Analyze text with Azure Language

1. Introduction

https://learn.microsoft.com/en-us/training/modules/analyze-text-ai-language/1-introduction

Introduction

Completed

- 1 minute

Every day, the world generates a vast quantity of data; much of it text-based in the form of emails, social media posts, online reviews, business documents, and more. Artificial intelligence techniques that apply statistical and semantic models enable you to create applications that extract meaning and insights from this text-based data.

The Azure Language provides an API for common text analysis techniques that you can easily integrate into your own application code.

In this module, you will learn how to use Azure Language to:

- Detect language from text.
- Analyze text sentiment.
- Extract key phrases, entities, and linked entities.

2. Provision an Azure Language resource

https://learn.microsoft.com/en-us/training/modules/analyze-text-ai-language/2-provision-resource

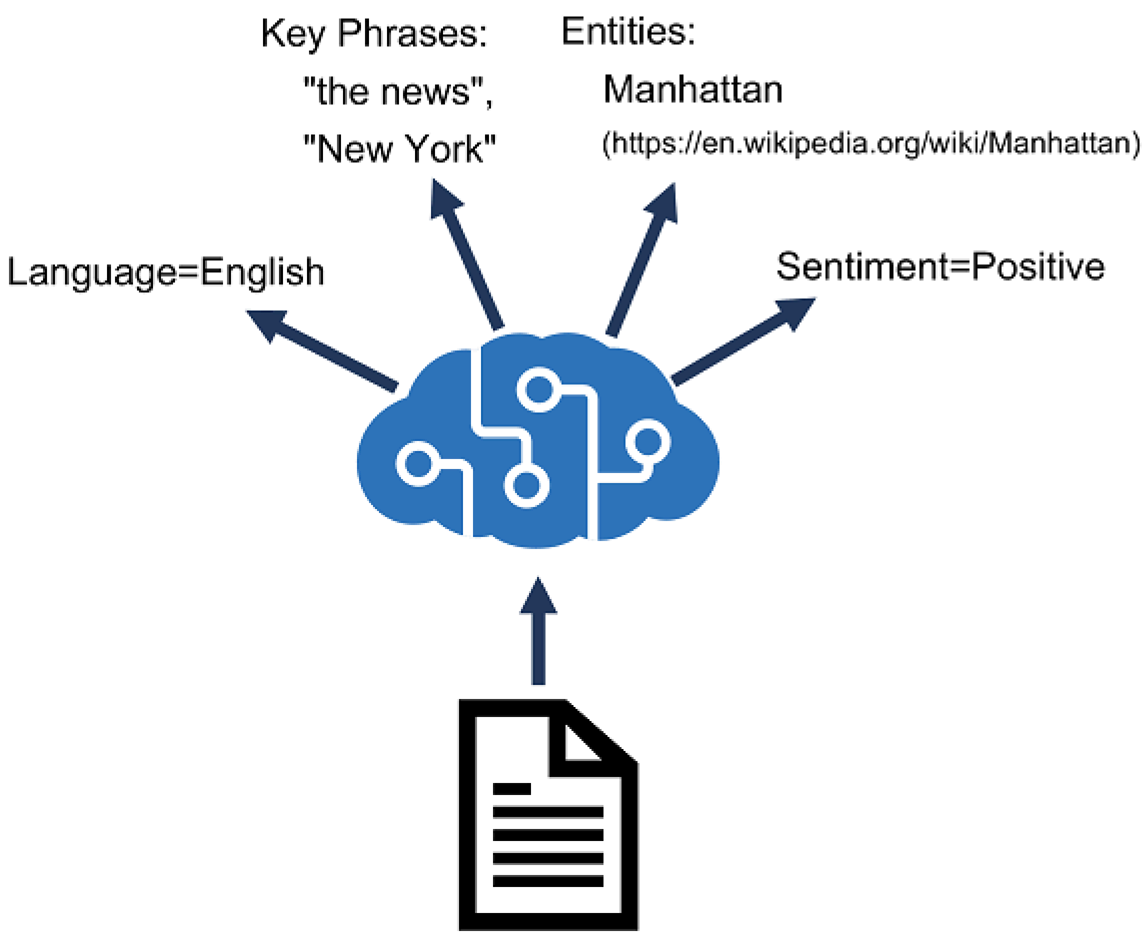
Provision an Azure Language resource

Completed

- 3 minutes

Azure Language is designed to help you extract information from text. It provides functionality that you can use for tasks like:

- *Language detection* - determining the language in which text is written.
- *Key phrase extraction* - identifying important words and phrases in the text that indicate the main points.
- *Sentiment analysis* - quantifying how positive or negative the text is.
- *Named entity recognition* - detecting references to entities, including people, locations, time periods, organizations, and more.
- *Entity linking* - identifying specific entities by providing reference links to Wikipedia articles.



