FIAP ON

INTELIGÊNCIA ARTIFICIAL
E COMPUTACIONAL

AGENTES DE SISTEMAS

ERICK GALANI MAZIERO

LISTA DE FIGURAS

LISTA DE CÓDIGOS-FONTE

SUMÁRIO

7 AGENTES DE SISTEMAS

Sistemas computacionais clássicos atuam em suas tarefas por meio de algoritmos desenvolvidos para lidar com situações e dados já previstos. Por exemplo, um sistema de cadastro de pacientes de um consultório médico pode estar programado para apenas receber os dados digitados por um usuário e designá-los para suas respectivas tabelas em um banco de dados. Nesse sistema hipotético, algumas funcionalidades, como busca por pacientes e criação de relatórios, podem, também, estar implementadas.

Mas qual a diferença de um sistema desses para um sistema inteligente? Como tornar esse suposto sistema em um sistema inteligente? Imagine que, dentre as informações inseridas pelo usuário no sistema, estejam dados do paciente, como: data de nascimento, peso, altura, doenças anteriores e atuais, medicamentos que o paciente toma, se já fez alguma cirurgia, etc. etc. Um sistema inteligente pode analisar essas informações e correlacioná-las com as informações de outros pacientes com o intuito de diagnosticar alguma doença ou criar um alerta para o médico que atenderá o paciente, indicando algo que deve ser verificado.

Um sistema que seja dotado dessa “inteligência” foge dos moldes padrões de desenvolvimento de um sistema de consultório médico e contém um agente inteligente que o auxilia no tratamento da informação, assim como nas tomadas de decisão.

Neste curso, até o momento, vimos diversas abordagens e técnicas que permitem a criação de agentes inteligentes, tais como a representação do conhecimento, sistemas especialistas e o uso de aprendizado de máquina, considerando, inclusive, a incerteza nas tomadas de decisão.

Neste capítulo, veremos que tais abordagens e técnicas podem ser aplicadas em diversos escopos e tipos de dados. Cada qual com suas peculiaridades e técnicas mais indicadas. Consideraremos o Processamento da Linguagem Natural (tanto escrita quanto falada), o Processamento de Imagens e Robótica.

7.1 Processamento da Linguagem Natural

O Processamento da Linguagem Natural (PLN) visa capacitar os computadores a processar a linguagem natural. Uma linguagem natural, como o Português (usado para a escrita deste documento), tem peculiaridades que linguagens de programação não têm. Um compilador não fica na dúvida quando vê um comando, ele sabe exatamente o que fazer. Já uma frase em linguagem natural pode levar a uma ambiguidade de interpretação. Considere a simples frase a seguir:

“Eu vi o homem de binóculos.”

Quem estava de binóculos, “Eu” ou o “homem” que foi visto? Talvez, se disséssemos:

“Eu vi o leão de binóculos.”

Teríamos mais chances de interpretar que “Eu” estava de binóculos, pois leões não usam binóculos. Certo?

O PLN é composto de diversas técnicas que possibilitam a interpretação textual de forma automática, desde o entendimento de cada palavra até porções maiores como sentenças, parágrafos e documentos. Tal interpretação também ocorre em diversos níveis linguísticos: léxico, morfossintático, sintático, semântico e discursivo, visto que a complexidade e subjetividade de interpretação aumenta do léxico ao discursivo.

Para se familiarizar um pouco mais com o PLN, nessa área são desenvolvidos os tradutores automáticos, revisores ortográficos e gramaticais, e quaisquer outras ferramentas que necessitem lidar com textos de forma inteligente.

Um fato que tem tornado o interesse em PLN mais forte é que boa parte das informações disponíveis na Web estão em formato textual. Inclusive, nas redes sociais (como Facebook e Twitter) e blogs, as pessoas expressam suas opiniões, fazem suas críticas e sugestões, falam sobre os mais diversos assuntos, tornando tais conteúdos preciosíssimos para empresas, por exemplo. Empresas podem verificar a aceitação de seus produtos e serviços por meio dessas informações, assim como obter sugestões de melhoria em tais produtos e serviços.

Podemos citar diversas tarefas realizadas pelo PLN, tanto de análise, como de transformação e geração textual:

- Análise:
    - Etiquetação morfossintática (identificação desambiguada das classes de palavras: substantivo, verbo, adjetivo, etc.);
    - Parsing Sintático (identificação da estrutura das sentenças em sujeito, verbo, objetos, adjuntos, etc.);
    - Análise de Sentimentos (se um trecho de texto tem um sentimento positivo, negativo ou neutro);
    - Identificação de entidades (identificação de locais, organizações, pessoas, etc.);
    - Parsing Discursivo (identificação dos motivos que levaram o produtor do texto a dizer o que disse e da forma que disse).
- Transformação:
    - Tradução automática (transformação automática da língua de um texto para outra);
    - Sumarização automática (redução do tamanho do texto mantendo as informações mais importantes).
- Geração textual (geralmente para colocar em linguagem natural algum conhecimento em alguma base de conhecimento, tal como é feito pelos *chatbots* mais avançados).

