu falar de redes neurais, pode acreditar, o assunto é extenso. Existem muitas variações de redes: vanilla, rnns, transformers, cnns…

A ideia aqui é que você se arrisque um pouco. Mesmo que não tenha conhecimento formal sobre como funciona uma rede neural, a proposta é que você faça uma. Você pode escolher a arquitetura e o framework que vai utilizar.

Lembrando que em Deep Learning nós não seguimos o modelo tradicional de pré-processamento do texto. Em geral, o que é feito é um mapeamento de cada

palavra para um id no vocabulário e, então, cada palavra é substituída por seu id no vocabulário.

Aqui algumas referências que podem ser úteis:

- https://pythonprogramming.net/introduction-deep-learning-neural-network-pytorch/
- https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc
- https://www.tensorflow.org/tutorials/keras/text_classification_with_hub
- https://keras.io/api/datasets/imdb/
- https://wesinalves.github.io/tensorflow/2018/10/16/text-classification.html
- https://medium.com/@dehhmesquita/classificando-textos-com-redes-neurais-e-tensorflow-5063784a1b31
- https://www.alura.com.br/conteudo/treinando-rede-neural-pytorch
- https://mageswaran1989.medium.com/text-preprocessing-with-tensorflow-apis-67ca1aaf3fc3
- https://pytorch.org/tutorials/beginner/text_sentiment_ngrams_tutorial.html

Parte 3: Turing Talks

20 pontos | Nível Avançado

Questão Única

Submissão: 3_turing-talks/turing-talks.pdf

O Turing USP possui três pilares principais que promovem a sustentação de todas as atividades desenvolvidas interna e externamente pelo grupo, são eles: Estudar, Aplicar e Disseminar conhecimentos de inteligência artificial e ciência de dados.

O pilar de disseminação de conhecimento é de extrema importância para o objetivo do grupo de se tornar uma referência no tema ao redor de São Paulo e do Brasil. Para tal temos diversos meios que são importantes: fazemos workshops, rodas de conversas e outros eventos, buscamos a publicação de projetos desenvolvidos internamente, como, por exemplo, a publicação de um artigo sobre o projeto do Fernando Pessoa na revista Superinteressante.

Além disso, gravamos aulas e workshops para serem disponibilizados no canal do youtube do grupo. Em especial, uma das atividades mais importantes para o pilar de disseminação é a publicação semanal dos Turing Talks, na plataforma Medium. Estes textos possuem um alcance muito grande por conta do caráter específico de divulgação científica da plataforma. Devido ao seu fácil acesso, linguagem descontraída, abordagem mais direta, periodicidade e por muitas vezes apresentar de forma consistente um tema dificilmente encontrado com a mesma simplicidade na língua portuguesa, o Turing Talks é um dos projetos recorrentes mais importantes do grupo.

Neste contexto da importância, você deve redigir um **Turing Talks** sobre **um dos temas listados abaixo** que aborda conceitos de inteligência artificial. Como referência para esse trabalho você tem a sua disposição um grande acervo em nossa página no Medium: Turing Talks, bem como qualquer outra fonte na internet. O objetivo é escrever um texto que seja informativo e conciso, não queremos textos puramente expositivos como uma página do wikipédia.

O texto não precisa seguir a norma culta, mas imagine que será um texto publicado com seu nome nele, para milhares de pessoas que buscam aquele conhecimento vão ler

e julgar o conteúdo, por isso é esperado que não existam erros ortográficos e gramaticais graves.

Temas:

- Otimização de hiperparâmetros
- Random Forest
- PCA

Limite de tamanho: 4000 caracteres, sem contar espaços. Recomendamos a fonte Arial ou Times New Roman 12.

Formato de entrega: PDF com o nome especificado na seção *Formato de Submissão*.

Critérios de avaliação:

- Respeitar o limite máximo de caracteres.
- Texto coeso, com começo, meio e fim.
- Referências de pesquisa e construção do texto são **obrigatórias**.
- O título e a estrutura estética em um turing talks são importantes, atente-se às imagens que você usa para construir o seu texto.
- Originalidade e criatividade do texto.