Azure resources for text analysis

To use Azure Language to analyze text, you must provision a resource for it in your Azure subscription. You can provision a resource through Microsoft Foundry portal

After you have provisioned a suitable resource in your Azure subscription, you can use its **endpoint** and one of its **keys** to call the Azure Language APIs from your code. You can call the Azure Language APIs by submitting requests in JSON format to the REST interface, or by using any of the available programming language-specific SDKs.

Note

The code examples in the subsequent units in this module show the JSON requests and responses exchanged with the REST interface. When using an SDK, the JSON requests are abstracted by appropriate objects and methods that encapsulate the same data values. You'll get a chance to try the SDK for C# or Python for yourself in the exercise later in the module.

3. Detect language

https://learn.microsoft.com/en-us/training/modules/analyze-text-ai-language/3-detect-language

Detect language

Completed

- 5 minutes

The Azure Language detection API evaluates text input and, for each document submitted, returns language identifiers with a score indicating the strength of the analysis.

This capability is useful for content stores that collect arbitrary text, where language is unknown. Another scenario could involve a chat bot. If a user starts a session with the chat bot, language detection can be used to determine which language they are using and allow you to configure your bot responses in the appropriate language.

You can parse the results of this analysis to determine which language is used in the input document. The response also returns a score, which reflects the confidence of the model (a value between 0 and 1).

Language detection can work with documents or single phrases. It's important to note that the document size must be under 5,120 characters. The size limit is per document and each collection is restricted to 1,000 items (IDs). A sample of a properly formatted JSON payload that you might submit to the service in the request body is shown here, including a collection of **documents**, each containing a unique **id** and the **text** to be analyzed. Optionally, you can provide a **countryHint** to improve prediction performance.

```
{
    "kind": "LanguageDetection",
    "parameters": {
        "modelVersion": "latest"
    },
    "analysisInput":{
        "documents":[
              {
                "id": "1",
                "text": "Hello world",
                "countryHint": "US"
```

```
            },
            {
              "id": "2",
              "text": "Bonjour tout le monde"
            }
        ]
    }
}
```

The service will return a JSON response that contains a result for each **document** in the request body, including the predicted language and a value indicating the confidence level of the prediction. The confidence level is a value ranging from 0 to 1 with values closer to 1 being a higher confidence level. Here's an example of a standard JSON response that maps to the above request JSON.

```
{   "kind": "LanguageDetectionResults",
    "results": {
        "documents": [
          {
            "detectedLanguage": {
              "confidenceScore": 1,
              "iso6391Name": "en",
              "name": "English"
            },
            "id": "1",
            "warnings": []
          },
          {
            "detectedLanguage": {
              "confidenceScore": 1,
              "iso6391Name": "fr",
              "name": "French"
            },
            "id": "2",
            "warnings": []
          }
        ],
        "errors": [],
        "modelVersion": "2022-10-01"
    }
}
```

In our sample, all of the languages show a confidence of 1, mostly because the text is relatively simple and easy to identify the language for.

If you pass in a document that has multilingual content, the service will behave a bit differently. Mixed language content within the same document returns the language with the largest representation in the content, but with a lower positive rating, reflecting the marginal strength of that assessment. In the following example, the input is a blend of English, Spanish, and French. The analyzer uses statistical analysis of the text to determine the *predominant* language.

```
{
  "documents": [
    {
      "id": "1",
      "text": "Hello, I would like to take a class at your University. ¿Se ofrecen clases en españ
    }
  ]
}
```

The following sample shows a response for this multi-language example.

```
{
    "documents": [
        {
            "id": "1",
            "detectedLanguage": {
                "name": "Spanish",
                "iso6391Name": "es",
                "confidenceScore": 0.9375
            },
            "warnings": []
        }
    ],
    "errors": [],
    "modelVersion": "2022-10-01"
}
```