Ainda neste capítulo teremos mais um *hands on* e nele utilizaremos APIs que realizam diversos níveis de interpretação textual, possibilitando o desenvolvimento de sistemas que tenham certa capacidade linguística para tomada de decisão inteligente.

7.2 Processamento de Imagens

Diz um ditado: “uma imagem vale mais que mil palavras”. Então, se eu tiver a capacidade de interpretar textos, quero ter também a capacidade de interpretar imagens, que podem sumarizar, em si mesmas, muita informação. Um agente

inteligente, dependendo de seu propósito, deve ser capaz de perceber o ambiente à sua volta e isso inclui o Processamento de Imagens.

Imagine que você vá a uma loja para comprar uma camiseta igual a de uma pessoa que você viu na internet. Você pode levar para o vendedor uma foto da pessoa que você viu vestindo a camiseta que você quer comprar. O vendedor identificará os detalhes, como cor, modelo, estampa e dirá se tem ou não tal camiseta. Ou então, ele poderá sugerir camisetas semelhantes. Será que um site de vendas online poderia fazer o mesmo? Se ele for dotado de um sistema de Processamento de Imagens, sim!

Um outro exemplo são os carros autônomos, que são munidos de diversas câmeras para perceber o ambiente (além dos mais diversos tipos de sensores, como sonares e infravermelhos).

O Processamento de Imagens processa um lote de informações (imagem) e pode detectar os limites de cada objeto presente, assim como classificar esses objetos.

A figura “Ilustração da segmentação de imagens” ilustra a tarefa de segmentação de imagens. À esquerda da figura está a imagem original e à direita, a imagem com sombreamentos coloridos de acordo com os objetos identificados automaticamente nas imagens. Isso habilita um computador a identificar objetos nas imagens e fazer sua classificação, sejam pedestres, vias, carros, árvores, placas de sinalização, etc.

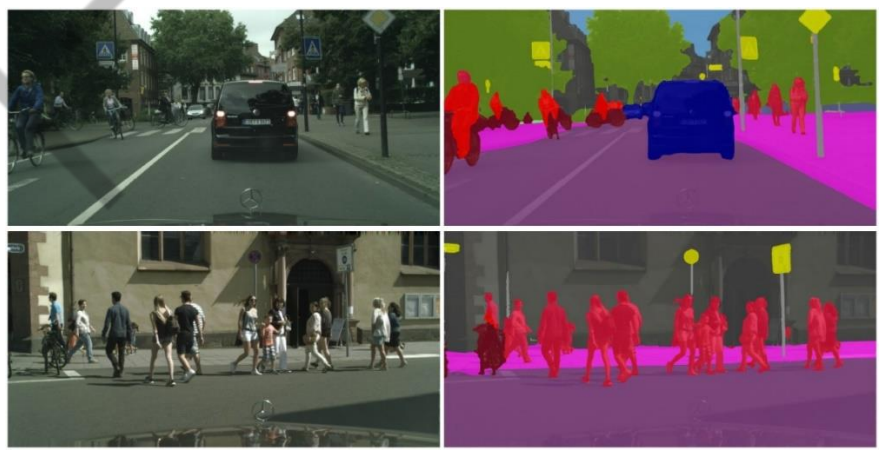

Figura 7.1. Ilustração da segmentação de imagens
Fonte: http://vladlen.info/publications/feature-space-optimization-for-semantic-video-segmentation/ (2016)

Dentre as diversas tarefas do Processamento de Imagens, podemos citar:

- Segmentação de imagens (identificando os objetos presentes na imagem);
- Reconhecimento dos objetos segmentados (classificação dos objetos reconhecidos na imagem em ‘coisas’ do mundo real);
- Reconstrução de cenas (reconstrução de partes danificadas de uma ou mais imagens de um local, por exemplo);
- Restauração de imagens (removendo ruídos, por exemplo).

As APIs de IA também têm serviços de Processamento de Imagens, facilitando a integração desse tipo de inteligência computacional em aplicações.

7.3 Hands On! Usando APIs de Inteligência Artificial!

7.3.1 Ajustando o ambiente

Agora vamos utilizar um ambiente bem parecido com o da prática anterior. Vamos precisar do Python e seu gerenciador de pacotes, o PIP, e do Jupyter.

Antes de mais nada, vamos aprender a utilizar algumas APIs de um dos gigantes da computação: a IBM. Obviamente outras APIs, de outras companhias, poderiam ser utilizadas, tais como o Google Cloud, da Google, ou o Azure, da Microsoft.

