bookworm

August 21, 2017

1 Bookworm

1.1 Overview

In this project, you will build a simple question-answering agent that is able to learn from any text data you provide, and answer queries posed in natural language. You will use IBM Watson's cloud-based services to process the input text data and find relevant responses.

1.2 Learning Objectives

By completing this project, you will learn how to:

- Create a cloud-based NLP service instance and configure it.
- Ingest a set of text documents using the service and analyze the results.
- Accept questions in natural language and parse them.
- Find relevant answers from the preprocessed text data.

1.3 Getting Started

In order to use Watson's cloud-based services, you first need to create an account on the IBM Bluemix platform.

```
<div style="display: table-cell; width: 50%;">
    <img src="images/watson-logo.png" alt="IBM Watson logo" width="200" />
</div>
<div style="display: table-cell; width: 50%;">
    <img src="images/bluemix-logo.png" alt="IBM Bluemix logo" width="400" />
</div>
```

Then, for each service you want to use, you have to create an instance of that service. You can continue with the tasks below, and create a service instance when indicated.

1.4 1. Create and configure Discovery service

Create an instance of the **Discovery** service. You will use this to process a set of text documents, and *discover* relevant facts and relationships.

- Go to the IBM Bluemix Catalog.
- Select the service you want, **Discovery**, under the **Watson** category.

- Enter a Service Name for that instance, e.g. **Disco1** and a Credential Name, e.g. **Disco1-Creds** (these are just for you to be able to refer to later, they do not affect the functioning of the service).
- You should be able to see your newly-created service in your Services Dashboard.
- Open the service instance, click on the **Service credentials** tab, and then **View credentials** under Actions. This is where you will find the username and password to use when connecting to the service.

Save the credentials for the discovery service in a JSON file in the current directory named `service-credentials.json` with the following format:

```
{
    "discovery": {
        "username": "<your Discovery username here>",
        "password": "<your Discovery password here>"
    },
    "conversation": {
        "username": "",
        "password": ""
    }
}
```

You will be filling out the Conversation service credentials later, when you create an instance for it. Note that you should keep these credentials secret. Please do not turn them in with your submission!

1.4.1 Connect to the service instance

Let's connect to the service instance you just created using IBM Watson's Python SDK. You will first need to install the SDK:

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

Now execute each code cell below using `Shift+Enter`, and complete any steps indicated by a `TODO` comment. For more information on the Discovery service, please read the Documentation and look at the API Reference as needed.

```
In [1]: # Usual Python imports
        import sys
        import os
        import glob
        import json

        # BeautifulSoup, for parsing HTML
        from bs4 import BeautifulSoup

        # Matplotlib, for plotting
        import matplotlib.pyplot as plt
        %matplotlib inline
```

```
        # Watson Python SDK
        import watson_developer_cloud

        # Utility functions
        import helper
```

```
In [2]: # Connect to the Discovery service instance
        # TODO: Ensure that your username and password from the Service Credentials tab are in s
        # Note that these credentials are different from your IBM Bluemix login, and are specifi
        discovery_creds = helper.fetch_credentials('discovery')
        discovery = watson_developer_cloud.DiscoveryV1(
            version='2016-11-07',
            username=discovery_creds['username'],
            password=discovery_creds['password'])
```

1.4.2 Create an environment

The Discovery service organizes everything needed for a particular application in an *environment*. Let's create one called "Bookworm" for this project.

> ***Note**: It is okay to run this block multiple times - it will not create duplicate environments with the same name.*

```
In [3]: # Prepare an environment to work in
        env, env_id = helper.fetch_object(
            discovery, "environment", "Bookworm",
            create=True, create_args=dict(
                description="A space to read and understand stories",  # feel free to edit
                size=0  # use 0 for free plan (see API reference for more on sizing)
            ))
        print(json.dumps(env, indent=2))
```

```
Found environment: Bookworm (9c7d2980-8d3a-4dbb-80c9-1f7a5054edb7)
{
  "environment_id": "9c7d2980-8d3a-4dbb-80c9-1f7a5054edb7",
  "name": "Bookworm",
  "description": "A space to read and understand stories",
  "created": "2017-08-18T08:02:52.898Z",
  "updated": "2017-08-18T08:02:52.898Z",
  "status": "active",
  "read_only": false,
  "size": -1,
  "index_capacity": {
    "documents": {
      "available": 126,
      "maximum_allowed": 2000
    },
    "disk_usage": {
```

```
      "used_bytes": 11668587,
      "total_bytes": 200000000,
      "used": "11.669 MB",
      "total": "200 MB",
      "percent_used": 5.83
    },
    "memory_usage": {
      "used_bytes": -1,
      "total_bytes": -1,
      "used": "N/A",
      "total": "N/A",
      "percent_used": -1.0
    }
  }
}
```


1.4.3 Verify configuration options

A *configuration* defines what natural language processing routines are applied to any documents that are submitted to the service. Each environment gets a default configuration when it is created.

You can fetch the default configuration and view the different options using the following piece of code.

```
In [4]: # View default configuration
        cfg_id = discovery.get_default_configuration_id(environment_id=env_id)
        cfg = discovery.get_configuration(environment_id=env_id,
                                          configuration_id=cfg_id)
        print(json.dumps(cfg, indent=2))
```


```
{
  "configuration_id": "102e52ba-ec95-427e-bb88-b3f978034f76",
  "name": "Default Configuration",
  "description": "The configuration used by default when creating a new collection without speci
  "created": "2017-08-18T08:02:52.937Z",
  "updated": "2017-08-18T08:02:52.937Z",
  "conversions": {
    "word": {
      "heading": {
        "fonts": [
          {
            "level": 1,
            "min_size": 24,
            "bold": false,
            "italic": false
          },
          {
            "level": 2,
```

```
        "min_size": 18,
        "max_size": 23,
        "bold": true,
        "italic": false
      },
      {
        "level": 3,
        "min_size": 14,
        "max_size": 17,
        "bold": false,
        "italic": false
      },
      {
        "level": 4,
        "min_size": 13,
        "max_size": 13,
        "bold": true,
        "italic": false
      }
    ],
    "styles": [
      {
        "level": 1,
        "names": [
          "pullout heading",
          "pulloutheading",
          "header"
        ]
      },
      {
        "level": 2,
        "names": [
          "subtitle"
        ]
      }
    ]
  }
},
"pdf": {
  "heading": {
    "fonts": [
      {
        "level": 1,
        "min_size": 24,
        "max_size": 80
      },
      {
        "level": 2,
```

```
        "min_size": 18,
        "max_size": 24,
        "bold": false,
        "italic": false
      },
      {
        "level": 2,
        "min_size": 18,
        "max_size": 24,
        "bold": true
      },
      {
        "level": 3,
        "min_size": 13,
        "max_size": 18,
        "bold": false,
        "italic": false
      },
      {
        "level": 3,
        "min_size": 13,
        "max_size": 18,
        "bold": true
      },
      {
        "level": 4,
        "min_size": 11,
        "max_size": 13,
        "bold": false,
        "italic": false
      }
    ]
  }
},
"html": {
  "exclude_tags_completely": [
    "script",
    "sup"
  ],
  "exclude_tags_keep_content": [
    "font",
    "em",
    "span"
  ],
  "exclude_content": {
    "xpaths": []
  },
  "keep_content": {
```

```
        "xpaths": []
      },
      "exclude_tag_attributes": [
        "EVENT_ACTIONS"
      ]
    },
    "json_normalizations": []
  },
  "enrichments": [
    {
      "destination_field": "enriched_text",
      "source_field": "text",
      "enrichment": "alchemy_language",
      "options": {
        "extract": "keyword, entity, doc-sentiment, taxonomy, concept, relation",
        "sentiment": true,
        "quotations": true
      }
    }
  ],
  "normalizations": []
}
```

There are 3 main configuration blocks that affect how input documents are processed:

1. `conversions`: How to convert documents in various formats (Word, PDF, HTML) and extract elements that indicate some structure (e.g. headings).
2. `enrichments`: What NLP output results are we interested in (keywords, entities, sentiment, etc.).
3. `normalizations`: Post-processing steps to be applied to the output. This can be left empty in most cases, unless you need the output to be normalized into a very specific format.

***Note**: The default configuration for an environment cannot be modified. If you need to change any of the options, you will need to create a new one, and then edit it. The easiest way to do this is using the service dashboard, which is described later.*

1.4.4 Test your configuration

It is a good idea to test your configuration on a small sample text before you apply it to a larger document collection.

***Note**: We have supplied a sample document (`data/sample.html`) containing the opening crawl text for Star Wars: Episode IV, but you are free to use a text of your choosing.*

Q: (optional) If you use your own sample text, provide a brief title and description below.

