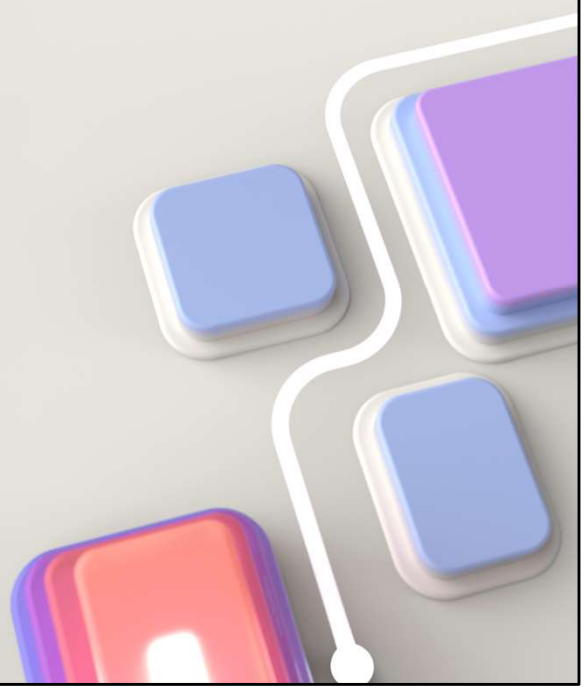
Microsoft

Develop natural language processing solutions

Agenda

- Analyzing and translating text
- Build a question answering solution
- Build a conversational language understanding app
- Custom classification and named entity extraction
- Speech recognition, synthesis, and translation

Analyzing text



Learning Objectives

After completing this module, you will be able to:

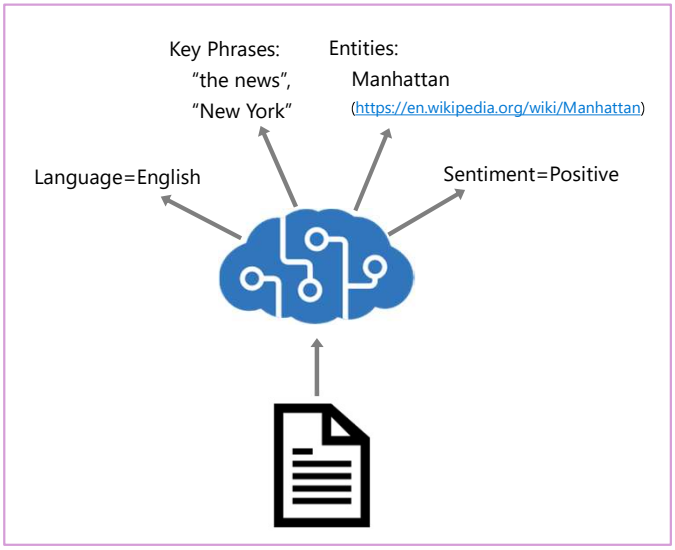
1. Detect language and extract key phrases
2. Analyze sentiment and detect PII
3. Summarize text
4. Extract entities and linked entities
5. Translate text

The Azure AI Language Service

Preconfigured features:

- Language detection
- Key phrase extraction
- Sentiment analysis
- Named entity recognition
- Entity linking
- Summarization
- PII detection

Customizable features are covered in another section



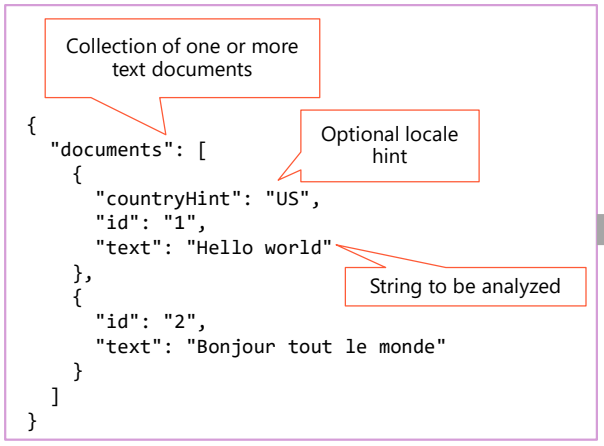
Här är en kort förklaring av varje **förkonfigurerad funktion** i **Azure AI Language Service**:

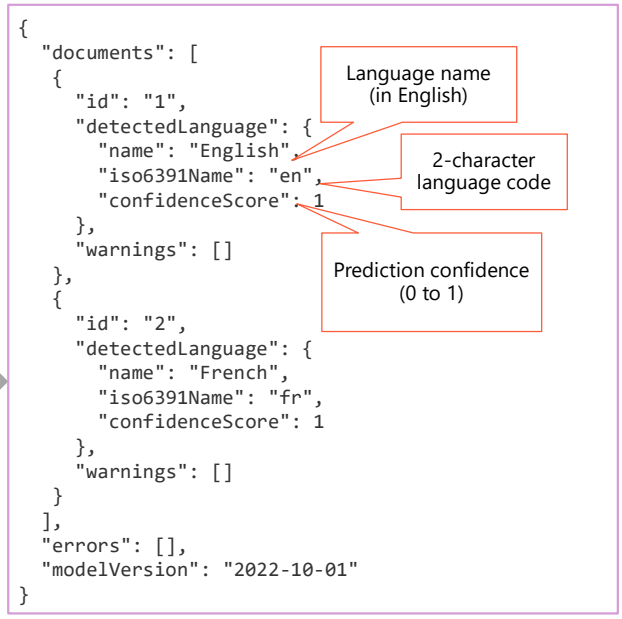
- **Language detection** – Identifierar vilket språk en text är skriven på.
- **Key phrase extraction** – Extraherar viktiga ord och fraser från en text.
- **Sentiment analysis** – Bestämmer om tonen i en text är positiv, negativ eller neutral.
- **Named entity recognition (NER)** – Identifierar namn på personer, platser och organisationer i texten.
- **Entity linking** – Kopplar identifierade entiteter till externa källor, t.ex. Wikipedia.
- **Summarization** – Skapar en kortfattad sammanfattning av en längre text.
- **PII detection** – Upptäcker och döljer personligt identifierbar information (t.ex. namn, telefonnummer).

Add note that further modules for the customizable features are covered later in this section. Also, some of the prebuilt can be customized – info on that can be found on the docs pages

Language detection

- Determine the language in which text is written
- Often useful as a pre-cursor to further analysis that requires a known language





Key phrase extraction

- Identify the main "talking points" of the text
- Works best with larger documents (up to 5,120 characters)

```
{
  "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."
    }
  ]
}
```

Language (defaults to English if not present)

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

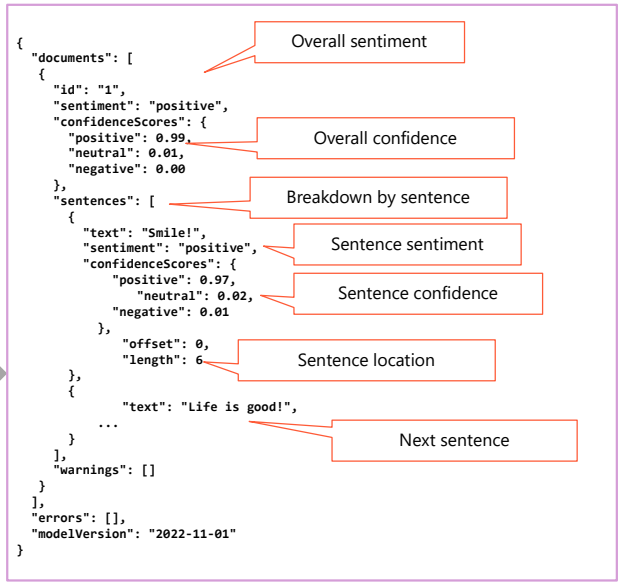
List of key phrases in document 1

List of key phrases in document 2

Sentiment analysis

- Scores overall document sentiment and individual sentence sentiment
- Sentence sentiment is based on confidence scores for *positive*, *negative*, and *neutral*
- Overall document sentiment is based on sentences:
  - All sentences are *neutral* = ***neutral***
  - Sentences include *positive* and *neutral* = ***positive***
  - Sentences include *negative* and *neutral* = ***negative***
  - Sentences include *positive* and *negative* = ***mixed***

```
{
  "documents": [
    {
      "language": "en",
      "id": "1",
      "text": "Smile! Life is good!"
    }
  ]
}
```



V 3.1 (currently in preview) also includes *opinion mining* – for example "The food was great!" would be evaluated as a *positive* opinion about the *food*.