A IBM disponibiliza uma série de serviços, para uma série de áreas, tais como Big Data, Internet das Coisas e Aprendizado de Máquina. Para essa última área, tem-se serviços para Processamento de Texto (escrito e falado) e Processamento de Imagens.

Conta IBM

O primeiro passo é a criação de uma conta IBM. Vá para o link a seguir e crie sua conta.

```
https://console.bluemix.net/registration/
```

Caso já tenha uma conta IBM, faça seu login.

`https://idaas.iam.ibm.com/`

O Watson é um conjunto de serviços disponíveis por meio de APIs, tais como a que utilizaremos para processar textos e imagens. Você pode visualizar esses serviços em:

`https://console.bluemix.net/developer/watson/services`

Na figura “Exemplos de serviços disponíveis no Watson da IBM”, são apresentados exemplos de serviços disponíveis na IBM Cloud. Há diversos relacionados ao Processamento da Linguagem Natural, tais como *Speech to Text* (que converte fala para texto escrito), *Language Translator* (que faz a tradução automática de uma língua destino para um alvo) e *Natural Language Understanding* (que realiza a interpretação textual, em seus diversos níveis). É exibido também o serviço de reconhecimento de imagens (*Visual Recognition*).

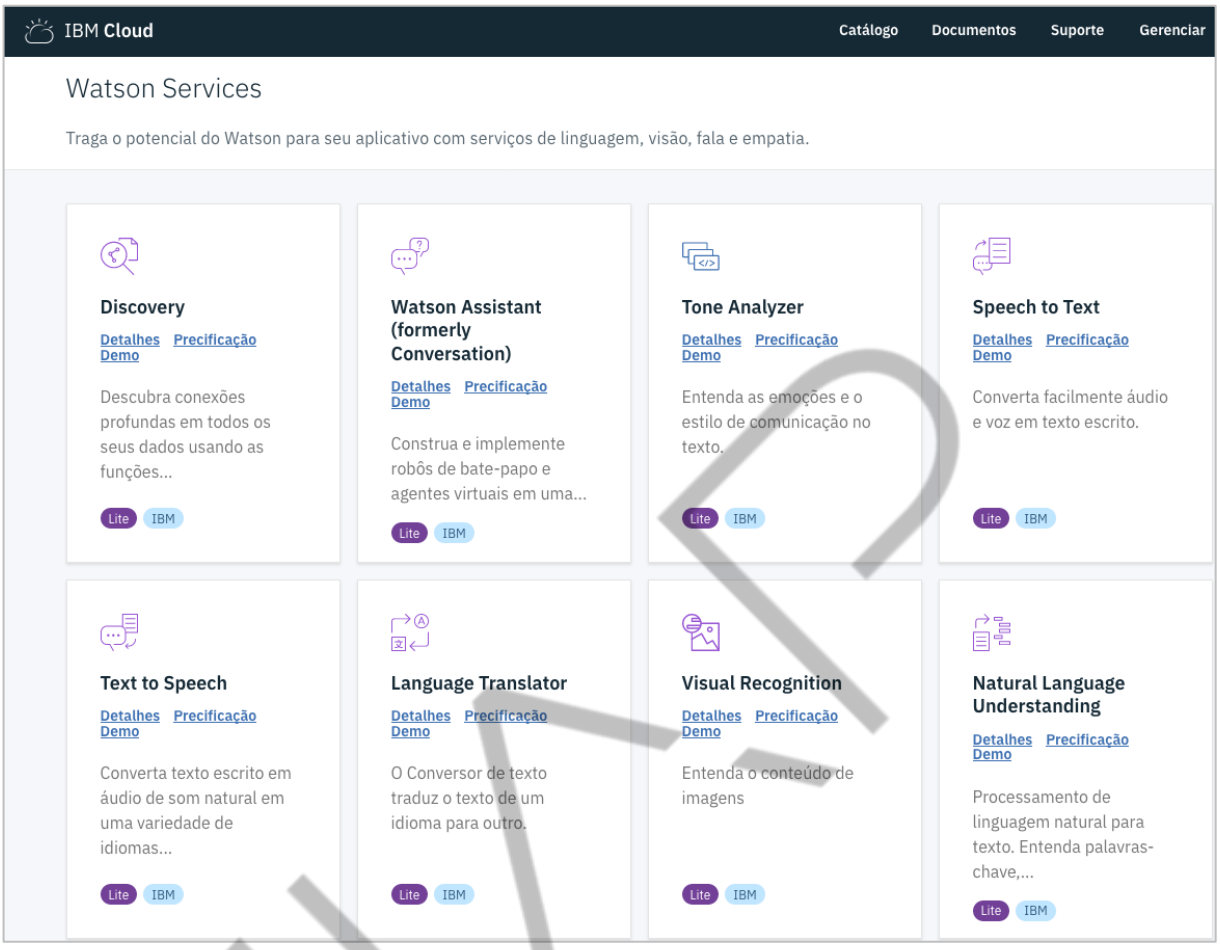

Figura 7.2. Exemplos de serviços disponíveis no Watson da IBM
Fonte: https://console.bluemix.net/developer/watson/services (2018)

Habilitando o serviço de interpretação textual

Acesse o serviço de interpretação textual (*Natural Language Understanding*):

`https://console.bluemix.net/catalog/services/natural-language-understanding`

Defina um nome para o serviço, por exemplo, “Natural Language Understanding-fiap”. Pode deixar as outras opções com seus valores padrões. Por exemplo, a região/local em que vai implementar o serviço diz respeito à infraestrutura física em que o serviço será executado.