A: NA

```
In [5]: # Test configuration on some sample text
        data_dir = "data"
        filename = os.path.join(data_dir, "sample.html")
```

```
        with open(filename, "r") as f:
            res = discovery.test_document(environment_id=env_id,
                                          configuration_id=cfg_id,
                                          fileinfo=f)
        print(json.dumps(res, indent=2))
```


```
{
  "status": "completed",
  "original_media_type": "text/html",
  "snapshots": [
    {
      "step": "html_input",
      "snapshot": {
        "html": "<html>\n<head>\n    <title>Star Wars: Episode IV - A New Hope (Opening Crawl)</
      }
    },
    {
      "step": "html_output",
      "snapshot": {
        "html": "<?xml version='1.0' encoding='UTF-8' standalone='yes'?><html>\n<head>\n    <met
      }
    },
    {
      "step": "json_output",
      "snapshot": {
        "extracted_metadata": {
          "title": "Star Wars: Episode IV - A New Hope (Opening Crawl)"
        },
        "html": "<?xml version='1.0' encoding='UTF-8' standalone='yes'?><html>\n<head>\n    <met
        "text": "Star Wars: Episode IV - A New Hope (Opening Crawl)\n\nStar Wars: Episode IV - A
        "metadata": {}
      }
    },
    {
      "step": "json_normalizations_output",
      "snapshot": {
        "extracted_metadata": {
          "title": "Star Wars: Episode IV - A New Hope (Opening Crawl)"
        },
        "html": "<?xml version='1.0' encoding='UTF-8' standalone='yes'?><html>\n<head>\n    <met
        "text": "Star Wars: Episode IV - A New Hope (Opening Crawl)\n\nStar Wars: Episode IV - A
        "metadata": {}
      }
    },
    {
      "step": "enrichments_output",
      "snapshot": {
        "extracted_metadata": {
```

```
  "title": "Star Wars: Episode IV - A New Hope (Opening Crawl)"
},
"html": "<?xml version='1.0' encoding='UTF-8' standalone='yes'?><html>\n<head>\n    <met
"text": "Star Wars: Episode IV - A New Hope (Opening Crawl)\n\nStar Wars: Episode IV - A
"metadata": {},
"enriched_text": {
  "status": "OK",
  "language": "english",
  "docSentiment": {
    "type": "negative",
    "score": -0.575489,
    "mixed": false
  },
  "concepts": [
    {
      "text": "Star Wars Episode IV: A New Hope",
      "relevance": 0.98887,
      "website": "http://www.starwars.com/movies/episode-iv",
      "dbpedia": "http://dbpedia.org/resource/Star_Wars_Episode_IV:_A_New_Hope",
      "freebase": "http://rdf.freebase.com/ns/m.0dtfn",
      "opencyc": "http://sw.opencyc.org/concept/Mx4rwVH1ipwpEbGdrcN5Y29ycA"
    },
    {
      "text": "Star Wars",
      "relevance": 0.985705,
      "website": "http://www.starwars.com",
      "dbpedia": "http://dbpedia.org/resource/Star_Wars",
      "freebase": "http://rdf.freebase.com/ns/m.06mmr",
      "yago": "http://yago-knowledge.org/resource/Star_Wars"
    },
    {
      "text": "Rebel Alliance",
      "relevance": 0.90639,
      "dbpedia": "http://dbpedia.org/resource/Rebel_Alliance",
      "freebase": "http://rdf.freebase.com/ns/m.0nr1k",
      "yago": "http://yago-knowledge.org/resource/Rebel_Alliance"
    },
    {
      "text": "Luke Skywalker",
      "relevance": 0.805771,
      "dbpedia": "http://dbpedia.org/resource/Luke_Skywalker",
      "freebase": "http://rdf.freebase.com/ns/m.0f1bg",
      "yago": "http://yago-knowledge.org/resource/Luke_Skywalker"
    },
    {
      "text": "Star Wars Episode V: The Empire Strikes Back",
      "relevance": 0.801062,
      "website": "http://www.starwars.com/movies/episode-v",
```

```
  "dbpedia": "http://dbpedia.org/resource/Star_Wars_Episode_V:_The_Empire_Strikes_Ba
  "freebase": "http://rdf.freebase.com/ns/m.0f3m1"
},
{
  "text": "Darth Vader",
  "relevance": 0.777337,
  "dbpedia": "http://dbpedia.org/resource/Darth_Vader",
  "freebase": "http://rdf.freebase.com/ns/m.0f2y0",
  "yago": "http://yago-knowledge.org/resource/Darth_Vader"
},
{
  "text": "Star Wars Episode VI: Return of the Jedi",
  "relevance": 0.77159,
  "website": "http://www.starwars.com/movies/episode-vi",
  "dbpedia": "http://dbpedia.org/resource/Star_Wars_Episode_VI:_Return_of_the_Jedi",
  "freebase": "http://rdf.freebase.com/ns/m.0ddjy"
},
{
  "text": "Grand Moff Tarkin",
  "relevance": 0.748553,
  "dbpedia": "http://dbpedia.org/resource/Grand_Moff_Tarkin",
  "freebase": "http://rdf.freebase.com/ns/m.0jrsz",
  "yago": "http://yago-knowledge.org/resource/Grand_Moff_Tarkin"
},
{
  "text": "Star Wars Episode III: Revenge of the Sith",
  "relevance": 0.697065,
  "dbpedia": "http://dbpedia.org/resource/Star_Wars_Episode_III:_Revenge_of_the_Sith
  "freebase": "http://rdf.freebase.com/ns/m.0fdv3"
},
{
  "text": "Princess Leia Organa",
  "relevance": 0.607163,
  "dbpedia": "http://dbpedia.org/resource/Princess_Leia_Organa",
  "yago": "http://yago-knowledge.org/resource/Princess_Leia_Organa"
},
{
  "text": "Death Star",
  "relevance": 0.59271,
  "dbpedia": "http://dbpedia.org/resource/Death_Star",
  "freebase": "http://rdf.freebase.com/ns/m.0f325",
  "yago": "http://yago-knowledge.org/resource/Death_Star"
},
{
  "text": "Jabba the Hutt",
  "relevance": 0.543114,
  "dbpedia": "http://dbpedia.org/resource/Jabba_the_Hutt",
  "freebase": "http://rdf.freebase.com/ns/m.0fjms",
```

```
  "yago": "http://yago-knowledge.org/resource/Jabba_the_Hutt"
},
{
  "text": "Galactic Empire",
  "relevance": 0.523169,
  "dbpedia": "http://dbpedia.org/resource/Galactic_Empire_(Star_Wars)",
  "freebase": "http://rdf.freebase.com/ns/m.0hwyg",
  "yago": "http://yago-knowledge.org/resource/Galactic_Empire_(Star_Wars)"
},
{
  "text": "Star Wars galaxy",
  "relevance": 0.519032,
  "dbpedia": "http://dbpedia.org/resource/Star_Wars_galaxy",
  "yago": "http://yago-knowledge.org/resource/Star_Wars_galaxy"
},
{
  "text": "Star Wars Expanded Universe",
  "relevance": 0.507885,
  "website": "http://www.starwars.com/eu",
  "dbpedia": "http://dbpedia.org/resource/Star_Wars_Expanded_Universe",
  "freebase": "http://rdf.freebase.com/ns/m.01qbrc",
  "yago": "http://yago-knowledge.org/resource/Star_Wars_Expanded_Universe"
},
{
  "text": "Alderaan",
  "relevance": 0.503196,
  "dbpedia": "http://dbpedia.org/resource/Alderaan",
  "freebase": "http://rdf.freebase.com/ns/m.065lhy",
  "yago": "http://yago-knowledge.org/resource/Alderaan"
},
{
  "text": "Carrie Fisher",
  "relevance": 0.481464,
  "dbpedia": "http://dbpedia.org/resource/Carrie_Fisher",
  "freebase": "http://rdf.freebase.com/ns/m.01tnbn",
  "opencyc": "http://sw.opencyc.org/concept/Mx4rwSJnvZwpEbGdrcN5Y29ycA",
  "yago": "http://yago-knowledge.org/resource/Carrie_Fisher"
},
{
  "text": "Jedi",
  "relevance": 0.474365,
  "dbpedia": "http://dbpedia.org/resource/Jedi",
  "freebase": "http://rdf.freebase.com/ns/m.045n4",
  "opencyc": "http://sw.opencyc.org/concept/Mx4rwNLAD5wpEbGdrcN5Y29ycA",
  "yago": "http://yago-knowledge.org/resource/Jedi"
},
{
  "text": "The Star Wars Holiday Special",
```

```
  "relevance": 0.4659,
  "dbpedia": "http://dbpedia.org/resource/The_Star_Wars_Holiday_Special",
  "freebase": "http://rdf.freebase.com/ns/m.0199wf",
  "yago": "http://yago-knowledge.org/resource/The_Star_Wars_Holiday_Special"
},
{
  "text": "Wedge Antilles",
  "relevance": 0.46456,
  "dbpedia": "http://dbpedia.org/resource/Wedge_Antilles",
  "freebase": "http://rdf.freebase.com/ns/m.0ddp_",
  "yago": "http://yago-knowledge.org/resource/Wedge_Antilles"
},
{
  "text": "English-language films",
  "relevance": 0.460469,
  "dbpedia": "http://dbpedia.org/resource/English-language_films"
},
{
  "text": "Padm\u00e9 Amidala",
  "relevance": 0.453766,
  "dbpedia": "http://dbpedia.org/resource/Padm\u00e9_Amidala",
  "freebase": "http://rdf.freebase.com/ns/m.0drf_",
  "yago": "http://yago-knowledge.org/resource/Padm%C3%A9_Amidala"
},
{
  "text": "Galactic Civil War",
  "relevance": 0.447243,
  "dbpedia": "http://dbpedia.org/resource/Galactic_Civil_War",
  "freebase": "http://rdf.freebase.com/ns/m.01qc6_",
  "yago": "http://yago-knowledge.org/resource/Galactic_Civil_War"
},
{
  "text": "Spaceballs",
  "relevance": 0.436864,
  "dbpedia": "http://dbpedia.org/resource/Spaceballs",
  "freebase": "http://rdf.freebase.com/ns/m.0j_nw",
  "yago": "http://yago-knowledge.org/resource/Spaceballs"
},
{
  "text": "Clone Wars",
  "relevance": 0.43403,
  "dbpedia": "http://dbpedia.org/resource/Clone_Wars_(Star_Wars)",
  "freebase": "http://rdf.freebase.com/ns/m.01qc6n",
  "yago": "http://yago-knowledge.org/resource/Clone_Wars_(Star_Wars)"
},
{
  "text": "C-3PO",
  "relevance": 0.432345,
```

```
      "dbpedia": "http://dbpedia.org/resource/C-3PO",
      "freebase": "http://rdf.freebase.com/ns/m.0g_sv",
      "yago": "http://yago-knowledge.org/resource/C-3PO"
    },
    {
      "text": "Obi-Wan Kenobi",
      "relevance": 0.431988,
      "dbpedia": "http://dbpedia.org/resource/Obi-Wan_Kenobi",
      "freebase": "http://rdf.freebase.com/ns/m.0fkm7",
      "yago": "http://yago-knowledge.org/resource/Obi-Wan_Kenobi"
    },
    {
      "text": "R2-D2",
      "relevance": 0.430085,
      "dbpedia": "http://dbpedia.org/resource/R2-D2",
      "freebase": "http://rdf.freebase.com/ns/m.0g_rg",
      "yago": "http://yago-knowledge.org/resource/R2-D2"
    }
  ],
  "entities": [
    {
      "type": "FieldTerminology",
      "relevance": 0.972336,
      "sentiment": {
        "type": "negative",
        "score": -0.409172,
        "mixed": false
      },
      "count": 1,
      "text": "civil war"
    }
  ],
  "relations": [
    {
      "sentence": " It is a period of civil war.",
      "subject": {
        "text": "It"
      },
      "action": {
        "text": "is",
        "lemmatized": "be",
        "verb": {
          "text": "be",
          "tense": "present"
        }
      },
      "object": {
        "text": "a period of civil war",
```

```
    "sentiment": {
      "type": "negative",
      "score": -0.564509,
      "mixed": false
    },
    "entities": [
      {
        "type": "FieldTerminology",
        "text": "civil war"
      }
    ],
    "keywords": [
      {
        "text": "civil war"
      },
      {
        "text": "period"
      }
    ]
  }
},
{
  "sentence": " Rebel spaceships, striking from a hidden base, have won their first
  "subject": {
    "text": "Rebel spaceships, striking from a hidden base,",
    "keywords": [
      {
        "text": "Rebel spaceships"
      },
      {
        "text": "hidden base"
      }
    ]
  },
  "action": {
    "text": "have won",
    "lemmatized": "have win",
    "verb": {
      "text": "win",
      "tense": "past"
    }
  },
  "object": {
    "text": "their first victory",
    "keywords": [
      {
        "text": "victory"
      }
```

```
    ]
  }
},
{
  "sentence": " During the battle, Rebel spies managed to steal secret plans to the
  "subject": {
    "text": "Rebel spies",
    "keywords": [
      {
        "text": "Rebel spies"
      }
    ]
  },
  "action": {
    "text": "managed",
    "lemmatized": "manage",
    "verb": {
      "text": "manage",
      "tense": "past"
    }
  },
  "object": {
    "text": "to steal secret plans to the Empire's ultimate weapon, the DEATH STAR,
    "sentiment": {
      "type": "negative",
      "score": -0.596586,
      "mixed": false
    },
    "keywords": [
      {
        "text": "armored space station"
      },
      {
        "text": "ultimate weapon"
      },
      {
        "text": "secret plans"
      },
      {
        "text": "entire planet"
      }
    ]
  }
},
{
  "sentence": " During the battle, Rebel spies managed to steal secret plans to the
  "subject": {
    "text": "Rebel spies",
```

```
      "keywords": [
        {
          "text": "Rebel spies"
        }
      ]
    },
    "action": {
      "text": "managed to steal",
      "lemmatized": "manage to steal",
      "verb": {
        "text": "steal",
        "tense": "future"
      }
    },
    "object": {
      "text": "secret plans",
      "keywords": [
        {
          "text": "secret plans"
        }
      ]
    }
  },
  {
    "sentence": " During the battle, Rebel spies managed to steal secret plans to the
    "subject": {
      "text": "Rebel spies",
      "keywords": [
        {
          "text": "Rebel spies"
        }
      ]
    },
    "action": {
      "text": "to destroy",
      "lemmatized": "to destroy",
      "verb": {
        "text": "destroy",
        "tense": "future"
      }
    },
    "object": {
      "text": "an entire planet",
      "sentiment": {
        "type": "positive",
        "score": 0.748639,
        "mixed": false
      },
```

```
    "sentimentFromSubject": {
      "type": "negative",
      "score": -0.754579,
      "mixed": false
    },
    "keywords": [
      {
        "text": "entire planet"
      }
    ]
  }
},
{
  "sentence": " Pursued by the Empire's sinister agents, Princess Leia races home ab
  "subject": {
    "text": "by the Empire",
    "sentiment": {
      "type": "positive",
      "score": 0.554106,
      "mixed": false
    },
    "keywords": [
      {
        "text": "Empire"
      }
    ]
  },
  "action": {
    "text": "Pursued",
    "lemmatized": "Pursued",
    "verb": {
      "text": "Pursued",
      "tense": "past"
    }
  },
  "object": {
    "text": "Princess Leia",
    "sentiment": {
      "type": "positive",
      "score": 0.682563,
      "mixed": false
    },
    "keywords": [
      {
        "text": "Princess Leia"
      }
    ]
  }
```

```
},
{
  "sentence": " Pursued by the Empire's sinister agents, Princess Leia races home ab
  "subject": {
    "text": "the stolen plans",
    "sentiment": {
      "type": "negative",
      "score": -0.44097,
      "mixed": false
    },
    "keywords": [
      {
        "text": "plans"
      }
    ]
  },
  "action": {
    "text": "can save",
    "lemmatized": "can save",
    "verb": {
      "text": "save",
      "tense": "future"
    }
  },
  "object": {
    "text": "her people",
    "keywords": [
      {
        "text": "people"
      }
    ]
  }
},
{
  "sentence": " Pursued by the Empire's sinister agents, Princess Leia races home ab
  "subject": {
    "text": "freedom",
    "keywords": [
      {
        "text": "freedom"
      }
    ]
  },
  "action": {
    "text": "restore",
    "lemmatized": "restore",
    "verb": {
      "text": "restore",
```

```
        "tense": "future"
      }
    },
    "object": {
      "text": "to the galaxy",
      "sentimentFromSubject": {
        "type": "positive",
        "score": 0.698799,
        "mixed": false
      },
      "keywords": [
        {
          "text": "galaxy"
        }
      ]
    }
  }
],
"taxonomy": [
  {
    "label": "/art and entertainment/movies and tv/movies",
    "score": 0.584247,
    "confident": false
  },
  {
    "label": "/society/unrest and war",
    "score": 0.517031,
    "confident": false
  },
  {
    "confident": false,
    "label": "/law, govt and politics/armed forces/army",
    "score": 0.215561
  }
],
"keywords": [
  {
    "relevance": 0.920159,
    "sentiment": {
      "type": "neutral",
      "mixed": false
    },
    "text": "Opening Crawl"
  },
  {
    "relevance": 0.785031,
    "sentiment": {
      "score": 0.447888,
```

```
    "type": "positive",
    "mixed": false
  },
  "text": "New Hope"
},
{
  "relevance": 0.775799,
  "sentiment": {
    "score": 0.40059,
    "type": "positive",
    "mixed": false
  },
  "text": "Princess Leia races"
},
{
  "relevance": 0.771384,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "Star Wars"
},
{
  "relevance": 0.767527,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "evil Galactic Empire"
},
{
  "relevance": 0.747615,
  "sentiment": {
    "score": -0.494714,
    "type": "negative",
    "mixed": false
  },
  "text": "armored space station"
},
{
  "relevance": 0.669776,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "Rebel spaceships"
},
{
```

```
  "relevance": 0.656828,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "Rebel spies"
},
{
  "relevance": 0.609404,
  "sentiment": {
    "score": 0.484685,
    "type": "positive",
    "mixed": false
  },
  "text": "sinister agents"
},
{
  "relevance": 0.600682,
  "sentiment": {
    "score": -0.409172,
    "type": "negative",
    "mixed": false
  },
  "text": "civil war"
},
{
  "relevance": 0.597836,
  "sentiment": {
    "score": 0.488854,
    "type": "positive",
    "mixed": false
  },
  "text": "hidden base"
},
{
  "relevance": 0.594694,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "ultimate weapon"
},
{
  "relevance": 0.582765,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
```

```
  "text": "secret plans"
},
{
  "relevance": 0.582596,
  "sentiment": {
    "score": -0.494714,
    "type": "negative",
    "mixed": false
  },
  "text": "entire planet"
},
{
  "relevance": 0.574413,
  "sentiment": {
    "score": -0.430026,
    "type": "negative",
    "mixed": false
  },
  "text": "DEATH STAR"
},
{
  "relevance": 0.465394,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "custodian"
},
{
  "relevance": 0.462247,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "Episode"
},
{
  "relevance": 0.443969,
  "sentiment": {
    "score": 0.40059,
    "type": "positive",
    "mixed": false
  },
  "text": "starship"
},
{
  "relevance": 0.441799,
  "sentiment": {
```

```
    "type": "neutral",
    "mixed": false
  },
  "text": "victory"
},
{
  "relevance": 0.437824,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "galaxy"
},
{
  "relevance": 0.432092,
  "sentiment": {
    "score": -0.409172,
    "type": "negative",
    "mixed": false
  },
  "text": "period"
},
{
  "relevance": 0.429238,
  "sentiment": {
    "score": -0.37594,
    "type": "negative",
    "mixed": false
  },
  "text": "battle"
},
{
  "relevance": 0.428916,
  "sentiment": {
    "score": 0.40059,
    "type": "positive",
    "mixed": false
  },
  "text": "home"
},
{
  "relevance": 0.428498,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "people"
}
```

```
      ]
    }
  }
},
{
  "step": "normalizations_output",
  "snapshot": {
    "extracted_metadata": {
      "title": "Star Wars: Episode IV - A New Hope (Opening Crawl)"
    },
    "html": "<?xml version='1.0' encoding='UTF-8' standalone='yes'?><html>\n<head>\n    <met
    "text": "Star Wars: Episode IV - A New Hope (Opening Crawl)\n\nStar Wars: Episode IV - A
    "metadata": {},
    "enriched_text": {
      "status": "OK",
      "language": "english",
      "docSentiment": {
        "type": "negative",
        "score": -0.575489,
        "mixed": false
      },
      "concepts": [
        {
          "text": "Star Wars Episode IV: A New Hope",
          "relevance": 0.98887,
          "website": "http://www.starwars.com/movies/episode-iv",
          "dbpedia": "http://dbpedia.org/resource/Star_Wars_Episode_IV:_A_New_Hope",
          "freebase": "http://rdf.freebase.com/ns/m.0dtfn",
          "opencyc": "http://sw.opencyc.org/concept/Mx4rwVH1ipwpEbGdrcN5Y29ycA"
        },
        {
          "text": "Star Wars",
          "relevance": 0.985705,
          "website": "http://www.starwars.com",
          "dbpedia": "http://dbpedia.org/resource/Star_Wars",
          "freebase": "http://rdf.freebase.com/ns/m.06mmr",
          "yago": "http://yago-knowledge.org/resource/Star_Wars"
        },
        {
          "text": "Rebel Alliance",
          "relevance": 0.90639,
          "dbpedia": "http://dbpedia.org/resource/Rebel_Alliance",
          "freebase": "http://rdf.freebase.com/ns/m.0nr1k",
          "yago": "http://yago-knowledge.org/resource/Rebel_Alliance"
        },
        {
          "text": "Luke Skywalker",
          "relevance": 0.805771,
```

```
  "dbpedia": "http://dbpedia.org/resource/Luke_Skywalker",
  "freebase": "http://rdf.freebase.com/ns/m.0f1bg",
  "yago": "http://yago-knowledge.org/resource/Luke_Skywalker"
},
{
  "text": "Star Wars Episode V: The Empire Strikes Back",
  "relevance": 0.801062,
  "website": "http://www.starwars.com/movies/episode-v",
  "dbpedia": "http://dbpedia.org/resource/Star_Wars_Episode_V:_The_Empire_Strikes_Ba
  "freebase": "http://rdf.freebase.com/ns/m.0f3m1"
},
{
  "text": "Darth Vader",
  "relevance": 0.777337,
  "dbpedia": "http://dbpedia.org/resource/Darth_Vader",
  "freebase": "http://rdf.freebase.com/ns/m.0f2y0",
  "yago": "http://yago-knowledge.org/resource/Darth_Vader"
},
{
  "text": "Star Wars Episode VI: Return of the Jedi",
  "relevance": 0.77159,
  "website": "http://www.starwars.com/movies/episode-vi",
  "dbpedia": "http://dbpedia.org/resource/Star_Wars_Episode_VI:_Return_of_the_Jedi",
  "freebase": "http://rdf.freebase.com/ns/m.0ddjy"
},
{
  "text": "Grand Moff Tarkin",
  "relevance": 0.748553,
  "dbpedia": "http://dbpedia.org/resource/Grand_Moff_Tarkin",
  "freebase": "http://rdf.freebase.com/ns/m.0jrsz",
  "yago": "http://yago-knowledge.org/resource/Grand_Moff_Tarkin"
},
{
  "text": "Star Wars Episode III: Revenge of the Sith",
  "relevance": 0.697065,
  "dbpedia": "http://dbpedia.org/resource/Star_Wars_Episode_III:_Revenge_of_the_Sith
  "freebase": "http://rdf.freebase.com/ns/m.0fdv3"
},
{
  "text": "Princess Leia Organa",
  "relevance": 0.607163,
  "dbpedia": "http://dbpedia.org/resource/Princess_Leia_Organa",
  "yago": "http://yago-knowledge.org/resource/Princess_Leia_Organa"
},
{
  "text": "Death Star",
  "relevance": 0.59271,
  "dbpedia": "http://dbpedia.org/resource/Death_Star",
```

```
  "freebase": "http://rdf.freebase.com/ns/m.0f325",
  "yago": "http://yago-knowledge.org/resource/Death_Star"
},
{
  "text": "Jabba the Hutt",
  "relevance": 0.543114,
  "dbpedia": "http://dbpedia.org/resource/Jabba_the_Hutt",
  "freebase": "http://rdf.freebase.com/ns/m.0fjms",
  "yago": "http://yago-knowledge.org/resource/Jabba_the_Hutt"
},
{
  "text": "Galactic Empire",
  "relevance": 0.523169,
  "dbpedia": "http://dbpedia.org/resource/Galactic_Empire_(Star_Wars)",
  "freebase": "http://rdf.freebase.com/ns/m.0hwyg",
  "yago": "http://yago-knowledge.org/resource/Galactic_Empire_(Star_Wars)"
},
{
  "text": "Star Wars galaxy",
  "relevance": 0.519032,
  "dbpedia": "http://dbpedia.org/resource/Star_Wars_galaxy",
  "yago": "http://yago-knowledge.org/resource/Star_Wars_galaxy"
},
{
  "text": "Star Wars Expanded Universe",
  "relevance": 0.507885,
  "website": "http://www.starwars.com/eu",
  "dbpedia": "http://dbpedia.org/resource/Star_Wars_Expanded_Universe",
  "freebase": "http://rdf.freebase.com/ns/m.01qbrc",
  "yago": "http://yago-knowledge.org/resource/Star_Wars_Expanded_Universe"
},
{
  "text": "Alderaan",
  "relevance": 0.503196,
  "dbpedia": "http://dbpedia.org/resource/Alderaan",
  "freebase": "http://rdf.freebase.com/ns/m.065lhy",
  "yago": "http://yago-knowledge.org/resource/Alderaan"
},
{
  "text": "Carrie Fisher",
  "relevance": 0.481464,
  "dbpedia": "http://dbpedia.org/resource/Carrie_Fisher",
  "freebase": "http://rdf.freebase.com/ns/m.01tnbn",
  "opencyc": "http://sw.opencyc.org/concept/Mx4rwSJnvZwpEbGdrcN5Y29ycA",
  "yago": "http://yago-knowledge.org/resource/Carrie_Fisher"
},
{
  "text": "Jedi",
```

```
  "relevance": 0.474365,
  "dbpedia": "http://dbpedia.org/resource/Jedi",
  "freebase": "http://rdf.freebase.com/ns/m.045n4",
  "opencyc": "http://sw.opencyc.org/concept/Mx4rwNLAD5wpEbGdrcN5Y29ycA",
  "yago": "http://yago-knowledge.org/resource/Jedi"
},
{
  "text": "The Star Wars Holiday Special",
  "relevance": 0.4659,
  "dbpedia": "http://dbpedia.org/resource/The_Star_Wars_Holiday_Special",
  "freebase": "http://rdf.freebase.com/ns/m.0199wf",
  "yago": "http://yago-knowledge.org/resource/The_Star_Wars_Holiday_Special"
},
{
  "text": "Wedge Antilles",
  "relevance": 0.46456,
  "dbpedia": "http://dbpedia.org/resource/Wedge_Antilles",
  "freebase": "http://rdf.freebase.com/ns/m.0ddp_",
  "yago": "http://yago-knowledge.org/resource/Wedge_Antilles"
},
{
  "text": "English-language films",
  "relevance": 0.460469,
  "dbpedia": "http://dbpedia.org/resource/English-language_films"
},
{
  "text": "Padm\u00e9 Amidala",
  "relevance": 0.453766,
  "dbpedia": "http://dbpedia.org/resource/Padm\u00e9_Amidala",
  "freebase": "http://rdf.freebase.com/ns/m.0drf_",
  "yago": "http://yago-knowledge.org/resource/Padm%C3%A9_Amidala"
},
{
  "text": "Galactic Civil War",
  "relevance": 0.447243,
  "dbpedia": "http://dbpedia.org/resource/Galactic_Civil_War",
  "freebase": "http://rdf.freebase.com/ns/m.01qc6_",
  "yago": "http://yago-knowledge.org/resource/Galactic_Civil_War"
},
{
  "text": "Spaceballs",
  "relevance": 0.436864,
  "dbpedia": "http://dbpedia.org/resource/Spaceballs",
  "freebase": "http://rdf.freebase.com/ns/m.0j_nw",
  "yago": "http://yago-knowledge.org/resource/Spaceballs"
},
{
  "text": "Clone Wars",
```

```
      "relevance": 0.43403,
      "dbpedia": "http://dbpedia.org/resource/Clone_Wars_(Star_Wars)",
      "freebase": "http://rdf.freebase.com/ns/m.01qc6n",
      "yago": "http://yago-knowledge.org/resource/Clone_Wars_(Star_Wars)"
    },
    {
      "text": "C-3PO",
      "relevance": 0.432345,
      "dbpedia": "http://dbpedia.org/resource/C-3PO",
      "freebase": "http://rdf.freebase.com/ns/m.0g_sv",
      "yago": "http://yago-knowledge.org/resource/C-3PO"
    },
    {
      "text": "Obi-Wan Kenobi",
      "relevance": 0.431988,
      "dbpedia": "http://dbpedia.org/resource/Obi-Wan_Kenobi",
      "freebase": "http://rdf.freebase.com/ns/m.0fkm7",
      "yago": "http://yago-knowledge.org/resource/Obi-Wan_Kenobi"
    },
    {
      "text": "R2-D2",
      "relevance": 0.430085,
      "dbpedia": "http://dbpedia.org/resource/R2-D2",
      "freebase": "http://rdf.freebase.com/ns/m.0g_rg",
      "yago": "http://yago-knowledge.org/resource/R2-D2"
    }
  ],
  "entities": [
    {
      "type": "FieldTerminology",
      "relevance": 0.972336,
      "sentiment": {
        "type": "negative",
        "score": -0.409172,
        "mixed": false
      },
      "count": 1,
      "text": "civil war"
    }
  ],
  "relations": [
    {
      "sentence": " It is a period of civil war.",
      "subject": {
        "text": "It"
      },
      "action": {
        "text": "is",
```

```
      "lemmatized": "be",
      "verb": {
        "text": "be",
        "tense": "present"
      }
    },
    "object": {
      "text": "a period of civil war",
      "sentiment": {
        "type": "negative",
        "score": -0.564509,
        "mixed": false
      },
      "entities": [
        {
          "type": "FieldTerminology",
          "text": "civil war"
        }
      ],
      "keywords": [
        {
          "text": "civil war"
        },
        {
          "text": "period"
        }
      ]
    }
  },
  {
    "sentence": " Rebel spaceships, striking from a hidden base, have won their first
    "subject": {
      "text": "Rebel spaceships, striking from a hidden base,",
      "keywords": [
        {
          "text": "Rebel spaceships"
        },
        {
          "text": "hidden base"
        }
      ]
    },
    "action": {
      "text": "have won",
      "lemmatized": "have win",
      "verb": {
        "text": "win",
        "tense": "past"
```

```
      }
    },
    "object": {
      "text": "their first victory",
      "keywords": [
        {
          "text": "victory"
        }
      ]
    }
  },
  {
    "sentence": " During the battle, Rebel spies managed to steal secret plans to the
    "subject": {
      "text": "Rebel spies",
      "keywords": [
        {
          "text": "Rebel spies"
        }
      ]
    },
    "action": {
      "text": "managed",
      "lemmatized": "manage",
      "verb": {
        "text": "manage",
        "tense": "past"
      }
    },
    "object": {
      "text": "to steal secret plans to the Empire's ultimate weapon, the DEATH STAR,
      "sentiment": {
        "type": "negative",
        "score": -0.596586,
        "mixed": false
      },
      "keywords": [
        {
          "text": "armored space station"
        },
        {
          "text": "ultimate weapon"
        },
        {
          "text": "secret plans"
        },
        {
          "text": "entire planet"
```

```
      }
    ]
  }
},
{
  "sentence": " During the battle, Rebel spies managed to steal secret plans to the
  "subject": {
    "text": "Rebel spies",
    "keywords": [
      {
        "text": "Rebel spies"
      }
    ]
  },
  "action": {
    "text": "managed to steal",
    "lemmatized": "manage to steal",
    "verb": {
      "text": "steal",
      "tense": "future"
    }
  },
  "object": {
    "text": "secret plans",
    "keywords": [
      {
        "text": "secret plans"
      }
    ]
  }
},
{
  "sentence": " During the battle, Rebel spies managed to steal secret plans to the
  "subject": {
    "text": "Rebel spies",
    "keywords": [
      {
        "text": "Rebel spies"
      }
    ]
  },
  "action": {
    "text": "to destroy",
    "lemmatized": "to destroy",
    "verb": {
      "text": "destroy",
      "tense": "future"
    }
```

```
    },
    "object": {
      "text": "an entire planet",
      "sentiment": {
        "type": "positive",
        "score": 0.748639,
        "mixed": false
      },
      "sentimentFromSubject": {
        "type": "negative",
        "score": -0.754579,
        "mixed": false
      },
      "keywords": [
        {
          "text": "entire planet"
        }
      ]
    }
  },
  {
    "sentence": " Pursued by the Empire's sinister agents, Princess Leia races home ab
    "subject": {
      "text": "by the Empire",
      "sentiment": {
        "type": "positive",
        "score": 0.554106,
        "mixed": false
      },
      "keywords": [
        {
          "text": "Empire"
        }
      ]
    },
    "action": {
      "text": "Pursued",
      "lemmatized": "Pursued",
      "verb": {
        "text": "Pursued",
        "tense": "past"
      }
    },
    "object": {
      "text": "Princess Leia",
      "sentiment": {
        "type": "positive",
        "score": 0.682563,
```

```
      "mixed": false
    },
    "keywords": [
      {
        "text": "Princess Leia"
      }
    ]
  }
},
{
  "sentence": " Pursued by the Empire's sinister agents, Princess Leia races home ab
  "subject": {
    "text": "the stolen plans",
    "sentiment": {
      "type": "negative",
      "score": -0.44097,
      "mixed": false
    },
    "keywords": [
      {
        "text": "plans"
      }
    ]
  },
  "action": {
    "text": "can save",
    "lemmatized": "can save",
    "verb": {
      "text": "save",
      "tense": "future"
    }
  },
  "object": {
    "text": "her people",
    "keywords": [
      {
        "text": "people"
      }
    ]
  }
},
{
  "sentence": " Pursued by the Empire's sinister agents, Princess Leia races home ab
  "subject": {
    "text": "freedom",
    "keywords": [
      {
        "text": "freedom"
```

```
        }
      ]
    },
    "action": {
      "text": "restore",
      "lemmatized": "restore",
      "verb": {
        "text": "restore",
        "tense": "future"
      }
    },
    "object": {
      "text": "to the galaxy",
      "sentimentFromSubject": {
        "type": "positive",
        "score": 0.698799,
        "mixed": false
      },
      "keywords": [
        {
          "text": "galaxy"
        }
      ]
    }
  }
],
"taxonomy": [
  {
    "label": "/art and entertainment/movies and tv/movies",
    "score": 0.584247,
    "confident": false
  },
  {
    "label": "/society/unrest and war",
    "score": 0.517031,
    "confident": false
  },
  {
    "confident": false,
    "label": "/law, govt and politics/armed forces/army",
    "score": 0.215561
  }
],
"keywords": [
  {
    "relevance": 0.920159,
    "sentiment": {
      "type": "neutral",
```

```
    "mixed": false
  },
  "text": "Opening Crawl"
},
{
  "relevance": 0.785031,
  "sentiment": {
    "score": 0.447888,
    "type": "positive",
    "mixed": false
  },
  "text": "New Hope"
},
{
  "relevance": 0.775799,
  "sentiment": {
    "score": 0.40059,
    "type": "positive",
    "mixed": false
  },
  "text": "Princess Leia races"
},
{
  "relevance": 0.771384,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "Star Wars"
},
{
  "relevance": 0.767527,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "evil Galactic Empire"
},
{
  "relevance": 0.747615,
  "sentiment": {
    "score": -0.494714,
    "type": "negative",
    "mixed": false
  },
  "text": "armored space station"
},
{
```

```
  "relevance": 0.669776,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "Rebel spaceships"
},
{
  "relevance": 0.656828,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "Rebel spies"
},
{
  "relevance": 0.609404,
  "sentiment": {
    "score": 0.484685,
    "type": "positive",
    "mixed": false
  },
  "text": "sinister agents"
},
{
  "relevance": 0.600682,
  "sentiment": {
    "score": -0.409172,
    "type": "negative",
    "mixed": false
  },
  "text": "civil war"
},
{
  "relevance": 0.597836,
  "sentiment": {
    "score": 0.488854,
    "type": "positive",
    "mixed": false
  },
  "text": "hidden base"
},
{
  "relevance": 0.594694,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
```

```
  "text": "ultimate weapon"
},
{
  "relevance": 0.582765,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "secret plans"
},
{
  "relevance": 0.582596,
  "sentiment": {
    "score": -0.494714,
    "type": "negative",
    "mixed": false
  },
  "text": "entire planet"
},
{
  "relevance": 0.574413,
  "sentiment": {
    "score": -0.430026,
    "type": "negative",
    "mixed": false
  },
  "text": "DEATH STAR"
},
{
  "relevance": 0.465394,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "custodian"
},
{
  "relevance": 0.462247,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "Episode"
},
{
  "relevance": 0.443969,
  "sentiment": {
    "score": 0.40059,
```

```
    "type": "positive",
    "mixed": false
  },
  "text": "starship"
},
{
  "relevance": 0.441799,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "victory"
},
{
  "relevance": 0.437824,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "galaxy"
},
{
  "relevance": 0.432092,
  "sentiment": {
    "score": -0.409172,
    "type": "negative",
    "mixed": false
  },
  "text": "period"
},
{
  "relevance": 0.429238,
  "sentiment": {
    "score": -0.37594,
    "type": "negative",
    "mixed": false
  },
  "text": "battle"
},
{
  "relevance": 0.428916,
  "sentiment": {
    "score": 0.40059,
    "type": "positive",
    "mixed": false
  },
  "text": "home"
},
```

```
            {
              "relevance": 0.428498,
              "sentiment": {
                "type": "neutral",
                "mixed": false
              },
              "text": "people"
            }
          ]
        }
      }
    }
  ],
  "notices": []
}
```