The last condition to consider is when there is ambiguity as to the language content. The scenario might happen if you submit textual content that the analyzer is not able to parse, for example because of character encoding issues when converting the text to a string variable. As a result, the response for the language name and ISO code will indicate (unknown) and the score value will be returned as `0`. The following example shows how the response would look.

```
{
    "documents": [
        {
            "id": "1",
            "detectedLanguage": {
                "name": "(Unknown)",
                "iso6391Name": "(Unknown)",
                "confidenceScore": 0.0
            },
            "warnings": []
        }
    ],
    "errors": [],
    "modelVersion": "2022-10-01"
}
```

4. Extract key phrases

https://learn.microsoft.com/en-us/training/modules/analyze-text-ai-language/4-extract-key-phrases

Extract key phrases

Completed

- 2 minutes

Key phrase extraction is the process of evaluating the text of a document, or documents, and then identifying the main points around the context of the document(s).

Key phrase extraction works best for larger documents (the maximum size that can be analyzed is 5,120 characters).

As with language detection, the REST interface enables you to submit one or more documents for analysis.

```
{
    "kind": "KeyPhraseExtraction",
    "parameters": {
        "modelVersion": "latest"
    },
    "analysisInput":{
        "documents":[
            {
              "id": "1",
              "language": "en",
              "text": "You must be the change you wish
                       to see in the world."
            },
            {
              "id": "2",
              "language": "en",
                "text": "The journey of a thousand miles
                         begins with a single step."
            }
        ]
    }
}
```

The response contains a list of key phrases detected in each document:

```
{
    "kind": "KeyPhraseExtractionResults",
    "results": {
    "documents": [
        {
         "id": "1",
         "keyPhrases": [
           "change",
           "world"
         ],
         "warnings": []
       },
       {
         "id": "2",
         "keyPhrases": [
           "miles",
           "single step",
           "journey"
         ],
         "warnings": []
       }
],
    "errors": [],
    "modelVersion": "2021-06-01"
    }
}
```

5. Analyze sentiment

https://learn.microsoft.com/en-us/training/modules/analyze-text-ai-language/5-analyze-sentiment

Analyze sentiment

Completed

- 3 minutes

Sentiment analysis is used to evaluate how positive or negative a text document is, which can be useful in various workloads, such as:

- Evaluating a movie, book, or product by quantifying sentiment based on reviews.
- Prioritizing customer service responses to correspondence received through email or social media messaging.

When using Azure Language to evaluate sentiment, the response includes overall document sentiment and individual sentence sentiment for each document submitted to the service.

For example, you could submit a single document for sentiment analysis like this:

```
{
  "kind": "SentimentAnalysis",
  "parameters": {
    "modelVersion": "latest"
  },
  "analysisInput": {
    "documents": [
      {
        "id": "1",
        "language": "en",
        "text": "Good morning!"
      }
    ]
  }
}
```

The response from the service might look like this:

```
{
  "kind": "SentimentAnalysisResults",
  "results": {
    "documents": [
      {
        "id": "1",
        "sentiment": "positive",
        "confidenceScores": {
          "positive": 0.89,
          "neutral": 0.1,
          "negative": 0.01
        },
        "sentences": [
          {
            "sentiment": "positive",
            "confidenceScores": {
              "positive": 0.89,
              "neutral": 0.1,
              "negative": 0.01
            },
            "offset": 0,
            "length": 13,
            "text": "Good morning!"
          }
        ],
        "warnings": []
      }
    ],
    "errors": [],
    "modelVersion": "2022-11-01"
  }
}
```

Sentence sentiment is based on confidence scores for **positive**, **negative**, and **neutral** classification values between 0 and 1.

Overall document sentiment is based on sentences:

- If all sentences are neutral, the overall sentiment is neutral.
- If sentence classifications include only positive and neutral, the overall sentiment is positive.
- If the sentence classifications include only negative and neutral, the overall sentiment is negative.
- If the sentence classifications include positive and negative, the overall sentiment is mixed.

6. Extract entities

https://learn.microsoft.com/en-us/training/modules/analyze-text-ai-language/6-extract-entities

Extract entities

Completed

- 3 minutes

Named Entity Recognition identifies entities that are mentioned in the text. Entities are grouped into categories and subcategories, for example:

- Person
- Location
- DateTime
- Organization
- Address
- Email
- URL

Note

For a full list of categories, see the documentation.