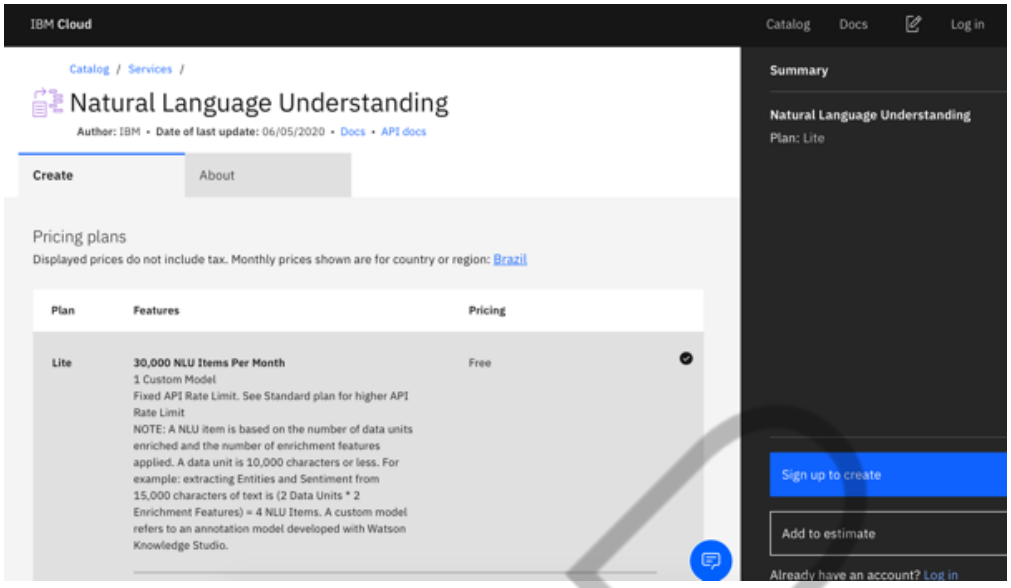

Figura 7.3. Habilitação do serviço de interpretação textual do Watson
Fonte: *https://console.bluemix.net/catalog/services/natural-language-understanding* (2020)

Esse tipo de serviço não é totalmente gratuito, então há um esquema de precificação. A IBM libera um plano (*Lite*) para testes, com precificação grátis. Essa é a primeira opção da tabela de Planos de precificação. Deixe essa opção selecionada e clique em “Criar”.

Você será direcionado a uma página com detalhes do serviço que acabou de habilitar, ilustrado na figura “Informações sobre o serviço habilitado”. Uma informação para uso posterior são as credenciais (*credentials*) com nome de usuário e senha. Anote essas duas informações, que serão necessárias para acessar a API habilitada.