1.4.5 Analyze test output

The results returned by the service contain a *snapshot* of the information extracted at each step of processing - document conversions, enrichments and normalizations. We are interested in the output of applying enrichments (`"enrichments_output"`) or after normalizing them (`"normalizations_output"`). These should be identical if no post-processing/normalizations were specified in the configuration.

```
In [6]: # Take a closer look at the results from the "enrichments_output" or "normalizations_out
        output = next((s["snapshot"] for s in res["snapshots"] \
                       if s["step"] == "enrichments_output"), None)
        print(json.dumps(output, indent=2))
```


```
{
  "extracted_metadata": {
    "title": "Star Wars: Episode IV - A New Hope (Opening Crawl)"
  },
  "html": "<?xml version='1.0' encoding='UTF-8' standalone='yes'?><html>\n<head>\n    <meta cont
  "text": "Star Wars: Episode IV - A New Hope (Opening Crawl)\n\nStar Wars: Episode IV - A New H
  "metadata": {},
  "enriched_text": {
    "status": "OK",
    "language": "english",
    "docSentiment": {
      "type": "negative",
      "score": -0.575489,
      "mixed": false
    },
    "concepts": [
      {
        "text": "Star Wars Episode IV: A New Hope",
```

```
  "relevance": 0.98887,
  "website": "http://www.starwars.com/movies/episode-iv",
  "dbpedia": "http://dbpedia.org/resource/Star_Wars_Episode_IV:_A_New_Hope",
  "freebase": "http://rdf.freebase.com/ns/m.0dtfn",
  "opencyc": "http://sw.opencyc.org/concept/Mx4rwVH1ipwpEbGdrcN5Y29ycA"
},
{
  "text": "Star Wars",
  "relevance": 0.985705,
  "website": "http://www.starwars.com",
  "dbpedia": "http://dbpedia.org/resource/Star_Wars",
  "freebase": "http://rdf.freebase.com/ns/m.06mmr",
  "yago": "http://yago-knowledge.org/resource/Star_Wars"
},
{
  "text": "Rebel Alliance",
  "relevance": 0.90639,
  "dbpedia": "http://dbpedia.org/resource/Rebel_Alliance",
  "freebase": "http://rdf.freebase.com/ns/m.0nr1k",
  "yago": "http://yago-knowledge.org/resource/Rebel_Alliance"
},
{
  "text": "Luke Skywalker",
  "relevance": 0.805771,
  "dbpedia": "http://dbpedia.org/resource/Luke_Skywalker",
  "freebase": "http://rdf.freebase.com/ns/m.0f1bg",
  "yago": "http://yago-knowledge.org/resource/Luke_Skywalker"
},
{
  "text": "Star Wars Episode V: The Empire Strikes Back",
  "relevance": 0.801062,
  "website": "http://www.starwars.com/movies/episode-v",
  "dbpedia": "http://dbpedia.org/resource/Star_Wars_Episode_V:_The_Empire_Strikes_Back",
  "freebase": "http://rdf.freebase.com/ns/m.0f3m1"
},
{
  "text": "Darth Vader",
  "relevance": 0.777337,
  "dbpedia": "http://dbpedia.org/resource/Darth_Vader",
  "freebase": "http://rdf.freebase.com/ns/m.0f2y0",
  "yago": "http://yago-knowledge.org/resource/Darth_Vader"
},
{
  "text": "Star Wars Episode VI: Return of the Jedi",
  "relevance": 0.77159,
  "website": "http://www.starwars.com/movies/episode-vi",
  "dbpedia": "http://dbpedia.org/resource/Star_Wars_Episode_VI:_Return_of_the_Jedi",
  "freebase": "http://rdf.freebase.com/ns/m.0ddjy"
```

```
},
{
  "text": "Grand Moff Tarkin",
  "relevance": 0.748553,
  "dbpedia": "http://dbpedia.org/resource/Grand_Moff_Tarkin",
  "freebase": "http://rdf.freebase.com/ns/m.0jrsz",
  "yago": "http://yago-knowledge.org/resource/Grand_Moff_Tarkin"
},
{
  "text": "Star Wars Episode III: Revenge of the Sith",
  "relevance": 0.697065,
  "dbpedia": "http://dbpedia.org/resource/Star_Wars_Episode_III:_Revenge_of_the_Sith",
  "freebase": "http://rdf.freebase.com/ns/m.0fdv3"
},
{
  "text": "Princess Leia Organa",
  "relevance": 0.607163,
  "dbpedia": "http://dbpedia.org/resource/Princess_Leia_Organa",
  "yago": "http://yago-knowledge.org/resource/Princess_Leia_Organa"
},
{
  "text": "Death Star",
  "relevance": 0.59271,
  "dbpedia": "http://dbpedia.org/resource/Death_Star",
  "freebase": "http://rdf.freebase.com/ns/m.0f325",
  "yago": "http://yago-knowledge.org/resource/Death_Star"
},
{
  "text": "Jabba the Hutt",
  "relevance": 0.543114,
  "dbpedia": "http://dbpedia.org/resource/Jabba_the_Hutt",
  "freebase": "http://rdf.freebase.com/ns/m.0fjms",
  "yago": "http://yago-knowledge.org/resource/Jabba_the_Hutt"
},
{
  "text": "Galactic Empire",
  "relevance": 0.523169,
  "dbpedia": "http://dbpedia.org/resource/Galactic_Empire_(Star_Wars)",
  "freebase": "http://rdf.freebase.com/ns/m.0hwyg",
  "yago": "http://yago-knowledge.org/resource/Galactic_Empire_(Star_Wars)"
},
{
  "text": "Star Wars galaxy",
  "relevance": 0.519032,
  "dbpedia": "http://dbpedia.org/resource/Star_Wars_galaxy",
  "yago": "http://yago-knowledge.org/resource/Star_Wars_galaxy"
},
{
```

```
  "text": "Star Wars Expanded Universe",
  "relevance": 0.507885,
  "website": "http://www.starwars.com/eu",
  "dbpedia": "http://dbpedia.org/resource/Star_Wars_Expanded_Universe",
  "freebase": "http://rdf.freebase.com/ns/m.01qbrc",
  "yago": "http://yago-knowledge.org/resource/Star_Wars_Expanded_Universe"
},
{
  "text": "Alderaan",
  "relevance": 0.503196,
  "dbpedia": "http://dbpedia.org/resource/Alderaan",
  "freebase": "http://rdf.freebase.com/ns/m.065lhy",
  "yago": "http://yago-knowledge.org/resource/Alderaan"
},
{
  "text": "Carrie Fisher",
  "relevance": 0.481464,
  "dbpedia": "http://dbpedia.org/resource/Carrie_Fisher",
  "freebase": "http://rdf.freebase.com/ns/m.01tnbn",
  "opencyc": "http://sw.opencyc.org/concept/Mx4rwSJnvZwpEbGdrcN5Y29ycA",
  "yago": "http://yago-knowledge.org/resource/Carrie_Fisher"
},
{
  "text": "Jedi",
  "relevance": 0.474365,
  "dbpedia": "http://dbpedia.org/resource/Jedi",
  "freebase": "http://rdf.freebase.com/ns/m.045n4",
  "opencyc": "http://sw.opencyc.org/concept/Mx4rwNLAD5wpEbGdrcN5Y29ycA",
  "yago": "http://yago-knowledge.org/resource/Jedi"
},
{
  "text": "The Star Wars Holiday Special",
  "relevance": 0.4659,
  "dbpedia": "http://dbpedia.org/resource/The_Star_Wars_Holiday_Special",
  "freebase": "http://rdf.freebase.com/ns/m.0199wf",
  "yago": "http://yago-knowledge.org/resource/The_Star_Wars_Holiday_Special"
},
{
  "text": "Wedge Antilles",
  "relevance": 0.46456,
  "dbpedia": "http://dbpedia.org/resource/Wedge_Antilles",
  "freebase": "http://rdf.freebase.com/ns/m.0ddp_",
  "yago": "http://yago-knowledge.org/resource/Wedge_Antilles"
},
{
  "text": "English-language films",
  "relevance": 0.460469,
  "dbpedia": "http://dbpedia.org/resource/English-language_films"
```

```
},
{
  "text": "Padm\u00e9 Amidala",
  "relevance": 0.453766,
  "dbpedia": "http://dbpedia.org/resource/Padm\u00e9_Amidala",
  "freebase": "http://rdf.freebase.com/ns/m.0drf_",
  "yago": "http://yago-knowledge.org/resource/Padm%C3%A9_Amidala"
},
{
  "text": "Galactic Civil War",
  "relevance": 0.447243,
  "dbpedia": "http://dbpedia.org/resource/Galactic_Civil_War",
  "freebase": "http://rdf.freebase.com/ns/m.01qc6_",
  "yago": "http://yago-knowledge.org/resource/Galactic_Civil_War"
},
{
  "text": "Spaceballs",
  "relevance": 0.436864,
  "dbpedia": "http://dbpedia.org/resource/Spaceballs",
  "freebase": "http://rdf.freebase.com/ns/m.0j_nw",
  "yago": "http://yago-knowledge.org/resource/Spaceballs"
},
{
  "text": "Clone Wars",
  "relevance": 0.43403,
  "dbpedia": "http://dbpedia.org/resource/Clone_Wars_(Star_Wars)",
  "freebase": "http://rdf.freebase.com/ns/m.01qc6n",
  "yago": "http://yago-knowledge.org/resource/Clone_Wars_(Star_Wars)"
},
{
  "text": "C-3PO",
  "relevance": 0.432345,
  "dbpedia": "http://dbpedia.org/resource/C-3PO",
  "freebase": "http://rdf.freebase.com/ns/m.0g_sv",
  "yago": "http://yago-knowledge.org/resource/C-3PO"
},
{
  "text": "Obi-Wan Kenobi",
  "relevance": 0.431988,
  "dbpedia": "http://dbpedia.org/resource/Obi-Wan_Kenobi",
  "freebase": "http://rdf.freebase.com/ns/m.0fkm7",
  "yago": "http://yago-knowledge.org/resource/Obi-Wan_Kenobi"
},
{
  "text": "R2-D2",
  "relevance": 0.430085,
  "dbpedia": "http://dbpedia.org/resource/R2-D2",
  "freebase": "http://rdf.freebase.com/ns/m.0g_rg",
```

```
    "yago": "http://yago-knowledge.org/resource/R2-D2"
  }
],
"entities": [
  {
    "type": "FieldTerminology",
    "relevance": 0.972336,
    "sentiment": {
      "type": "negative",
      "score": -0.409172,
      "mixed": false
    },
    "count": 1,
    "text": "civil war"
  }
],
"relations": [
  {
    "sentence": " It is a period of civil war.",
    "subject": {
      "text": "It"
    },
    "action": {
      "text": "is",
      "lemmatized": "be",
      "verb": {
        "text": "be",
        "tense": "present"
      }
    },
    "object": {
      "text": "a period of civil war",
      "sentiment": {
        "type": "negative",
        "score": -0.564509,
        "mixed": false
      },
      "entities": [
        {
          "type": "FieldTerminology",
          "text": "civil war"
        }
      ],
      "keywords": [
        {
          "text": "civil war"
        },
        {
```

```
        "text": "period"
      }
    ]
  }
},
{
  "sentence": " Rebel spaceships, striking from a hidden base, have won their first victor
  "subject": {
    "text": "Rebel spaceships, striking from a hidden base,",
    "keywords": [
      {
        "text": "Rebel spaceships"
      },
      {
        "text": "hidden base"
      }
    ]
  },
  "action": {
    "text": "have won",
    "lemmatized": "have win",
    "verb": {
      "text": "win",
      "tense": "past"
    }
  },
  "object": {
    "text": "their first victory",
    "keywords": [
      {
        "text": "victory"
      }
    ]
  }
},
{
  "sentence": " During the battle, Rebel spies managed to steal secret plans to the Empire
  "subject": {
    "text": "Rebel spies",
    "keywords": [
      {
        "text": "Rebel spies"
      }
    ]
  },
  "action": {
    "text": "managed",
    "lemmatized": "manage",
```

```
    "verb": {
      "text": "manage",
      "tense": "past"
    }
  },
  "object": {
    "text": "to steal secret plans to the Empire's ultimate weapon, the DEATH STAR, an arm
    "sentiment": {
      "type": "negative",
      "score": -0.596586,
      "mixed": false
    },
    "keywords": [
      {
        "text": "armored space station"
      },
      {
        "text": "ultimate weapon"
      },
      {
        "text": "secret plans"
      },
      {
        "text": "entire planet"
      }
    ]
  }
},
{
  "sentence": " During the battle, Rebel spies managed to steal secret plans to the Empire
  "subject": {
    "text": "Rebel spies",
    "keywords": [
      {
        "text": "Rebel spies"
      }
    ]
  },
  "action": {
    "text": "managed to steal",
    "lemmatized": "manage to steal",
    "verb": {
      "text": "steal",
      "tense": "future"
    }
  },
  "object": {
    "text": "secret plans",
```

```
    "keywords": [
      {
        "text": "secret plans"
      }
    ]
  }
},
{
  "sentence": " During the battle, Rebel spies managed to steal secret plans to the Empire
  "subject": {
    "text": "Rebel spies",
    "keywords": [
      {
        "text": "Rebel spies"
      }
    ]
  },
  "action": {
    "text": "to destroy",
    "lemmatized": "to destroy",
    "verb": {
      "text": "destroy",
      "tense": "future"
    }
  },
  "object": {
    "text": "an entire planet",
    "sentiment": {
      "type": "positive",
      "score": 0.748639,
      "mixed": false
    },
    "sentimentFromSubject": {
      "type": "negative",
      "score": -0.754579,
      "mixed": false
    },
    "keywords": [
      {
        "text": "entire planet"
      }
    ]
  }
},
{
  "sentence": " Pursued by the Empire's sinister agents, Princess Leia races home aboard h
  "subject": {
    "text": "by the Empire",
```

```
      "sentiment": {
        "type": "positive",
        "score": 0.554106,
        "mixed": false
      },
      "keywords": [
        {
          "text": "Empire"
        }
      ]
    },
    "action": {
      "text": "Pursued",
      "lemmatized": "Pursued",
      "verb": {
        "text": "Pursued",
        "tense": "past"
      }
    },
    "object": {
      "text": "Princess Leia",
      "sentiment": {
        "type": "positive",
        "score": 0.682563,
        "mixed": false
      },
      "keywords": [
        {
          "text": "Princess Leia"
        }
      ]
    }
  },
  {
    "sentence": " Pursued by the Empire's sinister agents, Princess Leia races home aboard h
    "subject": {
      "text": "the stolen plans",
      "sentiment": {
        "type": "negative",
        "score": -0.44097,
        "mixed": false
      },
      "keywords": [
        {
          "text": "plans"
        }
      ]
    },
```

```
  "action": {
    "text": "can save",
    "lemmatized": "can save",
    "verb": {
      "text": "save",
      "tense": "future"
    }
  },
  "object": {
    "text": "her people",
    "keywords": [
      {
        "text": "people"
      }
    ]
  }
},
{
  "sentence": " Pursued by the Empire's sinister agents, Princess Leia races home aboard h
  "subject": {
    "text": "freedom",
    "keywords": [
      {
        "text": "freedom"
      }
    ]
  },
  "action": {
    "text": "restore",
    "lemmatized": "restore",
    "verb": {
      "text": "restore",
      "tense": "future"
    }
  },
  "object": {
    "text": "to the galaxy",
    "sentimentFromSubject": {
      "type": "positive",
      "score": 0.698799,
      "mixed": false
    },
    "keywords": [
      {
        "text": "galaxy"
      }
    ]
  }
```

```
  }
],
"taxonomy": [
  {
    "label": "/art and entertainment/movies and tv/movies",
    "score": 0.584247,
    "confident": false
  },
  {
    "label": "/society/unrest and war",
    "score": 0.517031,
    "confident": false
  },
  {
    "confident": false,
    "label": "/law, govt and politics/armed forces/army",
    "score": 0.215561
  }
],
"keywords": [
  {
    "relevance": 0.920159,
    "sentiment": {
      "type": "neutral",
      "mixed": false
    },
    "text": "Opening Crawl"
  },
  {
    "relevance": 0.785031,
    "sentiment": {
      "score": 0.447888,
      "type": "positive",
      "mixed": false
    },
    "text": "New Hope"
  },
  {
    "relevance": 0.775799,
    "sentiment": {
      "score": 0.40059,
      "type": "positive",
      "mixed": false
    },
    "text": "Princess Leia races"
  },
  {
    "relevance": 0.771384,
```

```
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "Star Wars"
},
{
  "relevance": 0.767527,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "evil Galactic Empire"
},
{
  "relevance": 0.747615,
  "sentiment": {
    "score": -0.494714,
    "type": "negative",
    "mixed": false
  },
  "text": "armored space station"
},
{
  "relevance": 0.669776,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "Rebel spaceships"
},
{
  "relevance": 0.656828,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "Rebel spies"
},
{
  "relevance": 0.609404,
  "sentiment": {
    "score": 0.484685,
    "type": "positive",
    "mixed": false
  },
  "text": "sinister agents"
},
```

```
{
  "relevance": 0.600682,
  "sentiment": {
    "score": -0.409172,
    "type": "negative",
    "mixed": false
  },
  "text": "civil war"
},
{
  "relevance": 0.597836,
  "sentiment": {
    "score": 0.488854,
    "type": "positive",
    "mixed": false
  },
  "text": "hidden base"
},
{
  "relevance": 0.594694,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "ultimate weapon"
},
{
  "relevance": 0.582765,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "secret plans"
},
{
  "relevance": 0.582596,
  "sentiment": {
    "score": -0.494714,
    "type": "negative",
    "mixed": false
  },
  "text": "entire planet"
},
{
  "relevance": 0.574413,
  "sentiment": {
    "score": -0.430026,
    "type": "negative",
```

```
    "mixed": false
  },
  "text": "DEATH STAR"
},
{
  "relevance": 0.465394,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "custodian"
},
{
  "relevance": 0.462247,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "Episode"
},
{
  "relevance": 0.443969,
  "sentiment": {
    "score": 0.40059,
    "type": "positive",
    "mixed": false
  },
  "text": "starship"
},
{
  "relevance": 0.441799,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "victory"
},
{
  "relevance": 0.437824,
  "sentiment": {
    "type": "neutral",
    "mixed": false
  },
  "text": "galaxy"
},
{
  "relevance": 0.432092,
  "sentiment": {
```

```
          "score": -0.409172,
          "type": "negative",
          "mixed": false
        },
        "text": "period"
      },
      {
        "relevance": 0.429238,
        "sentiment": {
          "score": -0.37594,
          "type": "negative",
          "mixed": false
        },
        "text": "battle"
      },
      {
        "relevance": 0.428916,
        "sentiment": {
          "score": 0.40059,
          "type": "positive",
          "mixed": false
        },
        "text": "home"
      },
      {
        "relevance": 0.428498,
        "sentiment": {
          "type": "neutral",
          "mixed": false
        },
        "text": "people"
      }
    ]
  }
}
```


```
In [7]: output.keys()
```

```
Out[7]: dict_keys(['extracted_metadata', 'html', 'text', 'metadata', 'enriched_text'])
```

```
In [8]: output['enriched_text'].keys()
```

```
Out[8]: dict_keys(['status', 'language', 'docSentiment', 'concepts', 'entities', 'relations', 't
```

```
In [9]: cfg['enrichments'][0]['options']['extract']
```

```
Out[9]: 'keyword, entity, doc-sentiment, taxonomy, concept, relation'
```