Input for entity recognition is similar to input for other Azure Language API functions:

```
{
  "kind": "EntityRecognition",
  "parameters": {
    "modelVersion": "latest"
  },
  "analysisInput": {
    "documents": [
      {
        "id": "1",
        "language": "en",
        "text": "Joe went to London on Saturday"
      }
    ]
  }
}
```

The response includes a list of categorized entities found in each document:

```
{
    "kind": "EntityRecognitionResults",
     "results": {
          "documents":[
              {
                  "entities":[
                  {
                    "text":"Joe",
                    "category":"Person",
                    "offset":0,
                    "length":3,
                    "confidenceScore":0.62
                  },
                  {
                    "text":"London",
                    "category":"Location",
                    "subcategory":"GPE",
                    "offset":12,
                    "length":6,
                    "confidenceScore":0.88
                  },
                  {
                    "text":"Saturday",
                    "category":"DateTime",
                    "subcategory":"Date",
                    "offset":22,
                    "length":8,
                    "confidenceScore":0.8
                  }
                ],
                "id":"1",
                "warnings":[]
              }
        ],
        "errors":[],
        "modelVersion":"2021-01-15"
    }
}
```

To learn more about entities see the Build a conversational language understanding model module.

7. Extract linked entities

https://learn.microsoft.com/en-us/training/modules/analyze-text-ai-language/7-extract-linked-entities

Extract linked entities

Completed

- 3 minutes

In some cases, the same name might be applicable to more than one entity. For example, does an instance of the word "Venus" refer to the planet or the goddess from mythology?

Entity linking can be used to disambiguate entities of the same name by referencing an article in a knowledge base. Wikipedia provides the knowledge base for the Text Analytics service. Specific article links are determined based on entity context within the text.

For example, "I saw Venus shining in the sky" is associated with the link https://en.wikipedia.org/wiki/Venus; while "Venus, the goddess of beauty" is associated with https://en.wikipedia.org/wiki/Venus_(mythology).

As with all Azure Language service functions, you can submit one or more documents for analysis:

```
{
  "kind": "EntityLinking",
  "parameters": {
    "modelVersion": "latest"
  },
  "analysisInput": {
    "documents": [
      {
        "id": "1",
        "language": "en",
        "text": "I saw Venus shining in the sky"
      }
    ]
  }
}
```

The response includes the entities identified in the text along with links to associated articles:

```
 {
  "kind": "EntityLinkingResults",
  "results": {
    "documents": [
      {
        "id": "1",
        "entities": [
          {
            "bingId": "89253af3-5b63-e620-9227-f839138139f6",
            "name": "Venus",
            "matches": [
              {
                "text": "Venus",
                "offset": 6,
                "length": 5,
                "confidenceScore": 0.01
              }
            ],
            "language": "en",
            "id": "Venus",
            "url": "https://en.wikipedia.org/wiki/Venus",
            "dataSource": "Wikipedia"
          }
        ],
        "warnings": []
      }
    ],
    "errors": [],
    "modelVersion": "2021-06-01"
  }
}
```

8. Exercise - Analyze text

https://learn.microsoft.com/en-us/training/modules/analyze-text-ai-language/8-exercise-analyze-text

Exercise - Analyze text

Completed

- 30 minutes

In this exercise, you use the Azure Language SDK to develop a client application that analyzes text.

Note

To complete this lab, you need an **Azure subscription** in which you have administrative access.

Launch the exercise and follow the instructions.

Launch Exercise

Tip

After completing the exercise, if you've finished exploring Foundry Tools, delete the Azure resources that you created during the exercise.

9. Module assessment

https://learn.microsoft.com/en-us/training/modules/analyze-text-ai-language/9-knowledge-check

Module assessment

Completed

- 3 minutes

10. Summary

https://learn.microsoft.com/en-us/training/modules/analyze-text-ai-language/10-summary

Summary

Completed

- 1 minute

In this module, you learned how to use Azure Language to:

- Detect language from text.
- Analyze text sentiment.
- Extract key phrases, entities, and linked entities.

To learn more about Azure Language and some of the concepts covered in this module, you can explore the following documentation pages:

- Azure Language documentation
- Build a conversational language understanding model
- Create a custom named entity extraction solution