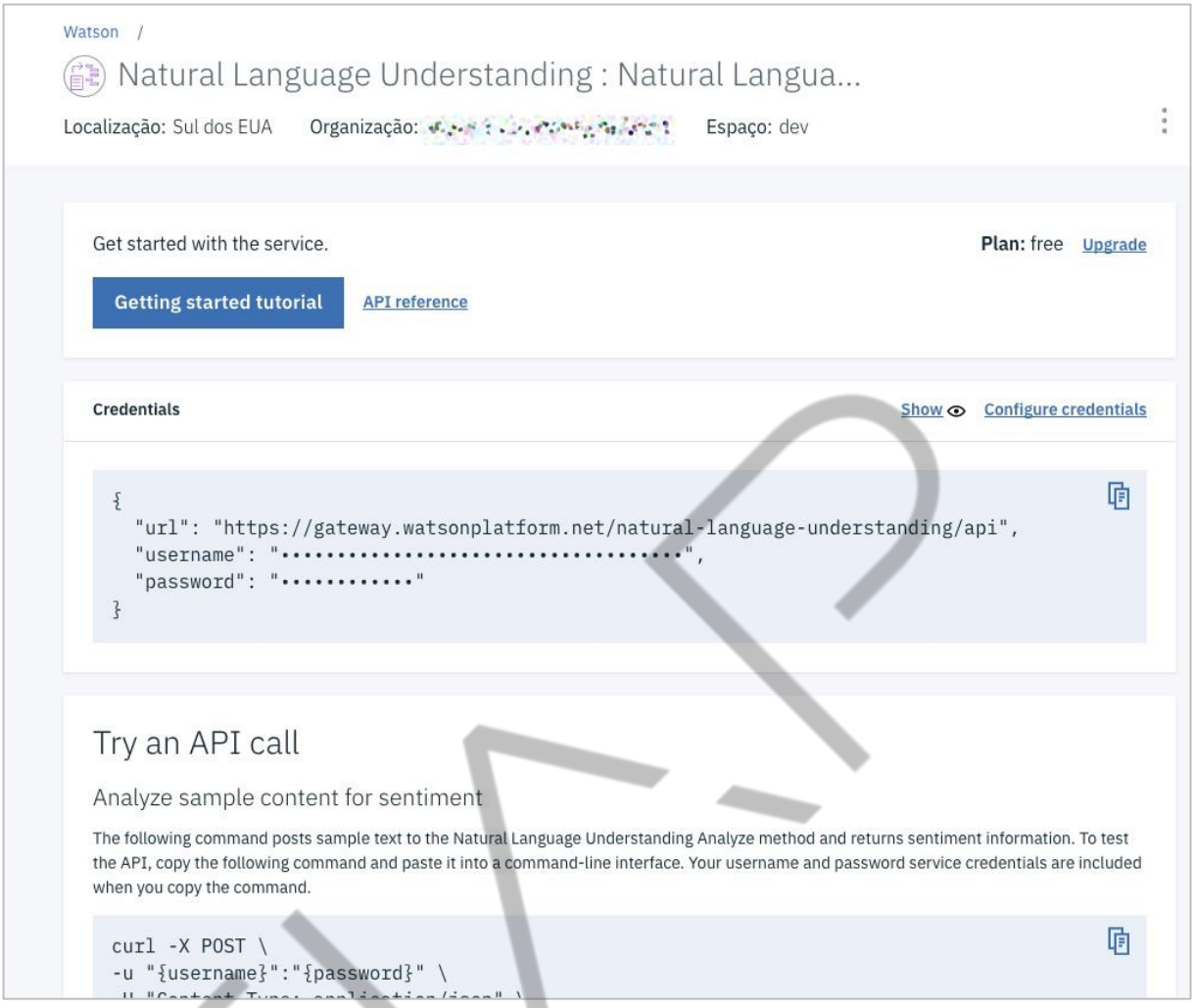

Figura 7.4. Informações sobre o serviço habilitado
Fonte: https://console.bluemix.net/ (2018)

Caso seja necessário acessar novamente as credenciais, apagar, gerar uma nova ou realizar outra operação, no menu lateral esquerdo (figura “Menu de opções do serviço habilitado”), clique em “Credenciais de serviço”. Para cada serviço que for habilitado, deve-se gerenciar uma respectiva credencial de acesso.

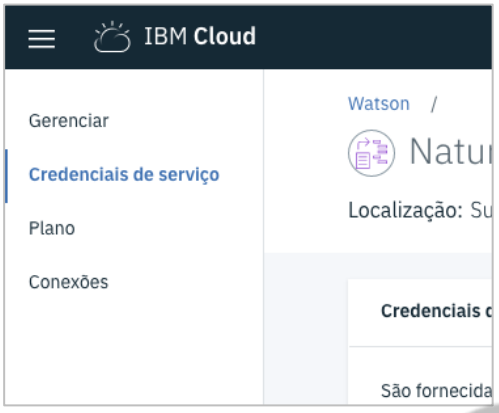

Figura 7.5. Menu de opções do serviço habilitado
Fonte: https://console.bluemix.net/ (2018)

Instalação do módulo de acesso às APIs

Antes de acessar a API utilizando o Python, devemos instalar o módulo necessário. Podemos fazer isso pelo comando a seguir:

```
pip install --upgrade watson-developer-cloud
```

Caso deseje ver o código desse módulo de acesso às APIs, o projeto está disponível em:

```
https://github.com/watson-developer-cloud/python-sdk
```

7.3.2 Acessando as Apis

Pronto, se tudo correu certo até agora, vamos fazer nosso primeiro acesso à API. Inicie o Jupyter (basta abrir o terminal e digitar: `jupyter notebook`). Logo após, o browser padrão irá abrir e a seguinte tela da figura “Tela inicial do Jupyter” será exibida.

Figura 7.6. Tela inicial do Jupyter. No exemplo, é listado um notebook criado com nome "Experiment.ipynb"
Fonte: Imagem gerada pelo autor (2018)

Para iniciar um código Python, basta clicar em "*New->Python 2*", como ilustrado na figura "Iniciando um novo código Python".