Answer the following questions based on the output above. Note that it contains the input HTML, extracted text and metadata as well as the actual enrichment results (`"enriched_text"` block).

```
In [10]: enriched = output['enriched_text']
         enriched.keys()

Out[10]: dict_keys(['status', 'language', 'docSentiment', 'concepts', 'entities', 'relations', '
```

Sentiment **Q**: What is the overall sentiment detected in this text? Mention the `type` (positive/negative) and `score`. (*Hint: Look for the `"docSentiment"` key in the output.*)

```
In [11]: enriched['docSentiment']

Out[11]: {'mixed': False, 'score': -0.575489, 'type': 'negative'}
```

A: The sentiment detected is negative and the score is -0.575489.

Concepts **Q**: List 3 concepts that have been identified with a relevance > 0.5. Note that not all concepts here may be present directly in the text, some may have been inferred by Watson. (*Hint: Look for `"concepts"`.*)

```
In [12]: [concept['text'] for concept in enriched['concepts'] \
                 if concept['relevance'] > 0.5][:3]

Out[12]: ['Star Wars Episode IV: A New Hope', 'Star Wars', 'Rebel Alliance']
```

A: The three concepts with the highest relevance scores are 'Star Wars Episode IV: A New Hope', 'Star Wars', and 'Rebel Alliance'.

Relations Each relation is essentially a deeper analysis of a sentence (or part of a sentence). Here is a sample relation:


```
{
  "sentence": " During the battle, Rebel spies managed to steal secret plans to the Empire's ult
  "subject": {
    "text": "Rebel spies",
    "keywords": [
      {
        "text": "Rebel spies"
      }
    ]
  },
  "action": {
    "text": "managed to steal",
    "lemmatized": "manage to steal",
    "verb": {
      "text": "steal",
      "tense": "future"
```

```
    }
  },
  "object": {
    "text": "secret plans",
    "keywords": [
      {
        "text": "secret plans"
      }
    ]
  }
}
```

In this case, Watson seems to have done a pretty good job of extracting some meaning from the sentence.

Q: Find a relation where the extracted meaning is not as accurate, or not what you would've expected. List the `sentence`, `subject`, `action` and `object` parts as identified, and what you would've marked instead. (*Hint: Look for `"relations"`.*)

```
In [13]: from pprint import pprint

         rel = [relation for relation in enriched['relations'] \
                if "restore freedom to the galaxy" in relation['sentence'] \
                and relation['action']['lemmatized'] == 'restore'][0]

         pprint(rel)
```

```
{'action': {'lemmatized': 'restore',
            'text': 'restore',
            'verb': {'tense': 'future', 'text': 'restore'}},
 'object': {'keywords': [{'text': 'galaxy'}],
            'sentimentFromSubject': {'mixed': False,
                                     'score': 0.698799,
                                     'type': 'positive'},
            'text': 'to the galaxy'},
 'sentence': " Pursued by the Empire's sinister agents, Princess Leia races "
             'home aboard her starship, custodian of the stolen plans that can '
             'save her people and restore freedom to the galaxy...',
 'subject': {'keywords': [{'text': 'freedom'}], 'text': 'freedom'}}
```

A: Watson has analysed the last part of the sentence *" Pursued by the Empire's sinister agents, Princess Leia races home aboard her starship, custodian of the stolen plans that can save her people and restore freedom to the galaxy..."* (namely *stolen plans... restore freedom to the galaxy*) as having as subject *freedom* and as object *to the galaxy*. The verb is correctly identified as *restore*. In fact, the subject should be *stolen plans* and the object *freedom*.

Keywords You may have noticed that Watson identifies some `"keywords"` in the relations, e.g. `"Rebel spies"` and `"secret plans"` in the Star Wars example above. The output also contains

a list of all keywords at the top level, for your convenience, along with their relevance to the document and sentiment conveyed. Let's visualize these keywords as a word cloud!

Note: We'll be using this handy worldcoud library to generate the visualization. So you will need to install it first:

```
pip install wordcloud
```

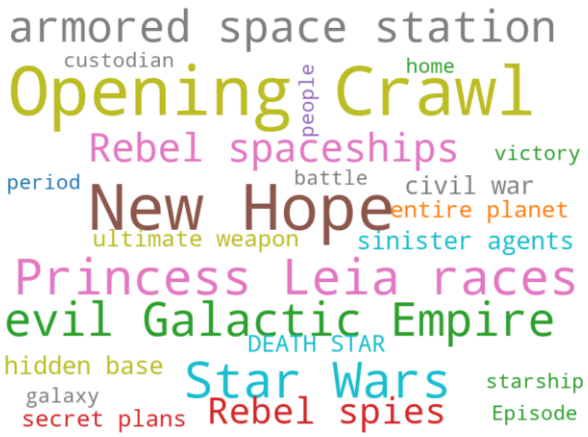
```
In [14]: # enriched['keywords']
```

```
In [15]: # Visualize keywords by relevance as a wordcloud
         from wordcloud import WordCloud

         wc_data = {w['text']: w['relevance'] for w in output['enriched_text']['keywords']}
         wc = WordCloud(width=400, height=300, scale=2, background_color='white', colormap='Vega
         wc.generate_from_frequencies(wc_data)  # use precomputed relevance instead of frequenci

         plt.figure(figsize=(4, 3), dpi=200)
         plt.imshow(wc, interpolation='bilinear')
         plt.axis('off')
```

```
Out[15]: (-0.5, 799.5, 599.5, -0.5)
```



Feel free to play with this visualization and improve it. What about using a different metric instead of relevance, e.g. direct word frequencies that the wordcloud library computes by default?

```
In [16]: # Plotting a word cloud from word frequencies.
         wc_freq = wc.generate(output['text'])
         plt.imshow(wc_freq, interpolation='bilinear')
         plt.axis("off")
```

```
Out[16]: (-0.5, 799.5, 599.5, -0.5)
```

Other results Watson's output also includes processed results from other enrichments that were applied to the text, including entities and taxonomy (what topic or category does this text relate to).

```
"taxonomy": [
  {
    "label": "/art and entertainment/movies and tv/movies",
    "score": 0.584247,
    "confident": false
  },
  {
    "label": "/society/unrest and war",
    "score": 0.517031,
    "confident": false
  },
  {
    "confident": false,
    "label": "/law, govt and politics/armed forces/army",
    "score": 0.215561
  }
],
```

Get a good sense of all the different pieces of information available in the results. Start thinking about which ones will be useful for looking up answers to questions, and how you might use them.

```
In [17]: enriched.keys()
```

```
Out[17]: dict_keys(['status', 'language', 'docSentiment', 'concepts', 'entities', 'relations', '
```

1.5 2. Ingest documents

1.5.1 Create a collection

A *collection* is used to organize documents of the same kind. For instance, you may want to create a collection of book reviews, or a collection of Wikipedia articles, but it may not make much sense to mix the two groups. This allows Watson to make meaningful inferences over the set of documents, find commonalities and identify important concepts.

Let's create one called "Story Chunks".

```
In [18]: # Prepare a collection of documents to use
         col, col_id = helper.fetch_object(discovery, "collection", "Story Chunks",
                                           environment_id=env_id,
                                           create=True, create_args=dict(
                 environment_id=env_id, configuration_id=cfg_id,
                 description="Stories and plots split up into chunks suitable for answering"
             ))
         print(json.dumps(col, indent=2))
```

```
Found collection: Story Chunks (628a54ef-f0ab-42c8-aec6-d4293a0a531a)
{
  "collection_id": "628a54ef-f0ab-42c8-aec6-d4293a0a531a",
  "name": "Story Chunks",
  "configuration_id": "102e52ba-ec95-427e-bb88-b3f978034f76",
  "language": "en",
  "status": "active",
  "description": "Stories and plots split up into chunks suitable for answering",
  "created": "2017-08-18T10:16:32.648Z",
  "updated": "2017-08-18T10:16:32.648Z",
  "document_counts": {
    "available": 126,
    "processing": 0,
    "failed": 0
  },
  "disk_usage": {
    "used_bytes": 11668587
  },
  "training_status": {
    "data_updated": "",
    "total_examples": 0,
    "sufficient_label_diversity": false,
    "processing": false,
    "minimum_examples_added": false,
    "successfully_trained": "",
    "available": false,
    "notices": 0,
    "minimum_queries_added": false
```

```
  }
}
```

Once you have created a collection, you should be able to view it using the Discovery Service tool. To open, go to the **Manage** tab for your service instance, and click the **Launch tool** button.

Here you should see the "Story Chunks" collection you just created.

You can open the collection to view more details about it. If you need to modify configuration options, click the **Switch** link and create a new configuration (the default one cannot be changed).

1.5.2 Add documents

Okay, now that we have everything set up, let's add a set of "documents" we want Watson to look up answers from, using the Python SDK. Note that Watson treats each "document" as a unit of text that is returned as the result of a query. But we want to retrieve a paragraph of text for each question. So, let's split each file up into individual paragraphs. We will use the BeautifulSoup library for this purpose.

***Note**: You could also add and manage documents in the collection using the Discovery tool, but you would have to split paragraphs up into separate files.*

***Note**: We have provided a set of files (`data/Star-Wars/*.html`) with summary plots for Star Wars movies, but you are free to use a collection of your choice. Open one of the files in a text editor to see how the paragraphs are delimited using `<p>...</p>` tags - this is how the code block below split paragraphs into separate "documents".*

```
In [19]: # glob.glob?

In [20]: glob.glob(os.path.join(data_dir, "Star-Wars", "*.html"))

Out[20]: ['data/Star-Wars/Episode-VI_Return-of-the-Jedi.html',
          'data/Star-Wars/Episode-VII_The-Force-Awakens.html',
          'data/Star-Wars/Episode-V_The-Empire-Strikes-Back.html',
          'data/Star-Wars/Rogue-One.html',
          'data/Star-Wars/Episode-III_Revenge-of-the-Sith.html',
          'data/Star-Wars/Episode-I_The-Phantom-Menace.html',
          'data/Star-Wars/Episode-IV_A-New-Hope.html',
          'data/Star-Wars/Episode-II_Attack-of-the-Clones.html']

In [21]: # Add documents to collection
         doc_ids = []  # to store the generated id for each document added
         for filename in glob.glob(os.path.join(data_dir, "Star-Wars", "*.html")):
             print("Adding file:", filename)
             with open(filename, "r") as f:
                 # Split each individual <p> into its own "document"
                 doc = f.read()
                 soup = BeautifulSoup(doc, 'html.parser')
                 for i, p in enumerate(soup.find_all('p')):
                     doc_info = discovery.add_document(environment_id=env_id,
                                                       collection_id=col_id,
                                                       file_data=json.dumps({"text": p.get_text(strip=True)}),
```

```
                                        mime_type="application/json",
                                        metadata={"title": soup.title.get_text(strip=True)})
                    doc_ids.append(doc_info["document_id"])
        print("Total", len(doc_ids), "documents added.")
```

```
Adding file: data/Star-Wars/Episode-VI_Return-of-the-Jedi.html
Adding file: data/Star-Wars/Episode-VII_The-Force-Awakens.html
Adding file: data/Star-Wars/Episode-V_The-Empire-Strikes-Back.html
Adding file: data/Star-Wars/Rogue-One.html
Adding file: data/Star-Wars/Episode-III_Revenge-of-the-Sith.html
Adding file: data/Star-Wars/Episode-I_The-Phantom-Menace.html
Adding file: data/Star-Wars/Episode-IV_A-New-Hope.html
Adding file: data/Star-Wars/Episode-II_Attack-of-the-Clones.html
Total 42 documents added.
```

If you look at the collection details, you may notice that the `"document_counts"` field now shows some documents as `available` or `processing`. Once processing is complete, you should see all the documents under the `available` count.

```
In [22]: # View collection details to verify all documents have been processed
         col, col_id = helper.fetch_object(discovery, "collection", "Story Chunks",
                                           environment_id=env_id)
         print(json.dumps(col, indent=2))
```

```
Found collection: Story Chunks (628a54ef-f0ab-42c8-aec6-d4293a0a531a)
{
  "collection_id": "628a54ef-f0ab-42c8-aec6-d4293a0a531a",
  "name": "Story Chunks",
  "configuration_id": "102e52ba-ec95-427e-bb88-b3f978034f76",
  "language": "en",
  "status": "active",
  "description": "Stories and plots split up into chunks suitable for answering",
  "created": "2017-08-18T10:16:32.648Z",
  "updated": "2017-08-18T10:16:32.648Z",
  "document_counts": {
    "available": 161,
    "processing": 7,
    "failed": 0
  },
  "disk_usage": {
    "used_bytes": 12705995
  },
  "training_status": {
    "data_updated": "",
    "total_examples": 0,
    "sufficient_label_diversity": false,
    "processing": false,
    "minimum_examples_added": false,
```

```
    "successfully_trained": "",
    "available": false,
    "notices": 0,
    "minimum_queries_added": false
  }
}
```

So, what did the Discovery service learn? If you list the fields extracted from the set of documents in the collection as part of the enrichment process, you'll see familiar fields like `concepts`, `entities` and `keywords` that were returned in the test analysis.

```
In [23]: # List all fields extracted
         discovery.list_collection_fields(environment_id=env_id, collection_id=col_id)

Out[23]: {'fields': [{'field': '628a54ef-f0ab-42c8-aec6-d4293a0a531a.mappings.document._all.anal
             'type': 'english'},
            {'field': 'enriched_text', 'type': 'nested'},
            {'field': 'extracted_metadata', 'type': 'nested'},
            {'field': 'metadata', 'type': 'nested'},
            {'field': 'text', 'type': 'string'},
            {'field': 'text.analyzer', 'type': 'english'},
            {'field': 'enriched_text.properties.concepts', 'type': 'nested'},
            {'field': 'enriched_text.properties.docSentiment', 'type': 'nested'},
            {'field': 'enriched_text.properties.entities', 'type': 'nested'},
            {'field': 'enriched_text.properties.keywords', 'type': 'nested'},
            {'field': 'enriched_text.properties.language', 'type': 'string'},
            {'field': 'enriched_text.properties.language.analyzer', 'type': 'english'},
            {'field': 'enriched_text.properties.relations', 'type': 'nested'},
            {'field': 'enriched_text.properties.status', 'type': 'string'},
            {'field': 'enriched_text.properties.status.analyzer', 'type': 'english'},
            {'field': 'enriched_text.properties.taxonomy', 'type': 'nested'},
            {'field': 'extracted_metadata.properties.file_type', 'type': 'string'},
            {'field': 'extracted_metadata.properties.file_type.analyzer',
             'type': 'english'},
            {'field': 'extracted_metadata.properties.filename', 'type': 'string'},
            {'field': 'extracted_metadata.properties.filename.analyzer',
             'type': 'english'},
            {'field': 'extracted_metadata.properties.sha1', 'type': 'string'},
            {'field': 'extracted_metadata.properties.sha1.analyzer', 'type': 'english'},
            {'field': 'metadata.properties.title', 'type': 'string'},
            {'field': 'metadata.properties.title.analyzer', 'type': 'english'},
            {'field': 'enriched_text.properties.concepts.properties.ciaFactbook',
             'type': 'string'},
            {'field': 'enriched_text.properties.concepts.properties.ciaFactbook.analyzer',
             'type': 'english'},
            {'field': 'enriched_text.properties.concepts.properties.dbpedia',
             'type': 'string'},
```

```
{'field': 'enriched_text.properties.concepts.properties.dbpedia.analyzer',
 'type': 'english'},
{'field': 'enriched_text.properties.concepts.properties.freebase',
 'type': 'string'},
{'field': 'enriched_text.properties.concepts.properties.freebase.analyzer',
 'type': 'english'},
{'field': 'enriched_text.properties.concepts.properties.geo',
 'type': 'string'},
{'field': 'enriched_text.properties.concepts.properties.geo.analyzer',
 'type': 'english'},
{'field': 'enriched_text.properties.concepts.properties.musicBrainz',
 'type': 'string'},
{'field': 'enriched_text.properties.concepts.properties.musicBrainz.analyzer',
 'type': 'english'},
{'field': 'enriched_text.properties.concepts.properties.opencyc',
 'type': 'string'},
{'field': 'enriched_text.properties.concepts.properties.opencyc.analyzer',
 'type': 'english'},
{'field': 'enriched_text.properties.concepts.properties.relevance',
 'type': 'double'},
{'field': 'enriched_text.properties.concepts.properties.text',
 'type': 'string'},
{'field': 'enriched_text.properties.concepts.properties.text.analyzer',
 'type': 'english'},
{'field': 'enriched_text.properties.concepts.properties.website',
 'type': 'string'},
{'field': 'enriched_text.properties.concepts.properties.website.analyzer',
 'type': 'english'},
{'field': 'enriched_text.properties.concepts.properties.yago',
 'type': 'string'},
{'field': 'enriched_text.properties.concepts.properties.yago.analyzer',
 'type': 'english'},
{'field': 'enriched_text.properties.docSentiment.properties.mixed',
 'type': 'boolean'},
{'field': 'enriched_text.properties.docSentiment.properties.score',
 'type': 'double'},
{'field': 'enriched_text.properties.docSentiment.properties.type',
 'type': 'string'},
{'field': 'enriched_text.properties.docSentiment.properties.type.analyzer',
 'type': 'english'},
{'field': 'enriched_text.properties.entities.properties.count',
 'type': 'double'},
{'field': 'enriched_text.properties.entities.properties.disambiguated',
 'type': 'nested'},
{'field': 'enriched_text.properties.entities.properties.relevance',
 'type': 'double'},
{'field': 'enriched_text.properties.entities.properties.sentiment',
 'type': 'nested'},
```

```
{'field': 'enriched_text.properties.entities.properties.text',
 'type': 'string'},
{'field': 'enriched_text.properties.entities.properties.text.analyzer',
 'type': 'english'},
{'field': 'enriched_text.properties.entities.properties.type',
 'type': 'string'},
{'field': 'enriched_text.properties.entities.properties.type.analyzer',
 'type': 'english'},
{'field': 'enriched_text.properties.keywords.properties.relevance',
 'type': 'double'},
{'field': 'enriched_text.properties.keywords.properties.sentiment',
 'type': 'nested'},
{'field': 'enriched_text.properties.keywords.properties.text',
 'type': 'string'},
{'field': 'enriched_text.properties.keywords.properties.text.analyzer',
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.action',
 'type': 'nested'},
{'field': 'enriched_text.properties.relations.properties.location',
 'type': 'nested'},
{'field': 'enriched_text.properties.relations.properties.object',
 'type': 'nested'},
{'field': 'enriched_text.properties.relations.properties.sentence',
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.sentence.analyzer',
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.subject',
 'type': 'nested'},
{'field': 'enriched_text.properties.taxonomy.properties.confident',
 'type': 'boolean'},
{'field': 'enriched_text.properties.taxonomy.properties.label',
 'type': 'string'},
{'field': 'enriched_text.properties.taxonomy.properties.label.analyzer',
 'type': 'english'},
{'field': 'enriched_text.properties.taxonomy.properties.score',
 'type': 'double'},
{'field': 'enriched_text.properties.entities.properties.disambiguated.properties.dbpe
 'type': 'string'},
{'field': 'enriched_text.properties.entities.properties.disambiguated.properties.dbpe
 'type': 'english'},
{'field': 'enriched_text.properties.entities.properties.disambiguated.properties.free
 'type': 'string'},
{'field': 'enriched_text.properties.entities.properties.disambiguated.properties.free
 'type': 'english'},
{'field': 'enriched_text.properties.entities.properties.disambiguated.properties.name
 'type': 'string'},
{'field': 'enriched_text.properties.entities.properties.disambiguated.properties.name
 'type': 'english'},
```

```
{'field': 'enriched_text.properties.entities.properties.disambiguated.properties.yago
 'type': 'string'},
{'field': 'enriched_text.properties.entities.properties.disambiguated.properties.yago
 'type': 'english'},
{'field': 'enriched_text.properties.entities.properties.sentiment.properties.mixed',
 'type': 'boolean'},
{'field': 'enriched_text.properties.entities.properties.sentiment.properties.score',
 'type': 'double'},
{'field': 'enriched_text.properties.entities.properties.sentiment.properties.type',
 'type': 'string'},
{'field': 'enriched_text.properties.entities.properties.sentiment.properties.type.ana
 'type': 'english'},
{'field': 'enriched_text.properties.keywords.properties.sentiment.properties.mixed',
 'type': 'boolean'},
{'field': 'enriched_text.properties.keywords.properties.sentiment.properties.score',
 'type': 'double'},
{'field': 'enriched_text.properties.keywords.properties.sentiment.properties.type',
 'type': 'string'},
{'field': 'enriched_text.properties.keywords.properties.sentiment.properties.type.ana
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.action.properties.lemmatized
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.action.properties.lemmatized
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.action.properties.text',
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.action.properties.text.analy
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.action.properties.verb',
 'type': 'nested'},
{'field': 'enriched_text.properties.relations.properties.location.properties.entities
 'type': 'nested'},
{'field': 'enriched_text.properties.relations.properties.location.properties.sentimen
 'type': 'nested'},
{'field': 'enriched_text.properties.relations.properties.location.properties.text',
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.location.properties.text.ana
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.object.properties.entities',
 'type': 'nested'},
{'field': 'enriched_text.properties.relations.properties.object.properties.keywords',
 'type': 'nested'},
{'field': 'enriched_text.properties.relations.properties.object.properties.sentiment'
 'type': 'nested'},
{'field': 'enriched_text.properties.relations.properties.object.properties.sentimentF
 'type': 'nested'},
{'field': 'enriched_text.properties.relations.properties.object.properties.text',
 'type': 'string'},
```

```
{'field': 'enriched_text.properties.relations.properties.object.properties.text.analy
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.entities'
 'type': 'nested'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.keywords'
 'type': 'nested'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.sentiment
 'type': 'nested'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.text',
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.text.anal
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.action.properties.verb.prope
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.action.properties.verb.prope
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.action.properties.verb.prope
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.action.properties.verb.prope
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.location.properties.entities
 'type': 'nested'},
{'field': 'enriched_text.properties.relations.properties.location.properties.entities
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.location.properties.entities
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.location.properties.entities
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.location.properties.entities
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.location.properties.sentimen
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.location.properties.sentimen
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.location.properties.sentimen
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.location.properties.sentimen
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.object.properties.entities.p
 'type': 'nested'},
{'field': 'enriched_text.properties.relations.properties.object.properties.entities.p
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.object.properties.entities.p
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.object.properties.entities.p
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.object.properties.entities.p
 'type': 'english'},
```

```
{'field': 'enriched_text.properties.relations.properties.object.properties.keywords.p
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.object.properties.keywords.p
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.object.properties.sentiment.
 'type': 'boolean'},
{'field': 'enriched_text.properties.relations.properties.object.properties.sentiment.
 'type': 'double'},
{'field': 'enriched_text.properties.relations.properties.object.properties.sentiment.
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.object.properties.sentiment.
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.object.properties.sentimentF
 'type': 'boolean'},
{'field': 'enriched_text.properties.relations.properties.object.properties.sentimentF
 'type': 'double'},
{'field': 'enriched_text.properties.relations.properties.object.properties.sentimentF
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.object.properties.sentimentF
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.entities.
 'type': 'nested'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.entities.
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.entities.
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.entities.
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.entities.
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.keywords.
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.keywords.
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.sentiment
 'type': 'boolean'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.sentiment
 'type': 'double'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.sentiment
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.sentiment
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.location.properties.entities
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.location.properties.entities
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.location.properties.entities
 'type': 'string'},
```

```
{'field': 'enriched_text.properties.relations.properties.location.properties.entities
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.location.properties.entities
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.location.properties.entities
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.location.properties.entities
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.location.properties.entities
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.object.properties.entities.p
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.object.properties.entities.p
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.object.properties.entities.p
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.object.properties.entities.p
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.object.properties.entities.p
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.object.properties.entities.p
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.object.properties.entities.p
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.object.properties.entities.p
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.entities.
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.entities.
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.entities.
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.entities.
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.entities.
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.entities.
 'type': 'english'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.entities.
 'type': 'string'},
{'field': 'enriched_text.properties.relations.properties.subject.properties.entities.
 'type': 'english'}]}
```

1.5.3 Test query

Let's perform a simple query to see if the service can fetch the proper document for us: > *Look for all paragraphs that have a relation (sentence) with "Jar Jar" as the subject, and return the title and text.*

```
In [24]: # A simple query
         results = discovery.query(environment_id=env_id,
                                   collection_id=col_id,
                                   query_options={
                                       "query": "enriched_text.relations.subject.text:\'Jar Jar\
                                       "return": "metadata.title,text"
                                   })
         print(json.dumps(results, indent=2))
```


```
{
  "matching_results": 8,
  "results": [
    {
      "id": "9be97794-0268-41c9-8edd-04d9580d2731",
      "score": 3.006289,
      "metadata": {
        "title": "Star Wars: Episode I - The Phantom Menace"
      },
      "text": "Supreme Chancellor Valorum, leader of the Galactic Republic, dispatches Jedi Knig
    },
    {
      "id": "ceb06648-f4dc-400b-b729-28024ad27a6e",
      "score": 3.006289,
      "metadata": {
        "title": "Star Wars: Episode I - The Phantom Menace"
      },
      "text": "Supreme Chancellor Valorum, leader of the Galactic Republic, dispatches Jedi Knig
    },
    {
      "id": "b7d73353-63de-41f2-a353-9e843cf1729e",
      "score": 3.006289,
      "metadata": {
        "title": "Star Wars: Episode I - The Phantom Menace"
      },
      "text": "Supreme Chancellor Valorum, leader of the Galactic Republic, dispatches Jedi Knig
    },
    {
      "id": "c98f13bd-a406-40af-b519-c9d08bfdeb36",
      "score": 3.006289,
      "metadata": {
        "title": "Star Wars: Episode I - The Phantom Menace"
      },
      "text": "Supreme Chancellor Valorum, leader of the Galactic Republic, dispatches Jedi Knig
    },
    {
      "id": "cce1d118-1fc1-414c-8842-cc30c1d32cd2",
      "score": 2.1473494,
      "metadata": {
```

```
        "title": "Star Wars: Episode I - The Phantom Menace"
      },
      "text": "Amidala's ship is unable to sustain its hyperdrive and lands for repairs on the d
    },
    {
      "id": "20d7fb14-7569-46f3-bc43-5b2378807bfa",
      "score": 2.1473494,
      "metadata": {
        "title": "Star Wars: Episode I - The Phantom Menace"
      },
      "text": "Amidala's ship is unable to sustain its hyperdrive and lands for repairs on the d
    },
    {
      "id": "59a6dec6-a7e6-43b9-a526-592857cf75ef",
      "score": 2.1473494,
      "metadata": {
        "title": "Star Wars: Episode I - The Phantom Menace"
      },
      "text": "Amidala's ship is unable to sustain its hyperdrive and lands for repairs on the d
    },
    {
      "id": "179a4d42-385f-407a-a1ec-735ff565b858",
      "score": 2.1473494,
      "metadata": {
        "title": "Star Wars: Episode I - The Phantom Menace"
      },
      "text": "Amidala's ship is unable to sustain its hyperdrive and lands for repairs on the d
    }
  ]
}
```


Change the above query and see what results you get! Try to find one that returns relevant results, and keep that (along with the output) for review.

> See Query building reference for descriptions of all possible parameters, operators and aggregations. You can also choose to build the query using the web interface (click the "Story Chunks" collection to query it), and then reproduce the query here.

```
In [25]: query = """enriched_text.entities:(relevance > 0.5,text::Leia),
                    enriched_text.entities.text:!Vader"""

         def disco_query(query, return_what="metadata.title, text",
                         environment_id=env_id, collection_id=col_id):
             return discovery.query(environment_id=env_id,
                                    collection_id=col_id,
                                    query_options={
                                       "query": query,
```

```
                                    "return": return_what
                                })

        results = disco_query(query=query)
        print(json.dumps(results, indent=2))
```


```
{
  "matching_results": 8,
  "results": [
    {
      "id": "da0c14e7-7640-4c8a-bd45-97b8b30df77d",
      "score": 6.736973,
      "metadata": {
        "title": "Star Wars: Episode V - The Empire Strikes Back"
      },
      "text": "On patrol, Han and Chewbacca discover the meteor Luke had planned to investigate
    },
    {
      "id": "ce6f8da4-8eb2-4d42-b622-e0d585843159",
      "score": 6.736973,
      "metadata": {
        "title": "Star Wars: Episode V - The Empire Strikes Back"
      },
      "text": "Vader goes back on his agreement with Lando to let Leia and Chewbacca stay in Clo
    },
    {
      "id": "616b3581-6e8f-4aa2-bbf6-1edd3ba1a380",
      "score": 6.736973,
      "metadata": {
        "title": "Star Wars: Episode V - The Empire Strikes Back"
      },
      "text": "On patrol, Han and Chewbacca discover the meteor Luke had planned to investigate
    },
    {
      "id": "f96303d8-80f7-4530-a5f1-b304bfc6e34c",
      "score": 6.736973,
      "metadata": {
        "title": "Star Wars: Episode V - The Empire Strikes Back"
      },
      "text": "Vader goes back on his agreement with Lando to let Leia and Chewbacca stay in Clo
    },
    {
      "id": "ef7b1bd6-b360-4245-aa66-499200a7f86d",
      "score": 6.736973,
      "metadata": {
        "title": "Star Wars: Episode V - The Empire Strikes Back"
      },
      "text": "On patrol, Han and Chewbacca discover the meteor Luke had planned to investigate
```

```
    },
    {
      "id": "cf5b23b3-f674-40e1-b87a-b960c46e191e",
      "score": 6.736973,
      "metadata": {
        "title": "Star Wars: Episode V - The Empire Strikes Back"
      },
      "text": "Vader goes back on his agreement with Lando to let Leia and Chewbacca stay in Clo
    },
    {
      "id": "7f99fb82-7245-44ac-92ab-bf3f4638d251",
      "score": 6.736973,
      "metadata": {
        "title": "Star Wars: Episode V - The Empire Strikes Back"
      },
      "text": "On patrol, Han and Chewbacca discover the meteor Luke had planned to investigate
    },
    {
      "id": "0b2b8f43-e31d-4683-832a-b755900ed7bb",
      "score": 6.736973,
      "metadata": {
        "title": "Star Wars: Episode V - The Empire Strikes Back"
      },
      "text": "Vader goes back on his agreement with Lando to let Leia and Chewbacca stay in Clo
    }
  ]
}
```


Then answer the questions below:

Q: What query did you try? Express it in plain words below.

A: I asked for all paragraphs that contain "Leia" but not "Vader" as entities, and which have a relevance of at least 0.5 for the entity "Leia". I asked for the title and text of those paragraphs to be returned.

Q: What answer did you get back from Watson? You only need to mention the relevant snippet of text from the paragraph(s) returned.

A: I got two paragraphs which mention Leia, but Vader also appear in both:

- "Han and Leia escape on the Millennium Falcon with C-3PO and Chewbacca, but their hyperdrive malfunctions. ... Vader summons bounty hunters..."
- "Vader goes back on his agreement with Lando to let Leia and Chewbacca stay in Cloud City..."

1.6 3. Parse natural language questions

In order to understand questions posed in natural language, we'll use another Watson service called Conversation. It can be used to design conversational agents or *chatbots* that exhibit complex behavior, but for the purpose of this project, we'll only use it to parse certain kinds of queries.

1.6.1 Create a Conversation service instance

Just like you did for the Discovery service, create an instance of the Conversation service. Then launch the associated tool from the service dashboard.

A *workspace* allows you to keep all the items you need for a particular application in one place, just like an *environment* in case of the Discovery service. Create one called "Bookworm" with a suitable description, such as "I know a lot of stories. Ask me a question!"

This should open up a blank workspace, where you can add intents, define the entities you want the agent to idenitfy and structure the overall dialog.

1.6.2 Add intents

An *intent* is the goal or purpose of a user's input. Create a set of intents (at least 3) that capture the different kinds of questions that you want the system to answer, e.g. *who*, *what* and *where*. Along with each intent, add a list of user examples or *utterances* that map to that intent.

For instance, you could enter the following examples for the *where* intent:

- Where is the Jedi temple located?
- Where was Luke born?

The Conversation service recommends at least 5 examples for each intent so that Watson learns how to recognize it. These don't have to be very precise, but more examples the better.

Feel free to create your own intents, based on the kinds of questions you want the system to answer, e.g. "How many ...", "What are the most common ..." etc. Each intent will need to be mapped to an appropriate query.

See **Defining intents** for a helpful video and further instructions.

Student's note: In order to answer the questions programmatically rather than by hand, I have moved up the cells where the connection to the Conversation service is made.

As before, let's connect to the Conversation service first. Remember to enter your service credentials below.

```
In [26]: # Connect to the Conversation service instance
         # TODO: Enter your username and password from the Service Credentials tab in service-cr
         conversation_creds = helper.fetch_credentials('conversation')
         conversation = watson_developer_cloud.ConversationV1(
             version='2017-02-03',
             username=conversation_creds['username'],
             password=conversation_creds['password'])
```

Fetch the workspace you just created called "Bookworm".

```
In [27]: wrk, wrk_id = helper.fetch_object(conversation, "workspace", "Bookworm")
         print(json.dumps(wrk, indent=2))
```

```
Found workspace: Bookworm (b5908cf6-3249-4b05-8969-16a347490af7)
{
  "name": "Bookworm",
  "created": "2017-08-20T19:51:14.836Z",
```

```
  "updated": "2017-08-21T17:49:07.759Z",
  "language": "en",
  "metadata": null,
  "description": "I know a lot of stories. Ask me a question!",
  "workspace_id": "b5908cf6-3249-4b05-8969-16a347490af7",
  "learning_opt_out": false,
  "status": "Available"
}
```

Q: What intents did you add to the Conversation service instance?
A: I added the following intents:

```
In [28]: [elt['intent'] for elt in conversation.list_intents(workspace_id=wrk_id)['intents']]

Out[28]: ['where', 'who', 'why']
```

Q: Pick one of these intents, and list at least 5 examples for the intent that you entered.
A: For the intent *why* I entered the following examples:

```
In [29]: [elt['text'] for elt in conversation.list_examples(workspace_id=wrk_id, intent="why")['

Out[29]: ["What is the purpose of Qui-Gon Jinn's and Obi-Wan Kenobi's mission to Naboo?",
          'What leads Amidala to the planet Tatooine?',
          'Why does Amidala travel to Coruscant?',
          'Why does Luke go to Dagobah?',
          'Why do Obi-Wan and Luke hire Han Solo and Chewbacca?']
```

1.6.3 Add entities

Once you have your intents set, let's tell the service what entities we want it to identify. One way to do this is using the tool interface, and entering them one-by-one.