Figura 7.7. Iniciando um novo código Python
Fonte: Imagem gerada pelo autor (2018)

Carregamentos dos módulos

Utilizaremos basicamente dois módulos: `json` e `watson_developer_cloud`.

O primeiro (`json`) é utilizado para manipular as informações retornadas pela API. O formato json (*Javascript Object Notation*) é amplamente utilizado em acessos a serviços. Ele é equivalente ao tipo dicionário do Python, isto é, um conjunto de um ou mais chave-valor. Já o módulo `watson_developer_cloud` contém os métodos e tipos necessários para acesso às diversas APIs do Watson.

O código-fonte a seguir apresenta a importação dos módulos. No primeiro bloco, são importados os módulos `json` e, a partir de `watson_developer_cloud`,

o procedimento `NaturalLanguageUnderstandingV1`. No bloco 2, são importadas quais as características do texto serão extraídas. A saber: Conceitos (`ConceptsOptions`), Relações (`RelationsOptions`), Emoções (`EmotionOptions`), Entidades (`EntitiesOptions`).

Além das características (Features) importadas, estão disponíveis outras, como Palavras-Chave (`KeywordsOptions`) e Metadados (`MetadataOptions`).

```
import json
from watson_developer_cloud import NaturalLanguageUnderstandingV1
from watson_developer_cloud.natural_language_understanding_v1 import
Features, ConceptsOptions, RelationsOptions, EmotionOptions, EntitiesOption
s, SemanticRolesOptions, SentimentOptions
```

Código-Fonte 7.1. Importação dos módulos necessários para acesso a uma das APIs do Watson
Fonte: Desenvolvido pelo autor (2018)

Essas características são utilizadas em cada requisição feita à API, dependendo do que se deseja obter como retorno.

Caso deseje ver todos as opções de importação a partir de `watson_developer_cloud`, faça como na Figura 7.8 “Verificando mais módulos a serem importados (*auto-complete*)”, aperte a tecla *tab* após a palavra reservada *import*, será exibida uma lista dos módulos que se pode importar.

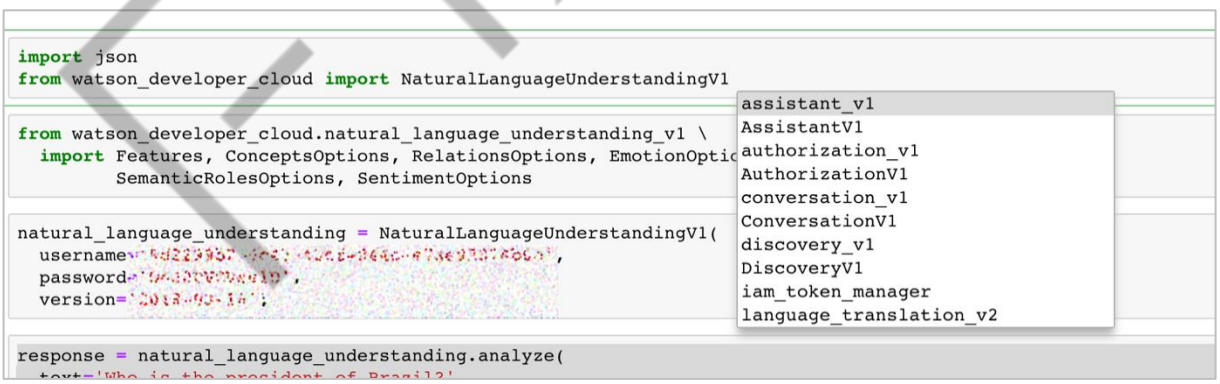

Figura 7.8. Verificando mais módulos a serem importados (*auto-complete*)
Fonte: Imagem gerada pelo autor (2018)

Outra funcionalidade interessante e útil do Jupyter é verificar a assinatura de um método, para isso, com o cursor dentro dos parênteses de um método, aperte <*shift*> + <*tab*>. Experimente!

Instanciando um objeto de acesso à API

Antes de chamarmos a API, temos de instanciar um objeto. Conforme é apresentado no código a seguir, o objeto natural_language_understanding é uma instância da classe NaturalLanguageUnderstandingV1. Na instanciação é necessário indicar o usuário (username) e senha (password), assim como a data da versão dos modelos de IA que serão utilizados (version).

```
natural_language_understanding = NaturalLanguageUnderstandingV1(
  username='coloque_aqui_seu_usuario',
  password='aqui_sua_senha',
  version='2018-03-16')
```

Código-Fonte 7.2. Instanciando o objeto de acesso à API
Fonte: Desenvolvido pelo autor (2018)

As informações de usuário e senha são obtidas logo na habilitação do serviço. Reveja a Figura 7.4, na seção de credenciais (*credentials*).