> Go to **Defining entities** to see how that is done.

But that can be tedious! So let's refer back to the entities that the Discovery service identified, and load them in programmatically.

Collect all the entities from the Discovery service collection.

```
In [30]: # Get all the entities from the collection and group them by type
         response = discovery.query(environment_id=env_id, collection_id=col_id,
             query_options={
                 "return": "enriched_text.entities.type, enriched_text.entities.text"
             })

         # Group individual entities by type ("Person", "Location", etc.)
         entities_by_type = {}
         for document in response["results"]:
             for entity in document["enriched_text"]["entities"]:
```

```
            if entity["type"] not in entities_by_type:
                entities_by_type[entity["type"]] = set()
            entities_by_type[entity["type"]].add(entity["text"])

    # Ignore case to avoid duplicates
    for entity_type in entities_by_type:
        entities_by_type[entity_type] = {
            e.lower(): e for e in entities_by_type[entity_type]
        }.values()

    # Restructure for loading into Conversation workspace
    entities_grouped = [{
        "entity": entity_type,
        "values": [{"value": entity} for entity in entities]}
            for entity_type, entities in entities_by_type.items()]
    entities_grouped
```

Out[30]:
```
[{'entity': 'Person',
  'values': [{'value': 'Ren'},
   {'value': 'Darth Vader'},
   {'value': 'Lando Calrissian'},
   {'value': 'Luke'},
   {'value': 'Poe Dameron'},
   {'value': 'Kylo Ren'},
   {'value': 'Boba Fett'},
   {'value': 'Finn'},
   {'value': 'Anakin'},
   {'value': 'Lando'},
   {'value': 'Palpatine'},
   {'value': 'Maz Kanata'},
   {'value': 'Emperor Palpatine'},
   {'value': 'Princess Leia'},
   {'value': 'Han Solo'},
   {'value': 'Han'},
   {'value': 'Rey'},
   {'value': 'Hux'},
   {'value': 'Obi-Wan Kenobi'},
   {'value': 'Leia'},
   {'value': 'Ben'},
   {'value': 'Anakin Skywalker'},
   {'value': 'Vader'}]},
 {'entity': 'Organization',
  'values': [{'value': 'Rebels'},
   {'value': 'Falcon'},
   {'value': 'Rebel Alliance'}]},
 {'entity': 'Crime', 'values': [{'value': 'assault'}]},
 {'entity': 'Facility',
  'values': [{'value': "Jabba's palace"},
```

```
          {'value': 'Galactic Empire'},
          {'value': 'Imperial shuttle'}]},
         {'entity': 'Quantity', 'values': [{'value': 'Three years'}]},
         {'entity': 'City',
          'values': [{'value': 'Sarlacc'},
          {'value': 'San Tekka'},
          {'value': 'Jakku'},
          {'value': 'BB-8'},
          {'value': 'Dagobah'}]},
         {'entity': 'Company',
          'values': [{'value': 'General Hux'}, {'value': 'Starkiller Base'}]},
         {'entity': 'GeographicFeature', 'values': [{'value': 'Sarlacc pit'}]}]
```

Update the workspace with these entities and verify that have been added correctly.

```
In [31]: # Add these grouped entities to the Conversation workspace
         conversation.update_workspace(workspace_id=wrk_id, entities=entities_grouped)

         workspace_details = conversation.get_workspace(workspace_id=wrk_id, export=True)
         print(json.dumps(workspace_details["entities"], indent=2))
```


```
[
  {
    "type": null,
    "entity": "City",
    "source": null,
    "values": [
      {
        "value": "Dagobah",
        "created": "2017-08-21T17:53:53.259Z",
        "updated": "2017-08-21T17:53:53.259Z",
        "metadata": null,
        "synonyms": []
      },
      {
        "value": "BB-8",
        "created": "2017-08-21T17:53:53.259Z",
        "updated": "2017-08-21T17:53:53.259Z",
        "metadata": null,
        "synonyms": []
      },
      {
        "value": "Sarlacc",
        "created": "2017-08-21T17:53:53.259Z",
        "updated": "2017-08-21T17:53:53.259Z",
        "metadata": null,
        "synonyms": []
      },
```

```
    {
      "value": "San Tekka",
      "created": "2017-08-21T17:53:53.259Z",
      "updated": "2017-08-21T17:53:53.259Z",
      "metadata": null,
      "synonyms": []
    },
    {
      "value": "Jakku",
      "created": "2017-08-21T17:53:53.259Z",
      "updated": "2017-08-21T17:53:53.259Z",
      "metadata": null,
      "synonyms": []
    }
  ],
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "open_list": null,
  "description": null
},
{
  "type": null,
  "entity": "Quantity",
  "source": null,
  "values": [
    {
      "value": "Three years",
      "created": "2017-08-21T17:53:53.259Z",
      "updated": "2017-08-21T17:53:53.259Z",
      "metadata": null,
      "synonyms": []
    }
  ],
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "open_list": null,
  "description": null
},
{
  "type": null,
  "entity": "Facility",
  "source": null,
  "values": [
    {
      "value": "Jabba's palace",
      "created": "2017-08-21T17:53:53.259Z",
      "updated": "2017-08-21T17:53:53.259Z",
      "metadata": null,
```

```
        "synonyms": []
      },
      {
        "value": "Galactic Empire",
        "created": "2017-08-21T17:53:53.259Z",
        "updated": "2017-08-21T17:53:53.259Z",
        "metadata": null,
        "synonyms": []
      },
      {
        "value": "Imperial shuttle",
        "created": "2017-08-21T17:53:53.259Z",
        "updated": "2017-08-21T17:53:53.259Z",
        "metadata": null,
        "synonyms": []
      }
    ],
    "created": "2017-08-21T17:53:53.259Z",
    "updated": "2017-08-21T17:53:53.259Z",
    "open_list": null,
    "description": null
  },
  {
    "type": null,
    "entity": "Crime",
    "source": null,
    "values": [
      {
        "value": "assault",
        "created": "2017-08-21T17:53:53.259Z",
        "updated": "2017-08-21T17:53:53.259Z",
        "metadata": null,
        "synonyms": []
      }
    ],
    "created": "2017-08-21T17:53:53.259Z",
    "updated": "2017-08-21T17:53:53.259Z",
    "open_list": null,
    "description": null
  },
  {
    "type": null,
    "entity": "Organization",
    "source": null,
    "values": [
      {
        "value": "Rebels",
        "created": "2017-08-21T17:53:53.259Z",
```

```
      "updated": "2017-08-21T17:53:53.259Z",
      "metadata": null,
      "synonyms": []
    },
    {
      "value": "Rebel Alliance",
      "created": "2017-08-21T17:53:53.259Z",
      "updated": "2017-08-21T17:53:53.259Z",
      "metadata": null,
      "synonyms": []
    },
    {
      "value": "Falcon",
      "created": "2017-08-21T17:53:53.259Z",
      "updated": "2017-08-21T17:53:53.259Z",
      "metadata": null,
      "synonyms": []
    }
  ],
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "open_list": null,
  "description": null
},
{
  "type": null,
  "entity": "GeographicFeature",
  "source": null,
  "values": [
    {
      "value": "Sarlacc pit",
      "created": "2017-08-21T17:53:53.259Z",
      "updated": "2017-08-21T17:53:53.259Z",
      "metadata": null,
      "synonyms": []
    }
  ],
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "open_list": null,
  "description": null
},
{
  "type": null,
  "entity": "Company",
  "source": null,
  "values": [
    {
```

```
      "value": "Starkiller Base",
      "created": "2017-08-21T17:53:53.259Z",
      "updated": "2017-08-21T17:53:53.259Z",
      "metadata": null,
      "synonyms": []
    },
    {
      "value": "General Hux",
      "created": "2017-08-21T17:53:53.259Z",
      "updated": "2017-08-21T17:53:53.259Z",
      "metadata": null,
      "synonyms": []
    }
  ],
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "open_list": null,
  "description": null
},
{
  "type": null,
  "entity": "Person",
  "source": null,
  "values": [
    {
      "value": "Ben",
      "created": "2017-08-21T17:53:53.259Z",
      "updated": "2017-08-21T17:53:53.259Z",
      "metadata": null,
      "synonyms": []
    },
    {
      "value": "Anakin",
      "created": "2017-08-21T17:53:53.259Z",
      "updated": "2017-08-21T17:53:53.259Z",
      "metadata": null,
      "synonyms": []
    },
    {
      "value": "Han Solo",
      "created": "2017-08-21T17:53:53.259Z",
      "updated": "2017-08-21T17:53:53.259Z",
      "metadata": null,
      "synonyms": []
    },
    {
      "value": "Rey",
      "created": "2017-08-21T17:53:53.259Z",
```

```
  "updated": "2017-08-21T17:53:53.259Z",
  "metadata": null,
  "synonyms": []
},
{
  "value": "Han",
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "metadata": null,
  "synonyms": []
},
{
  "value": "Anakin Skywalker",
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "metadata": null,
  "synonyms": []
},
{
  "value": "Finn",
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "metadata": null,
  "synonyms": []
},
{
  "value": "Darth Vader",
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "metadata": null,
  "synonyms": []
},
{
  "value": "Ren",
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "metadata": null,
  "synonyms": []
},
{
  "value": "Lando",
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "metadata": null,
  "synonyms": []
},
{
  "value": "Lando Calrissian",
```

```
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "metadata": null,
  "synonyms": []
},
{
  "value": "Kylo Ren",
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "metadata": null,
  "synonyms": []
},
{
  "value": "Princess Leia",
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "metadata": null,
  "synonyms": []
},
{
  "value": "Vader",
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "metadata": null,
  "synonyms": []
},
{
  "value": "Palpatine",
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "metadata": null,
  "synonyms": []
},
{
  "value": "Boba Fett",
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "metadata": null,
  "synonyms": []
},
{
  "value": "Hux",
  "created": "2017-08-21T17:53:53.259Z",
  "updated": "2017-08-21T17:53:53.259Z",
  "metadata": null,
  "synonyms": []
},
{
```

```
        "value": "Emperor Palpatine",
        "created": "2017-08-21T17:53:53.259Z",
        "updated": "2017-08-21T17:53:53.259Z",
        "metadata": null,
        "synonyms": []
      },
      {
        "value": "Leia",
        "created": "2017-08-21T17:53:53.259Z",
        "updated": "2017-08-21T17:53:53.259Z",
        "metadata": null,
        "synonyms": []
      },
      {
        "value": "Poe Dameron",
        "created": "2017-08-21T17:53:53.259Z",
        "updated": "2017-08-21T17:53:53.259Z",
        "metadata": null,
        "synonyms": []
      },
      {
        "value": "Maz Kanata",
        "created": "2017-08-21T17:53:53.259Z",
        "updated": "2017-08-21T17:53:53.259Z",
        "metadata": null,
        "synonyms": []
      },
      {
        "value": "Obi-Wan Kenobi",
        "created": "2017-08-21T17:53:53.259Z",
        "updated": "2017-08-21T17:53:53.259Z",
        "metadata": null,
        "synonyms": []
      },
      {
        "value": "Luke",
        "created": "2017-08-21T17:53:53.259Z",
        "updated": "2017-08-21T17:53:53.259Z",
        "metadata": null,
        "synonyms": []
      }
    ],
    "created": "2017-08-21T17:53:53.259Z",
    "updated": "2017-08-21T17:53:53.259Z",
    "open_list": null,
    "description": null
  }
]
```

***Note**: Ensure that at least 3 entity types, with at least 1 example entity each have been added.*

Here is what the list of entities should look like through the Conversation tool.

Student's note: I have added one more entity or I can get no relevant answer to any query at all in Section 4.

```
In [32]: # conversation.create_entity(workspace_id=wrk_id, entity="Learning", values=['teach', '

         # For some reason feeding in an array of values upon creation of the entity raises an e

         # conversation.create_entity(workspace_id=wrk_id, entity="Learning", fuzzy_match=True)

         # conversation.create_entity(workspace_id=wrk_id, entity="Travel", fuzzy_match=True)

In [33]: # for value in ['teach', 'learn', 'train', 'study']:
         #     conversation.create_value(workspace_id=wrk_id, entity="Learning", value=value)

         # for value in ['travel', 'go to', 'fly', 'accompany']:
         #     conversation.create_value(workspace_id=wrk_id, entity="Travel", value=value)
```

Q: Name 3 entity types that were added, with at least 1 example entity each (e.g. entity type: *City*, example: *Los Angeles*).

A: Here is the full list of the entities and their instances which have been added to my Conversation workspace:

```
In [34]: # Get updated workspace details.
         workspace_details = conversation.get_workspace(workspace_id=wrk_id, export=True)
         wrk_entities = workspace_details['entities']
         wrk_entities = [{'entity': entity['entity'], 'values': entity['values']} \
                         for entity in wrk_entities]

         # Keep only the values, without the metadata.
         for entity in wrk_entities:
             entity['values'] = [value['value'] for value in entity['values']]

         pprint(wrk_entities)
```

```
[{'entity': 'City',
  'values': ['Dagobah', 'BB-8', 'Sarlacc', 'San Tekka', 'Jakku']},
 {'entity': 'Quantity', 'values': ['Three years']},
 {'entity': 'Facility',
  'values': ["Jabba's palace", 'Galactic Empire', 'Imperial shuttle']},
 {'entity': 'Crime', 'values': ['assault']},
 {'entity': 'Organization', 'values': ['Rebels', 'Rebel Alliance', 'Falcon']},
 {'entity': 'GeographicFeature', 'values': ['Sarlacc pit']},
 {'entity': 'Company', 'values': ['Starkiller Base', 'General Hux']},
 {'entity': 'Person',
  'values': ['Ben',
```

```
            'Anakin',
            'Han Solo',
            'Rey',
            'Han',
            'Anakin Skywalker',
            'Finn',
            'Darth Vader',
            'Ren',
            'Lando',
            'Lando Calrissian',
            'Kylo Ren',
            'Princess Leia',
            'Vader',
            'Palpatine',
            'Boba Fett',
            'Hux',
            'Emperor Palpatine',
            'Leia',
            'Poe Dameron',
            'Maz Kanata',
            'Obi-Wan Kenobi',
            'Luke']}]
```

To name only three entities and an example of each:

```
In [35]: i = 1
         for entity in wrk_entities:
             print("{idx}. Entity: {ent} - Example: {ex}".format(idx=i,
                                                                 ent=entity['entity'],
                                                                 ex=entity['values'][0]))
             i += 1
             if i == 4:
                 break
```

```
1. Entity: City - Example: Dagobah
2. Entity: Quantity - Example: Three years
3. Entity: Facility - Example: Jabba's palace
```

1.6.4 Design dialog flow

As a final step in creating the Conversation interface, let's design a typical dialog with a user. The most intuitive way to do this is to use the Dialog tab in the tool. Here, you can add *nodes* that capture different stages in the dialog flow, and connect them in a meaningful way.

Go ahead and add at least 3 dialog nodes. Specify the triggers in terms of the intents and entities that you'd like to match, and an optional intermediate response like "Let me find that out for you." The actual response will be fetched by querying the Discovery service.

Here is what the dialog nodes should look like.

Q: Specify 3 dialog nodes you added, along with the trigger (intent and/or entities) for each.

A: I added the following nodes: - Agent, triggered by intent #who or entities @Person, @Company, or @Organization. - Location, triggered by intent #where or entities @City, @Facility, or @GeographicFeatures. - Cause_purpose, triggered by intent #why.

1.6.5 Test dialog

Let's run through a test dialog to demonstrate how the system transitions to one of the nodes you defined above.

```
In [37]: # Testing the dialog flow

         # Start conversation with a blank message
         results = conversation.message(workspace_id=wrk_id, message_input={})
         context = results["context"]

         # Then ask a sample question
         question= "Who is Luke's father?"
         results = conversation.message(workspace_id=wrk_id, message_input={
             "text": question,
             "context": context
         })
         print(json.dumps(results, indent=2))
```


```
{
  "intents": [
    {
      "intent": "who",
      "confidence": 0.9119018554687501
    }
  ],
  "entities": [
    {
      "entity": "Person",
      "location": [
        7,
        11
      ],
      "value": "Luke",
      "confidence": 1
    }
  ],
  "input": {
    "text": "Who is Luke's father?",
    "context": {
      "conversation_id": "f12ad69d-de88-41df-aec6-57d6cbc8b9d1",
      "system": {
        "dialog_stack": [
```

```
          {
            "dialog_node": "root"
          }
        ],
        "dialog_turn_counter": 1,
        "dialog_request_counter": 1
      }
    }
  },
  "output": {
    "text": [
      "Let me find that out for you."
    ],
    "nodes_visited": [
      "Agent"
    ],
    "log_messages": []
  },
  "context": {
    "conversation_id": "23ef9c34-e27f-488f-ae87-dd89ae60c085",
    "system": {
      "dialog_stack": [
        {
          "dialog_node": "root"
        }
      ],
      "dialog_turn_counter": 1,
      "dialog_request_counter": 1,
      "_node_output_map": {
        "Agent": [
          0,
          0,
          1,
          2
        ]
      },
      "branch_exited": true,
      "branch_exited_reason": "completed"
    }
  }
}
```


1.7 4. Query document collection to fetch answers

The Discovery service includes a simple mechanism to make queries against your enriched collection of documents. But you have a lot of control over what fields are searched, how results are aggregated and values are returned.

1.7.1 Process sample question

Choose a sample nautal language question to ask, and run it through the Conversation service, just like you did above when testing dialog flow.

Student's note: I have manually edited some of the values of the @Person entity, in the GUI interface, to make synonyms those that refer to the same person. Without this Obi-Wan is not even identified as a person, as Watson's Discovery has only identified Obi-Wan Kenobi as an entity.

```
In [99]: # TODO: Run a sample question through Conversation service

         # Start conversation with a blank message
         results = conversation.message(workspace_id=wrk_id, message_input={})
         context = results["context"]

         # Then ask a question
         question = "Who is Anakin's dark mentor?"
         # question = "Who is Anakin's mother?"
         # question = "Where does Leia hide the plans?"
         # question = "Who is Princess Leia?"
         # question = "Who captures Princess Leia's ship?"
         # question = "Who is Han?"
         # question = "Who is Han Solo?"
         # question = "Who is Palpatine?"
         # question = "Who severs Windu's hand?"
         # question = "Who is the leader of the Rebels?"
         # question = "Who is the Rebel leader?"
         # question = "Who trains Anakin as a Jedi?"
         # question = "Who trains Luke as a Jedi?"
         # question = "Why does Obi-Wan wish that Luke travel with him to Alderaan?"
         # question = "Why does Queen Amidala's ship land on Tatooine?"
         # question = "Who is the Death Star's commanding officer?"
         results = conversation.message(workspace_id=wrk_id,
                                        message_input={"text": question,
                                                       "context": context})
         print(json.dumps(results, indent=2))
```


```
{
  "intents": [
    {
      "intent": "who",
      "confidence": 0.9455078125
    }
  ],
  "entities": [
    {
      "entity": "Person",
      "location": [
        7,
        13
```

```
    ],
    "value": "Anakin",
    "confidence": 1
  }
],
"input": {
  "text": "Who is Anakin's dark mentor?",
  "context": {
    "conversation_id": "8549bb04-7a9a-4619-aeb5-6a6bce1c0f56",
    "system": {
      "dialog_stack": [
        {
          "dialog_node": "root"
        }
      ],
      "dialog_turn_counter": 1,
      "dialog_request_counter": 1
    }
  }
},
"output": {
  "text": [
    "I'm looking for an answer!"
  ],
  "nodes_visited": [
    "Agent"
  ],
  "log_messages": []
},
"context": {
  "conversation_id": "691ed749-51e7-4d2f-ae01-77a1acea7232",
  "system": {
    "dialog_stack": [
      {
        "dialog_node": "root"
      }
    ],
    "dialog_turn_counter": 1,
    "dialog_request_counter": 1,
    "_node_output_map": {
      "Agent": [
        0,
        1,
        2,
        0
      ]
    },
    "branch_exited": true,
```

```
      "branch_exited_reason": "completed"
    }
  }
}
```

Now extract the intent and entities identified in the question, and optionally what dialog node was triggered (in case you need it later to customize your response). Some sample code is provided below, but you may need to modify it.

```
In [102]: # results.keys()

In [103]: # results

In [104]: # TODO: Identify the intent(s) the user expressed (typically a single one)
          query_intents = [intent["intent"] for intent in results["intents"]]
          print("Intent(s): {}".format(list(query_intents)))

          # TODO: Extract the entities found in the question text
          query_entities = [entity["value"] for entity in results["entities"]]
          print("Entities: {}".format(list(query_entities)))

          # TODO: (optional) Find out what dialog node was triggered
          dialogue_node = list(results['context']['system']['_node_output_map'].keys())[0]
          print("Dialogue node: {}".format(dialogue_node))
```

```
Intent(s): ['who']
Entities: ['Anakin']
Dialogue node: Agent
```

1.7.2 Query the collection

Design a query based on the information extracted above, and run it against the document collection. The sample query provided below simple looks for all the entities in the raw `text` field. Modify it to suit your needs.

Take a look at the API Reference to learn more about the query options available, and for more guidance see this documentation page.

***Note**: You may want to design different queries based on the intent / dialog node that was triggered.*

```
In [105]: # EXAMPLE QUERY

          # TODO: Query the Discovery service based on the intent and entities

          # query_results = discovery.query(environment_id=env_id, collection_id=col_id,
          #     query_options={
          #         "query": "text:{}".format(",".join("\"{}\"".format(e) for e in query_entitie
          #         "return": "text"
          #     })
          # print(json.dumps(query_results, indent=2))
```

```
In [106]: # ", ".join("enriched_text.entities:(relevance > 0.8,text::{})".format(e) for e in que

In [107]: # TODO: Query the Discovery service based on the intent and entities

          query_results = discovery.query(environment_id=env_id, collection_id=col_id,
              query_options={
                  "query": ",".join("enriched_text.entities:(relevance > 0.8, text::{})".\
                                    format(e) for e in query_entities),
                  "return": """text,
                               enriched_text.entities.type,
                               enriched_text.entities.text,
                               enriched_text.entities.relevance"""
              })

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


```
{
  "matching_results": 8,
  "results": [
    {
      "id": "bde4092f-2504-40d7-9498-a11589979213",
      "score": 7.1049113,
      "enriched_text": {
        "entities": [
          {
            "text": "Anakin",
            "type": "Person",
            "relevance": 0.942309
          },
          {
            "text": "Palpatine",
            "type": "Person",
            "relevance": 0.669482
          },
          {
            "text": "Darth Sidious",
            "type": "Person",
            "relevance": 0.464408
          },
          {
            "text": "Jedi Council",
            "type": "Organization",
            "relevance": 0.302401
          },
          {
            "text": "Darth Vader",
            "type": "Person",
            "relevance": 0.242974
```

```
    },
    {
      "text": "Jedi Temple",
      "type": "Facility",
      "relevance": 0.228208
    },
    {
      "text": "Mace Windu",
      "type": "Person",
      "relevance": 0.194849
    },
    {
      "text": "Padm\u00e9",
      "type": "Person",
      "relevance": 0.177711
    },
    {
      "text": "Yoda",
      "type": "Person",
      "relevance": 0.167374
    },
    {
      "text": "Kashyyyk",
      "type": "City",
      "relevance": 0.158443
    },
    {
      "text": "representative",
      "type": "JobTitle",
      "relevance": 0.146907
    },
    {
      "text": "Coruscant",
      "type": "City",
      "relevance": 0.142806
    },
    {
      "text": "Senate",
      "type": "Organization",
      "relevance": 0.127914
    },
    {
      "text": "Mustafar",
      "type": "City",
      "relevance": 0.122259
    }
  ]
},
```

```
  "text": "Palpatine appoints Anakin to the Jedi Council as his representative, but the Coun
},
{
  "id": "d8c42ae4-4279-43b7-b908-1799b6e26a1f",
  "score": 7.1049113,
  "enriched_text": {
    "entities": [
      {
        "text": "Anakin",
        "type": "Person",
        "relevance": 0.938404
      },
      {
        "text": "Shmi",
        "type": "Person",
        "relevance": 0.832747
      },
      {
        "text": "Cliegg Lars",
        "type": "Person",
        "relevance": 0.32508
      },
      {
        "text": "Jango Fett",
        "type": "Person",
        "relevance": 0.254378
      },
      {
        "text": "Obi-Wan",
        "type": "Person",
        "relevance": 0.226119
      },
      {
        "text": "Tusken Raiders",
        "type": "Organization",
        "relevance": 0.20842
      },
      {
        "text": "Padm\u00e9",
        "type": "Person",
        "relevance": 0.192448
      },
      {
        "text": "Tusken",
        "type": "City",
        "relevance": 0.172401
      },
      {
```

```
        "text": "Boba",
        "type": "Person",
        "relevance": 0.156956
      },
      {
        "text": "Geonosis",
        "type": "City",
        "relevance": 0.131101
      }
    ]
  },
  "text": "Obi-Wan's investigation leads him to the remote ocean planet Kamino, where he dis
},
{
  "id": "fac02b54-bf41-4348-85eb-825e45e5e606",
  "score": 7.1049113,
  "enriched_text": {
    "entities": [
      {
        "text": "Anakin",
        "type": "Person",
        "relevance": 0.942309
      },
      {
        "text": "Palpatine",
        "type": "Person",
        "relevance": 0.669482
      },
      {
        "text": "Darth Sidious",
        "type": "Person",
        "relevance": 0.464408
      },
      {
        "text": "Jedi Council",
        "type": "Organization",
        "relevance": 0.302401
      },
      {
        "text": "Darth Vader",
        "type": "Person",
        "relevance": 0.242974
      },
      {
        "text": "Jedi Temple",
        "type": "Facility",
        "relevance": 0.228208
      },
```

```
      {
        "text": "Mace Windu",
        "type": "Person",
        "relevance": 0.194849
      },
      {
        "text": "Padm\u00e9",
        "type": "Person",
        "relevance": 0.177711
      },
      {
        "text": "Yoda",
        "type": "Person",
        "relevance": 0.167374
      },
      {
        "text": "Kashyyyk",
        "type": "City",
        "relevance": 0.158443
      },
      {
        "text": "representative",
        "type": "JobTitle",
        "relevance": 0.146907
      },
      {
        "text": "Coruscant",
        "type": "City",
        "relevance": 0.142806
      },
      {
        "text": "Senate",
        "type": "Organization",
        "relevance": 0.127914
      },
      {
        "text": "Mustafar",
        "type": "City",
        "relevance": 0.122259
      }
    ]
  },
  "text": "Palpatine appoints Anakin to the Jedi Council as his representative, but the Coun
},
{
  "id": "c5f9ef4c-e9ed-49a5-b9aa-b8fbe271d75d",
  "score": 7.1049113,
  "enriched_text": {
```

```
"entities": [
  {
    "text": "Anakin",
    "type": "Person",
    "relevance": 0.938404
  },
  {
    "text": "Shmi",
    "type": "Person",
    "relevance": 0.832747
  },
  {
    "text": "Cliegg Lars",
    "type": "Person",
    "relevance": 0.32508
  },
  {
    "text": "Jango Fett",
    "type": "Person",
    "relevance": 0.254378
  },
  {
    "text": "Obi-Wan",
    "type": "Person",
    "relevance": 0.226119
  },
  {
    "text": "Tusken Raiders",
    "type": "Organization",
    "relevance": 0.20842
  },
  {
    "text": "Padm\u00e9",
    "type": "Person",
    "relevance": 0.192448
  },
  {
    "text": "Tusken",
    "type": "City",
    "relevance": 0.172401
  },
  {
    "text": "Boba",
    "type": "Person",
    "relevance": 0.156956
  },
  {
    "text": "Geonosis",
```

```
        "type": "City",
        "relevance": 0.131101
      }
    ]
  },
  "text": "Obi-Wan's investigation leads him to the remote ocean planet Kamino, where he dis
},
{
  "id": "a2bcd931-80ec-42c8-ae3b-0a0aa218d8c8",
  "score": 7.1049113,
  "enriched_text": {
    "entities": [
      {
        "text": "Anakin",
        "type": "Person",
        "relevance": 0.942309
      },
      {
        "text": "Palpatine",
        "type": "Person",
        "relevance": 0.669482
      },
      {
        "text": "Darth Sidious",
        "type": "Person",
        "relevance": 0.464408
      },
      {
        "text": "Jedi Council",
        "type": "Organization",
        "relevance": 0.302401
      },
      {
        "text": "Darth Vader",
        "type": "Person",
        "relevance": 0.242974
      },
      {
        "text": "Jedi Temple",
        "type": "Facility",
        "relevance": 0.228208
      },
      {
        "text": "Mace Windu",
        "type": "Person",
        "relevance": 0.194849
      },
      {
```

```
        "text": "Padm\u00e9",
        "type": "Person",
        "relevance": 0.177711
      },
      {
        "text": "Yoda",
        "type": "Person",
        "relevance": 0.167374
      },
      {
        "text": "Kashyyyk",
        "type": "City",
        "relevance": 0.158443
      },
      {
        "text": "representative",
        "type": "JobTitle",
        "relevance": 0.146907
      },
      {
        "text": "Coruscant",
        "type": "City",
        "relevance": 0.142806
      },
      {
        "text": "Senate",
        "type": "Organization",
        "relevance": 0.127914
      },
      {
        "text": "Mustafar",
        "type": "City",
        "relevance": 0.122259
      }
    ]
  },
  "text": "Palpatine appoints Anakin to the Jedi Council as his representative, but the Coun
},
{
  "id": "40a99bed-8634-4417-b8a8-1126916ea23e",
  "score": 7.1049113,
  "enriched_text": {
    "entities": [
      {
        "text": "Anakin",
        "type": "Person",
        "relevance": 0.938404
      },
```

```
    {
      "text": "Shmi",
      "type": "Person",
      "relevance": 0.832747
    },
    {
      "text": "Cliegg Lars",
      "type": "Person",
      "relevance": 0.32508
    },
    {
      "text": "Jango Fett",
      "type": "Person",
      "relevance": 0.254378
    },
    {
      "text": "Obi-Wan",
      "type": "Person",
      "relevance": 0.226119
    },
    {
      "text": "Tusken Raiders",
      "type": "Organization",
      "relevance": 0.20842
    },
    {
      "text": "Padm\u00e9",
      "type": "Person",
      "relevance": 0.192448
    },
    {
      "text": "Tusken",
      "type": "City",
      "relevance": 0.172401
    },
    {
      "text": "Boba",
      "type": "Person",
      "relevance": 0.156956
    },
    {
      "text": "Geonosis",
      "type": "City",
      "relevance": 0.131101
    }
  ]
},
"text": "Obi-Wan's investigation leads him to the remote ocean planet Kamino, where he dis
```

```
},
{
  "id": "c506239a-f772-43c7-952b-b25180268750",
  "score": 7.1049113,
  "enriched_text": {
    "entities": [
      {
        "text": "Anakin",
        "type": "Person",
        "relevance": 0.942309
      },
      {
        "text": "Palpatine",
        "type": "Person",
        "relevance": 0.669482
      },
      {
        "text": "Darth Sidious",
        "type": "Person",
        "relevance": 0.464408
      },
      {
        "text": "Jedi Council",
        "type": "Organization",
        "relevance": 0.302401
      },
      {
        "text": "Darth Vader",
        "type": "Person",
        "relevance": 0.242974
      },
      {
        "text": "Jedi Temple",
        "type": "Facility",
        "relevance": 0.228208
      },
      {
        "text": "Mace Windu",
        "type": "Person",
        "relevance": 0.194849
      },
      {
        "text": "Padm\u00e9",
        "type": "Person",
        "relevance": 0.177711
      },
      {
        "text": "Yoda",
```

```
        "type": "Person",
        "relevance": 0.167374
      },
      {
        "text": "Kashyyyk",
        "type": "City",
        "relevance": 0.158443
      },
      {
        "text": "representative",
        "type": "JobTitle",
        "relevance": 0.146907
      },
      {
        "text": "Coruscant",
        "type": "City",
        "relevance": 0.142806
      },
      {
        "text": "Senate",
        "type": "Organization",
        "relevance": 0.127914
      },
      {
        "text": "Mustafar",
        "type": "City",
        "relevance": 0.122259
      }
    ]
  },
  "text": "Palpatine appoints Anakin to the Jedi Council as his representative, but the Coun
},
{
  "id": "0e46ad2e-bd03-471c-8bf0-ac68232fa4a0",
  "score": 7.1049113,
  "enriched_text": {
    "entities": [
      {
        "text": "Anakin",
        "type": "Person",
        "relevance": 0.938404
      },
      {
        "text": "Shmi",
        "type": "Person",
        "relevance": 0.832747
      },
      {
```

```
            "text": "Cliegg Lars",
            "type": "Person",
            "relevance": 0.32508
          },
          {
            "text": "Jango Fett",
            "type": "Person",
            "relevance": 0.254378
          },
          {
            "text": "Obi-Wan",
            "type": "Person",
            "relevance": 0.226119
          },
          {
            "text": "Tusken Raiders",
            "type": "Organization",
            "relevance": 0.20842
          },
          {
            "text": "Padm\u00e9",
            "type": "Person",
            "relevance": 0.192448
          },
          {
            "text": "Tusken",
            "type": "City",
            "relevance": 0.172401
          },
          {
            "text": "Boba",
            "type": "Person",
            "relevance": 0.156956
          },
          {
            "text": "Geonosis",
            "type": "City",
            "relevance": 0.131101
          }
        ]
      },
      "text": "Obi-Wan's investigation leads him to the remote ocean planet Kamino, where he dis
    }
  ]
}
```

1.7.3 Process returned results

If you properly structure the query, Watson is able to do a pretty good job of finding the relevant information. But the result returned is a JSON object. Now your task is to convert that result into an appropriate response that best addresses the original natural language question that was asked.

E.g. if the question was "Who saved Han Solo from Jabba the Hutt?" the answer should ideally just be "The Rebels" and not the entire paragraph describing Han Solo's rescue. But that can be a backup response if you cannot be more specific.

***Note**: You may have to go back to the previous step and modify the query, especially what you want the Discovery service to return, and this may depend on the intent / dialog node triggered. E.g. study the different parts of a "relation" structure to see how you might construct queries to match them.*

```
In [111]: # TODO: Process returned results and compose an appropriate response
          results = query_results['results']
          results
```

```
Out[111]: [{'enriched_text': {'entities': [{'relevance': 0.942309,
                'text': 'Anakin',
                'type': 'Person'},
               {'relevance': 0.669482, 'text': 'Palpatine', 'type': 'Person'},
               {'relevance': 0.464408, 'text': 'Darth Sidious', 'type': 'Person'},
               {'relevance': 0.302401, 'text': 'Jedi Council', 'type': 'Organization'},
               {'relevance': 0.242974, 'text': 'Darth Vader', 'type': 'Person'},
               {'relevance': 0.228208, 'text': 'Jedi Temple', 'type': 'Facility'},
               {'relevance': 0.194849, 'text': 'Mace Windu', 'type': 'Person'},
               {'relevance': 0.177711, 'text': 'Padmé', 'type': 'Person'},
               {'relevance': 0.167374, 'text': 'Yoda', 'type': 'Person'},
               {'relevance': 0.158443, 'text': 'Kashyyyk', 'type': 'City'},
               {'relevance': 0.146907, 'text': 'representative', 'type': 'JobTitle'},
               {'relevance': 0.142806, 'text': 'Coruscant', 'type': 'City'},
               {'relevance': 0.127914, 'text': 'Senate', 'type': 'Organization'},
               {'relevance': 0.122259, 'text': 'Mustafar', 'type': 'City'}]},
             'id': 'bde4092f-2504-40d7-9498-a11589979213',
             'score': 7.1049113,
             'text': "Palpatine appoints Anakin to the Jedi Council as his representative, but th
            {'enriched_text': {'entities': [{'relevance': 0.938404,
                'text': 'Anakin',
                'type': 'Person'},
               {'relevance': 0.832747, 'text': 'Shmi', 'type': 'Person'},
               {'relevance': 0.32508, 'text': 'Cliegg Lars', 'type': 'Person'},
               {'relevance': 0.254378, 'text': 'Jango Fett', 'type': 'Person'},
               {'relevance': 0.226119, 'text': 'Obi-Wan', 'type': 'Person'},
               {'relevance': 0.20842, 'text': 'Tusken Raiders', 'type': 'Organization'},
               {'relevance': 0.192448, 'text': 'Padmé', 'type': 'Person'},
               {'relevance': 0.172401, 'text': 'Tusken', 'type': 'City'},
               {'relevance': 0.156956, 'text': 'Boba', 'type': 'Person'},
               {'relevance': 0.131101, 'text': 'Geonosis', 'type': 'City'}]},
             'id': 'd8c42ae4-4279-43b7-b908-1799b6e26a1f',
```

```
 'score': 7.1049113,
 'text': "Obi-Wan's investigation leads him to the remote ocean planet Kamino, where
{'enriched_text': {'entities': [{'relevance': 0.942309,
    'text': 'Anakin',
    'type': 'Person'},
   {'relevance': 0.669482, 'text': 'Palpatine', 'type': 'Person'},
   {'relevance': 0.464408, 'text': 'Darth Sidious', 'type': 'Person'},
   {'relevance': 0.302401, 'text': 'Jedi Council', 'type': 'Organization'},
   {'relevance': 0.242974, 'text': 'Darth Vader', 'type': 'Person'},
   {'relevance': 0.228208, 'text': 'Jedi Temple', 'type': 'Facility'},
   {'relevance': 0.194849, 'text': 'Mace Windu', 'type': 'Person'},
   {'relevance': 0.177711, 'text': 'Padmé', 'type': 'Person'},
   {'relevance': 0.167374, 'text': 'Yoda', 'type': 'Person'},
   {'relevance': 0.158443, 'text': 'Kashyyyk', 'type': 'City'},
   {'relevance': 0.146907, 'text': 'representative', 'type': 'JobTitle'},
   {'relevance': 0.142806, 'text': 'Coruscant', 'type': 'City'},
   {'relevance': 0.127914, 'text': 'Senate', 'type': 'Organization'},
   {'relevance': 0.122259, 'text': 'Mustafar', 'type': 'City'}]},
 'id': 'fac02b54-bf41-4348-85eb-825e45e5e606',
 'score': 7.1049113,
 'text': "Palpatine appoints Anakin to the Jedi Council as his representative, but th
{'enriched_text': {'entities': [{'relevance': 0.938404,
    'text': 'Anakin',
    'type': 'Person'},
   {'relevance': 0.832747, 'text': 'Shmi', 'type': 'Person'},
   {'relevance': 0.32508, 'text': 'Cliegg Lars', 'type': 'Person'},
   {'relevance': 0.254378, 'text': 'Jango Fett', 'type': 'Person'},
   {'relevance': 0.226119, 'text': 'Obi-Wan', 'type': 'Person'},
   {'relevance': 0.20842, 'text': 'Tusken Raiders', 'type': 'Organization'},
   {'relevance': 0.192448, 'text': 'Padmé', 'type': 'Person'},
   {'relevance': 0.172401, 'text': 'Tusken', 'type': 'City'},
   {'relevance': 0.156956, 'text': 'Boba', 'type': 'Person'},
   {'relevance': 0.131101, 'text': 'Geonosis', 'type': 'City'}]},
 'id': 'c5f9ef4c-e9ed-49a5-b9aa-b8fbe271d75d',
 'score': 7.1049113,
 'text': "Obi-Wan's investigation leads him to the remote ocean planet Kamino, where
{'enriched_text': {'entities': [{'relevance': 0.942309,
    'text': 'Anakin',
    'type': 'Person'},
   {'relevance': 0.669482, 'text': 'Palpatine', 'type': 'Person'},
   {'relevance': 0.464408, 'text': 'Darth Sidious', 'type': 'Person'},
   {'relevance': 0.302401, 'text': 'Jedi Council', 'type': 'Organization'},
   {'relevance': 0.242974, 'text': 'Darth Vader', 'type': 'Person'},
   {'relevance': 0.228208, 'text': 'Jedi Temple', 'type': 'Facility'},
   {'relevance': 0.194849, 'text': 'Mace Windu', 'type': 'Person'},
   {'relevance': 0.177711, 'text': 'Padmé', 'type': 'Person'},
   {'relevance': 0.167374, 'text': 'Yoda', 'type': 'Person'},
   {'relevance': 0.158443, 'text': 'Kashyyyk', 'type': 'City'},
```