Consultando a API

Como um objeto, instanciamos e fazemos seu uso para chamar a API de interpretação textual, conforme o exemplo do código abaixo. O método *analyse()* é chamado e seu retorno é posto na variável *response*.

```
response = natural_language_understanding.analyze(
  text='Who is the president of Brazil?',
  features=Features(
    concepts=ConceptsOptions(),
    emotion=EmotionOptions(),
    entities=EntitiesOptions(),
    sentiment=SentimentOptions(),
    ))
```

Código-Fonte 7.3. Exemplo de consulta à API de interpretação textual (NLU) do Watson
Fonte: Desenvolvido pelo autor (2018)

Interessante notar que, no exemplo, o texto apresentado (argumento `text`) está em Inglês. Isso será feito para mostrar melhor as características de análise textual da API, com seus modelos padrões de IA. Para algumas das características (*features*) não há modelos padrões para o Português. Mais adiante, testaremos alguns textos em Português para ver o resultado.

O retorno da API, colocado em `response`, está estruturado como um dicionário. Então, para ver melhor o retorno, utilizamos o código a seguir.

```
print(json.dumps(response, indent=2))
```

Código-Fonte 7.4. Exibição identada do retorno da chamada à API
Fonte: Desenvolvido pelo autor (2018).

O retorno é apresentado a seguir. Nessa resposta, cada chave na raiz do dicionário diz respeito a uma das características (*features*) referenciadas na chamada à API.


```
{
  "emotion": {
    "document": {
      "emotion": {
        "anger": 0.149442,
        "joy": 0.151672,
        "sadness": 0.117418,
        "fear": 0.047087,
        "disgust": 0.198362
      }
    }
  },
  "sentiment": {
    "document": {
      "score": 0.0,
      "label": "neutral"
    }
  },
  "language": "en",
  "entities": [
    {
      "relevance": 0.33,
      "text": "president",
      "type": "JobTitle",
      "count": 1
    },
    {
      "relevance": 0.33,
      "text": "Brazil",
      "disambiguation": {
        "subtype": [
          "GovernmentalJurisdiction",
          "CompanyShareholder",
          "Country"
        ],
        "name": "Brazil",
        "dbpedia_resource": "http://dbpedia.org/resource/Brazil"
      },
      "type": "Location",
      "count": 1
    }
  ],
  "concepts": [
    {
      "relevance": 0.886784,
      "text": "President",
      "dbpedia_resource": "http://dbpedia.org/resource/President"
    },
    {
      "relevance": 0.862208,
      "text": "President of the United States",
      "dbpedia_resource": "http://dbpedia.org/resource/President_of_the
_United_States"
```

```
    },
    {
      "relevance": 0.845824,
      "text": "Politics of Brazil",
      "dbpedia_resource": "http://dbpedia.org/resource/Politics_of_Braz
il"
    }
  ],
  "usage": {
    "text_characters": 31,
    "features": 5,
    "text_units": 1
  }
}
```

No retorno, na característica “*sentiment*”, por exemplo, vemos que o texto é neutro, ou seja, não apresenta algum sentimento. Trata-se apenas de uma pergunta.

Já como entidades no texto (*entities*), foram identificadas duas: “Brazil” e “president”. Para “Brazil” apresentou-se que é um local e um país, além de um link para uma página da DBPedia, que contém mais informações sobre o país “Brazil”. Para a entidade “president”, a API também informou que se trata de um título de uma função, ou trabalho (“JobTitle”).

Veja e analise as outras características que a API identificou no texto.

```
response = natural_language_understanding.analyze(
  text='Steve Jobs is the founder of Apple',
  features=Features(
    entities=EntitiesOptions(),
    semantic_roles=SemanticRolesOptions(),
    ))

print(json.dumps(response, indent=2))
```

Código-Fonte 7.5. Outra chamada à API de NLU
Fonte: Desenvolvido pelo autor (2018)

Outra chamada, bem semelhante, para ilustrar a característica de papéis semânticos (*semantic roles*) é feita no código “Outra chamada à API de NLU”. A saída é exibida a seguir:

```
{
  "usage": {
    "text_characters": 34,
    "features": 2,
    "text_units": 1
  },
  "semantic_roles": [
    {
      "action": {
        "text": "is",
```

```
        "verb": {
          "text": "be",
          "tense": "present"
        },
        "normalized": "be"
      },
      "sentence": "Steve Jobs is the founder of Apple",
      "object": {
        "text": "the founder of Apple"
      },
      "subject": {
        "text": "Steve Jobs"
      }
    }
  ],
  "language": "en",
  "entities": [
    {
      "relevance": 0.33,
      "text": "Steve Jobs",
      "disambiguation": {
        "subtype": [
          "BoardMember",
          "CompanyFounder",
          "ComputerDesigner",
          "FilmProducer"
        ],
        "name": "Steve Jobs",
        "dbpedia_resource": "http://dbpedia.org/resource/Steve_Jobs"
      },
      "type": "Person",
      "count": 1
    },
    {
      "relevance": 0.33,
      "text": "founder",
      "type": "JobTitle",
      "count": 1
    },
    {
      "relevance": 0.33,
      "text": "Apple",
      "type": "Company",
      "count": 1
    }
  ]
}
```


Por exemplo, na análise de papéis semânticos (semantic_roles), o sujeito é "*Steve Jobs*", o verbo é "*is*" e o objeto é "*the founder of Apple*". Como entidades, são apresentadas "Steve Jobs", como uma pessoa, e "Apple", como uma companhia.

Já imaginou o trabalho de desenvolver um modelo com aprendizado de máquina para identificar tais tipos de informações em textos dos mais variados domínios? Essa é uma das grandes vantagens do uso de APIs já prontas, para as quais foram gastos muitos recursos para atingir tal grau de acurácia e maturidade.

Como o retorno já vem como um objeto json, quaisquer processos que deseja realizar com as informações retornadas é imediato no Python. Não é necessário ficar convertendo de um tipo para outro.

Teste com seus próprios textos. Pense em como utilizar as informações retornadas para tomada de decisão e programas que lidam com informações textuais.

Identificação da Língua do texto

A API de interpretação textual (NLU) faz a identificação automática da língua do texto passado como argumento. Se nós alterarmos o código para utilizar o texto “Steve Jobs é o fundador da Apple”, teremos o seguinte retorno:


```
{
  "usage": {
    "text_characters": 32,
    "features": 2,
    "text_units": 1
  },
  "semantic_roles": [],
  "language": "en",
  "entities": [
    {
      "relevance": 0.33,
      "text": "Steve Jobs",
      "disambiguation": {
        "subtype": [
          "BoardMember",
          "CompanyFounder",
          "ComputerDesigner",
          "FilmProducer"
        ],
        "name": "Steve Jobs",
        "dbpedia_resource": "http://dbpedia.org/resource/Steve_Jobs"
      },
      "type": "Person",
      "count": 1
    },
    {
      "relevance": 0.33,
      "text": "Apple",
      "type": "Company",
      "count": 1
    }
```

```
  ]
}
```

Observe que a API continua identificando o texto como sendo em Inglês (*language: "en"*), mas não consegue fazer a identificação dos papéis semânticos como no retorno do código "Outra chamada à API de NLU". Mesmo assim, identifica as mesmas duas entidades: "Steve Jobs" e "Apple".

Se fizermos outra requisição, com várias características (*features*), com um texto em Português, que seja identificado em Português, diversas características não irão funcionar. Veja no código a seguir:

```
response = natural_language_understanding.analyze(
  text='Na FIAP, os alunos são muito dedicados.',
  features=Features(
    relations=RelationsOptions(),
    concepts=ConceptsOptions(),
    emotion=EmotionOptions(),
    entities=EntitiesOptions(),
    semantic_roles=SemanticRolesOptions(),
    sentiment=SentimentOptions(),
    ))

print(json.dumps(response, indent=2))
```

Código-Fonte 7.6. Chamada à API com texto em Português
Fonte: Desenvolvido pelo autor (2018)

O retorno da última chamada é:

```
{
  "sentiment": {
    "document": {
      "score": 0.0,
      "label": "neutral"
    }
  },
  "language": "pt",
  "warnings": [
    "concepts: unsupported text language: pt",
    "emotion: unsupported text language: pt",
    "semantic_roles: unsupported text language: pt"
  ],
  "relations": [],
  "entities": [],
  "usage": {
    "text_characters": 39,
    "features": 3,
    "text_units": 1
  }
}
```

É apresentada uma chave "warnings" em que se diz que algumas características não são suportadas para o Português. Nesses casos, é possível o

desenvolvimento de modelos específicos para Línguas e para as características desejadas. Acesse o link a seguir para ter uma ideia de como fazer isso.

`https://console.bluemix.net/catalog/services/knowledge-studio`

Explore mais

O procedimento anterior é introdutório e, portanto, um tanto superficial. Se deseja ir mais a fundo, consulte o guia das APIs. Por exemplo, para a API explorada, de interpretação textual (*Natural Language Understanding*), você pode acessar o link a seguir e ver mais opções:

`https://www.ibm.com/watson/developercloud/natural-language-understanding/api/v1/`

Veja também as opções para o Processamento de Imagens em:

`https://www.ibm.com/watson/developercloud/visual-recognition/api/v3/python.html`

REFERÊNCIAS

IBM DEVELOPER. **API Explorer**. [s.d.]. Disponível em: <https://developer.ibm.com/api/list>. Acesso em: 24 abr. 2018.

RUSSEL, Stuart; NORVIG, Peter. **Inteligência Artificial.** Rio de Janeiro: Elsevier, 2013.

GLOSSÁRIO

API	Sigla para *Application Programming Interface*. Um conjunto de métodos para integração de software.