```
   {'relevance': 0.146907, 'text': 'representative', 'type': 'JobTitle'},
   {'relevance': 0.142806, 'text': 'Coruscant', 'type': 'City'},
   {'relevance': 0.127914, 'text': 'Senate', 'type': 'Organization'},
   {'relevance': 0.122259, 'text': 'Mustafar', 'type': 'City'}]},
 'id': 'a2bcd931-80ec-42c8-ae3b-0a0aa218d8c8',
 'score': 7.1049113,
 'text': "Palpatine appoints Anakin to the Jedi Council as his representative, but th
{'enriched_text': {'entities': [{'relevance': 0.938404,
    'text': 'Anakin',
    'type': 'Person'},
   {'relevance': 0.832747, 'text': 'Shmi', 'type': 'Person'},
   {'relevance': 0.32508, 'text': 'Cliegg Lars', 'type': 'Person'},
   {'relevance': 0.254378, 'text': 'Jango Fett', 'type': 'Person'},
   {'relevance': 0.226119, 'text': 'Obi-Wan', 'type': 'Person'},
   {'relevance': 0.20842, 'text': 'Tusken Raiders', 'type': 'Organization'},
   {'relevance': 0.192448, 'text': 'Padmé', 'type': 'Person'},
   {'relevance': 0.172401, 'text': 'Tusken', 'type': 'City'},
   {'relevance': 0.156956, 'text': 'Boba', 'type': 'Person'},
   {'relevance': 0.131101, 'text': 'Geonosis', 'type': 'City'}]},
 'id': '40a99bed-8634-4417-b8a8-1126916ea23e',
 'score': 7.1049113,
 'text': "Obi-Wan's investigation leads him to the remote ocean planet Kamino, where
{'enriched_text': {'entities': [{'relevance': 0.942309,
    'text': 'Anakin',
    'type': 'Person'},
   {'relevance': 0.669482, 'text': 'Palpatine', 'type': 'Person'},
   {'relevance': 0.464408, 'text': 'Darth Sidious', 'type': 'Person'},
   {'relevance': 0.302401, 'text': 'Jedi Council', 'type': 'Organization'},
   {'relevance': 0.242974, 'text': 'Darth Vader', 'type': 'Person'},
   {'relevance': 0.228208, 'text': 'Jedi Temple', 'type': 'Facility'},
   {'relevance': 0.194849, 'text': 'Mace Windu', 'type': 'Person'},
   {'relevance': 0.177711, 'text': 'Padmé', 'type': 'Person'},
   {'relevance': 0.167374, 'text': 'Yoda', 'type': 'Person'},
   {'relevance': 0.158443, 'text': 'Kashyyyk', 'type': 'City'},
   {'relevance': 0.146907, 'text': 'representative', 'type': 'JobTitle'},
   {'relevance': 0.142806, 'text': 'Coruscant', 'type': 'City'},
   {'relevance': 0.127914, 'text': 'Senate', 'type': 'Organization'},
   {'relevance': 0.122259, 'text': 'Mustafar', 'type': 'City'}]},
 'id': 'c506239a-f772-43c7-952b-b25180268750',
 'score': 7.1049113,
 'text': "Palpatine appoints Anakin to the Jedi Council as his representative, but th
{'enriched_text': {'entities': [{'relevance': 0.938404,
    'text': 'Anakin',
    'type': 'Person'},
   {'relevance': 0.832747, 'text': 'Shmi', 'type': 'Person'},
   {'relevance': 0.32508, 'text': 'Cliegg Lars', 'type': 'Person'},
   {'relevance': 0.254378, 'text': 'Jango Fett', 'type': 'Person'},
   {'relevance': 0.226119, 'text': 'Obi-Wan', 'type': 'Person'},
```

```
              {'relevance': 0.20842, 'text': 'Tusken Raiders', 'type': 'Organization'},
              {'relevance': 0.192448, 'text': 'Padmé', 'type': 'Person'},
              {'relevance': 0.172401, 'text': 'Tusken', 'type': 'City'},
              {'relevance': 0.156956, 'text': 'Boba', 'type': 'Person'},
              {'relevance': 0.131101, 'text': 'Geonosis', 'type': 'City'}]},
             'id': '0e46ad2e-bd03-471c-8bf0-ac68232fa4a0',
             'score': 7.1049113,
             'text': "Obi-Wan's investigation leads him to the remote ocean planet Kamino, where
```

```
In [116]: candidates = [fragment['enriched_text']['entities'] for fragment in results]
          candidates
```

```
Out[116]: [[{'relevance': 0.942309, 'text': 'Anakin', 'type': 'Person'},
            {'relevance': 0.669482, 'text': 'Palpatine', 'type': 'Person'},
            {'relevance': 0.464408, 'text': 'Darth Sidious', 'type': 'Person'},
            {'relevance': 0.302401, 'text': 'Jedi Council', 'type': 'Organization'},
            {'relevance': 0.242974, 'text': 'Darth Vader', 'type': 'Person'},
            {'relevance': 0.228208, 'text': 'Jedi Temple', 'type': 'Facility'},
            {'relevance': 0.194849, 'text': 'Mace Windu', 'type': 'Person'},
            {'relevance': 0.177711, 'text': 'Padmé', 'type': 'Person'},
            {'relevance': 0.167374, 'text': 'Yoda', 'type': 'Person'},
            {'relevance': 0.158443, 'text': 'Kashyyyk', 'type': 'City'},
            {'relevance': 0.146907, 'text': 'representative', 'type': 'JobTitle'},
            {'relevance': 0.142806, 'text': 'Coruscant', 'type': 'City'},
            {'relevance': 0.127914, 'text': 'Senate', 'type': 'Organization'},
            {'relevance': 0.122259, 'text': 'Mustafar', 'type': 'City'}],
           [{'relevance': 0.938404, 'text': 'Anakin', 'type': 'Person'},
            {'relevance': 0.832747, 'text': 'Shmi', 'type': 'Person'},
            {'relevance': 0.32508, 'text': 'Cliegg Lars', 'type': 'Person'},
            {'relevance': 0.254378, 'text': 'Jango Fett', 'type': 'Person'},
            {'relevance': 0.226119, 'text': 'Obi-Wan', 'type': 'Person'},
            {'relevance': 0.20842, 'text': 'Tusken Raiders', 'type': 'Organization'},
            {'relevance': 0.192448, 'text': 'Padmé', 'type': 'Person'},
            {'relevance': 0.172401, 'text': 'Tusken', 'type': 'City'},
            {'relevance': 0.156956, 'text': 'Boba', 'type': 'Person'},
            {'relevance': 0.131101, 'text': 'Geonosis', 'type': 'City'}],
           [{'relevance': 0.942309, 'text': 'Anakin', 'type': 'Person'},
            {'relevance': 0.669482, 'text': 'Palpatine', 'type': 'Person'},
            {'relevance': 0.464408, 'text': 'Darth Sidious', 'type': 'Person'},
            {'relevance': 0.302401, 'text': 'Jedi Council', 'type': 'Organization'},
            {'relevance': 0.242974, 'text': 'Darth Vader', 'type': 'Person'},
            {'relevance': 0.228208, 'text': 'Jedi Temple', 'type': 'Facility'},
            {'relevance': 0.194849, 'text': 'Mace Windu', 'type': 'Person'},
            {'relevance': 0.177711, 'text': 'Padmé', 'type': 'Person'},
            {'relevance': 0.167374, 'text': 'Yoda', 'type': 'Person'},
            {'relevance': 0.158443, 'text': 'Kashyyyk', 'type': 'City'},
            {'relevance': 0.146907, 'text': 'representative', 'type': 'JobTitle'},
            {'relevance': 0.142806, 'text': 'Coruscant', 'type': 'City'},
```

```
 {'relevance': 0.127914, 'text': 'Senate', 'type': 'Organization'},
 {'relevance': 0.122259, 'text': 'Mustafar', 'type': 'City'}],
[{'relevance': 0.938404, 'text': 'Anakin', 'type': 'Person'},
 {'relevance': 0.832747, 'text': 'Shmi', 'type': 'Person'},
 {'relevance': 0.32508, 'text': 'Cliegg Lars', 'type': 'Person'},
 {'relevance': 0.254378, 'text': 'Jango Fett', 'type': 'Person'},
 {'relevance': 0.226119, 'text': 'Obi-Wan', 'type': 'Person'},
 {'relevance': 0.20842, 'text': 'Tusken Raiders', 'type': 'Organization'},
 {'relevance': 0.192448, 'text': 'Padmé', 'type': 'Person'},
 {'relevance': 0.172401, 'text': 'Tusken', 'type': 'City'},
 {'relevance': 0.156956, 'text': 'Boba', 'type': 'Person'},
 {'relevance': 0.131101, 'text': 'Geonosis', 'type': 'City'}],
[{'relevance': 0.942309, 'text': 'Anakin', 'type': 'Person'},
 {'relevance': 0.669482, 'text': 'Palpatine', 'type': 'Person'},
 {'relevance': 0.464408, 'text': 'Darth Sidious', 'type': 'Person'},
 {'relevance': 0.302401, 'text': 'Jedi Council', 'type': 'Organization'},
 {'relevance': 0.242974, 'text': 'Darth Vader', 'type': 'Person'},
 {'relevance': 0.228208, 'text': 'Jedi Temple', 'type': 'Facility'},
 {'relevance': 0.194849, 'text': 'Mace Windu', 'type': 'Person'},
 {'relevance': 0.177711, 'text': 'Padmé', 'type': 'Person'},
 {'relevance': 0.167374, 'text': 'Yoda', 'type': 'Person'},
 {'relevance': 0.158443, 'text': 'Kashyyyk', 'type': 'City'},
 {'relevance': 0.146907, 'text': 'representative', 'type': 'JobTitle'},
 {'relevance': 0.142806, 'text': 'Coruscant', 'type': 'City'},
 {'relevance': 0.127914, 'text': 'Senate', 'type': 'Organization'},
 {'relevance': 0.122259, 'text': 'Mustafar', 'type': 'City'}],
[{'relevance': 0.938404, 'text': 'Anakin', 'type': 'Person'},
 {'relevance': 0.832747, 'text': 'Shmi', 'type': 'Person'},
 {'relevance': 0.32508, 'text': 'Cliegg Lars', 'type': 'Person'},
 {'relevance': 0.254378, 'text': 'Jango Fett', 'type': 'Person'},
 {'relevance': 0.226119, 'text': 'Obi-Wan', 'type': 'Person'},
 {'relevance': 0.20842, 'text': 'Tusken Raiders', 'type': 'Organization'},
 {'relevance': 0.192448, 'text': 'Padmé', 'type': 'Person'},
 {'relevance': 0.172401, 'text': 'Tusken', 'type': 'City'},
 {'relevance': 0.156956, 'text': 'Boba', 'type': 'Person'},
 {'relevance': 0.131101, 'text': 'Geonosis', 'type': 'City'}],
[{'relevance': 0.942309, 'text': 'Anakin', 'type': 'Person'},
 {'relevance': 0.669482, 'text': 'Palpatine', 'type': 'Person'},
 {'relevance': 0.464408, 'text': 'Darth Sidious', 'type': 'Person'},
 {'relevance': 0.302401, 'text': 'Jedi Council', 'type': 'Organization'},
 {'relevance': 0.242974, 'text': 'Darth Vader', 'type': 'Person'},
 {'relevance': 0.228208, 'text': 'Jedi Temple', 'type': 'Facility'},
 {'relevance': 0.194849, 'text': 'Mace Windu', 'type': 'Person'},
 {'relevance': 0.177711, 'text': 'Padmé', 'type': 'Person'},
 {'relevance': 0.167374, 'text': 'Yoda', 'type': 'Person'},
 {'relevance': 0.158443, 'text': 'Kashyyyk', 'type': 'City'},
 {'relevance': 0.146907, 'text': 'representative', 'type': 'JobTitle'},
 {'relevance': 0.142806, 'text': 'Coruscant', 'type': 'City'},
```

```
 {'relevance': 0.127914, 'text': 'Senate', 'type': 'Organization'},
 {'relevance': 0.122259, 'text': 'Mustafar', 'type': 'City'}],
[{'relevance': 0.938404, 'text': 'Anakin', 'type': 'Person'},
 {'relevance': 0.832747, 'text': 'Shmi', 'type': 'Person'},
 {'relevance': 0.32508, 'text': 'Cliegg Lars', 'type': 'Person'},
 {'relevance': 0.254378, 'text': 'Jango Fett', 'type': 'Person'},
 {'relevance': 0.226119, 'text': 'Obi-Wan', 'type': 'Person'},
 {'relevance': 0.20842, 'text': 'Tusken Raiders', 'type': 'Organization'},
 {'relevance': 0.192448, 'text': 'Padmé', 'type': 'Person'},
 {'relevance': 0.172401, 'text': 'Tusken', 'type': 'City'},
 {'relevance': 0.156956, 'text': 'Boba', 'type': 'Person'},
 {'relevance': 0.131101, 'text': 'Geonosis', 'type': 'City'}]]
```

```
In [118]: [[{'relevance': candidate['relevance'], 'text': candidate['text']} for candidate in ca
```

```
Out[118]: [[{'relevance': 0.942309, 'text': 'Anakin'},
             {'relevance': 0.669482, 'text': 'Palpatine'},
             {'relevance': 0.464408, 'text': 'Darth Sidious'},
             {'relevance': 0.242974, 'text': 'Darth Vader'},
             {'relevance': 0.194849, 'text': 'Mace Windu'},
             {'relevance': 0.177711, 'text': 'Padmé'},
             {'relevance': 0.167374, 'text': 'Yoda'}],
            [{'relevance': 0.938404, 'text': 'Anakin'},
             {'relevance': 0.832747, 'text': 'Shmi'},
             {'relevance': 0.32508, 'text': 'Cliegg Lars'},
             {'relevance': 0.254378, 'text': 'Jango Fett'},
             {'relevance': 0.226119, 'text': 'Obi-Wan'},
             {'relevance': 0.192448, 'text': 'Padmé'},
             {'relevance': 0.156956, 'text': 'Boba'}],
            [{'relevance': 0.942309, 'text': 'Anakin'},
             {'relevance': 0.669482, 'text': 'Palpatine'},
             {'relevance': 0.464408, 'text': 'Darth Sidious'},
             {'relevance': 0.242974, 'text': 'Darth Vader'},
             {'relevance': 0.194849, 'text': 'Mace Windu'},
             {'relevance': 0.177711, 'text': 'Padmé'},
             {'relevance': 0.167374, 'text': 'Yoda'}],
            [{'relevance': 0.938404, 'text': 'Anakin'},
             {'relevance': 0.832747, 'text': 'Shmi'},
             {'relevance': 0.32508, 'text': 'Cliegg Lars'},
             {'relevance': 0.254378, 'text': 'Jango Fett'},
             {'relevance': 0.226119, 'text': 'Obi-Wan'},
             {'relevance': 0.192448, 'text': 'Padmé'},
             {'relevance': 0.156956, 'text': 'Boba'}],
            [{'relevance': 0.942309, 'text': 'Anakin'},
             {'relevance': 0.669482, 'text': 'Palpatine'},
             {'relevance': 0.464408, 'text': 'Darth Sidious'},
             {'relevance': 0.242974, 'text': 'Darth Vader'},
             {'relevance': 0.194849, 'text': 'Mace Windu'},
```

```
 {'relevance': 0.177711, 'text': 'Padmé'},
 {'relevance': 0.167374, 'text': 'Yoda'}],
[{'relevance': 0.938404, 'text': 'Anakin'},
 {'relevance': 0.832747, 'text': 'Shmi'},
 {'relevance': 0.32508, 'text': 'Cliegg Lars'},
 {'relevance': 0.254378, 'text': 'Jango Fett'},
 {'relevance': 0.226119, 'text': 'Obi-Wan'},
 {'relevance': 0.192448, 'text': 'Padmé'},
 {'relevance': 0.156956, 'text': 'Boba'}],
[{'relevance': 0.942309, 'text': 'Anakin'},
 {'relevance': 0.669482, 'text': 'Palpatine'},
 {'relevance': 0.464408, 'text': 'Darth Sidious'},
 {'relevance': 0.242974, 'text': 'Darth Vader'},
 {'relevance': 0.194849, 'text': 'Mace Windu'},
 {'relevance': 0.177711, 'text': 'Padmé'},
 {'relevance': 0.167374, 'text': 'Yoda'}],
[{'relevance': 0.938404, 'text': 'Anakin'},
 {'relevance': 0.832747, 'text': 'Shmi'},
 {'relevance': 0.32508, 'text': 'Cliegg Lars'},
 {'relevance': 0.254378, 'text': 'Jango Fett'},
 {'relevance': 0.226119, 'text': 'Obi-Wan'},
 {'relevance': 0.192448, 'text': 'Padmé'},
 {'relevance': 0.156956, 'text': 'Boba'}]]
```

In [124]:
```
# Make a list of the top candidates, excluding the entity in the question, which is th
candidate_list = [candidate[1]['text'] for candidate in candidates]
candidate_list
```

Out[124]:
```
['Palpatine',
 'Shmi',
 'Palpatine',
 'Shmi',
 'Palpatine',
 'Shmi',
 'Palpatine',
 'Shmi']
```

In [123]:
```
# Count the top candidates.
from collections import Counter
Counter(candidate_list)
```

Out[123]:
```
Counter({'Palpatine': 4, 'Shmi': 4})
```

In [125]:
```
# At this point, we could, for instance, choose the candidate with most occurrences. B
print("The answer is: {}".format(candidate_list[0]))
```

```
The answer is: Palpatine
```

1.8 5. Reflections

Q: Now that you have gone through this exercise of designing a system that uses two IBM Watson services, what did you learn? What were some of the strengths and weaknesses of this approach?

A: I'll start with the weaknesses. I was very disappointed with the performance of both services (Conversation and Discovery). It is by mere chance that, after endless trials, I finally got a query whose actual answer happens to be at the top of the list. With the system set as instructed in this lesson there is no way any query can get a real answer. The analysis of the question is very coarse: the information extracted from the question is simply insufficient to discriminate between questions that have absolutely nothing in common. The Conversation service parses "Who is Anakin's mother/father/mentor/pet/friend/...?// Who is Anakin?" and other infinitely many queries as having intent(s): ['who'], entities: ['Anakin'], and dialogue node: Agent. Adding further entities to Conversation (such as @FamilyMember) is pointless without them being added to Discovery as well, because those would be found in the question but not in the document collection where the answers come from. Moreover, a query does not seem to be able to express the *simultaneous* presence of multiple entities in the candidate text, or to filter by relevance on *all* the entities. As a consequence, the user is limited to querying on one entity at a time. Once we get some results, checking if the actual answer is in there is another thing altogether. This is totally unfeasible programmatically. I had to do it by inspecting the results and cherry-picking by hand the one to return. There is no learning involved, at least as far as Conversation is concerned, and all the "knowledge" must be hand-coded. This is a 100% rule-based system. Where is the AI in all this?

On the bright side, it is awesome to be able to take advantage of the wealth of resources that Discovery is connected to. Furthermore, setting up a system like this is very fast: in half an hour one may have a basic system up and running.

1.9 (Optional) Extensions

We have provided a set of sample data files containing Star Wars plot summaries. But as mentioned before, you are free to use your own dataset. In fact, a larger dataset maybe more suitable for use with IBM Watson's NLP services. If you used your own dataset, answer the following questions.

Q: What dataset did you use, and in what ways is it different from the sample files provided?

A: NA

Q: Either include your dataset in the .zip file or repository you submit, or provide clear instructions on how to obtain the dataset, so that your reviewer can run your notebook or inspect the data to verify your results.

A: NA

You can also design a web-based application that utilizes these services and deploy that on Bluemix! If you do, please share with your instructors and peers.

```
In [ ]:
```