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
  - Others...

```
{
  "documents": [
    {
      "language": "en",
      "id": "1",
      "text": "Joe went to London on Saturday"
    }
  ]
}
```

```
{
  "documents":[
      {
        "id":"1",
        "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
        }
        ],
        "warnings":[]
      }
  ],
  "errors":[],
  "modelVersion":"2021-01-15"
}
```

Person entity

Location entity

DateTime entity

Entity Linking

- Used to disambiguate entities of the same name
  - For example, is "Venus" a planet or a goddess?
- Wikipedia provides the knowledge base
- Specific article links are determined based on entity context within the text

"I saw Venus shining in the sky":

https://en.wikipedia.org/wiki/Venus

"Venus, the goddess of beauty":

https://en.wikipedia.org/wiki/Venus_(mythology)

```
{
  "documents": [
    {
      "language": "en",
      "id": "1",
      "text": "I saw Venus shining in the sky"
    }
  ]
}
```

```
{
  "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"
}
```

Named entity

Wikipedia unique article ID

Article link

Summarization

- Can provide two different types of summarization
  - Extractive summarization: Produces summary by using most important sentences
  - Abstractive summarization: Produces a summary capturing the main idea, but not necessarily using the same words as the source document
- Can be customized by training on your own data

```
{
  “analysisInput”: {
    "documents": [{
        "language": "en",
        "id": "1",
        "text": “<long paragraph about Microsoft and
technology>"
     }]
    },
    “tasks”: [{
      “kind”: "ExtractiveSummarization",
      “taskName”: “docExtSummary1”,
      “parameters”: {
        “sentenceCount”: 2
      }
    }
  ]
}
```

```
{
  “documents”:
    [
      {
        “id”:”1”,
        “sentences”:[
          {
            “text”: “<first sentence best summarizing document>”
            “rankScore”: 0.71
            “offset”: 0
            “length”: 135
          },
          {
            “text”: “<first sentence best summarizing document>”
            “rankScore”:“0.67”,
            “offset”: 721
            “length”: 203
          },
        ],
        “warnings”:[]
      }
    ],
  “errors”:[],
  “modelVersion”:“latest"
}
```

Array of sentences specified

Sentence rank score

Bilden beskriver **sammanfattning (summarization) i Azure AI Language Service**.

- **Två typer av sammanfattning:**
  - **Extractive summarization** – Väljer de viktigaste meningarna direkt från texten.
  - **Abstractive summarization** – Genererar en ny sammanfattning med omformulerade meningar.
- **Exempel på API-begäran och svar:**
  - En text analyseras, och de mest relevanta meningarna extraheras.
  - Varje mening får en **rankScore** som indikerar dess relevans.
- **Anpassningsmöjlighet:**
  - Kan tränas på egen data för bättre resultat.

Personally Identifiable Information detection

- Used to detect and remove sensitive information
- Entity categories include Person, PhoneNumber, Email, Address, Credit card, and financial account identification
- Can be used in situations like applying sensitivity labels, removing information to reduce bias, and clean data for data science

```
{
  "documents":[
    {
      "id":"1",
      "language": "en",
      "text": "Call our office at 312-555-1234, or send an email
to support@contoso.com"
    }
  ]
}
```

```
{
  "documents":
    [
      {
        "redactedText": "Call our office at ************, or send
           an email to ******************",
        "id": "1",
        "entities": [{
          "text": "312-555-1234",
          "category": "PhoneNumber",
          "offset": 19,
          "length": 12,
          "confidenceScore": 0.8
        }, {
          "text": "support@contoso.com"
          "category": "Email",
          "offset": 53,
          "length": 19,
          "confidenceScore": 0.8
        }],
        "warnings": []
      }
    ],
  "errors":[],
  "modelVersion":"2021-06-01"
}
```

Text with PII removed

All entities detected

Type of PII detected

Confidence score

Exercise – Analyze Text

Detect Language

Evaluate Sentiment

Identify Key Phrases

Extract Entities

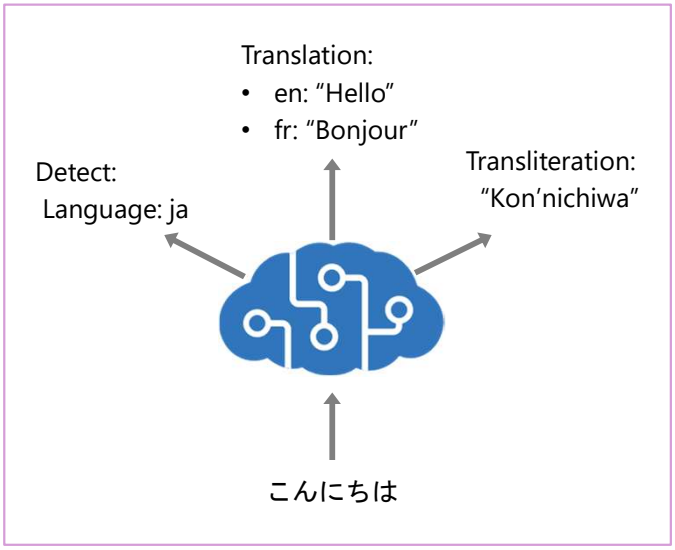
Extract Linked Entities

Translating Text

The Translator Service

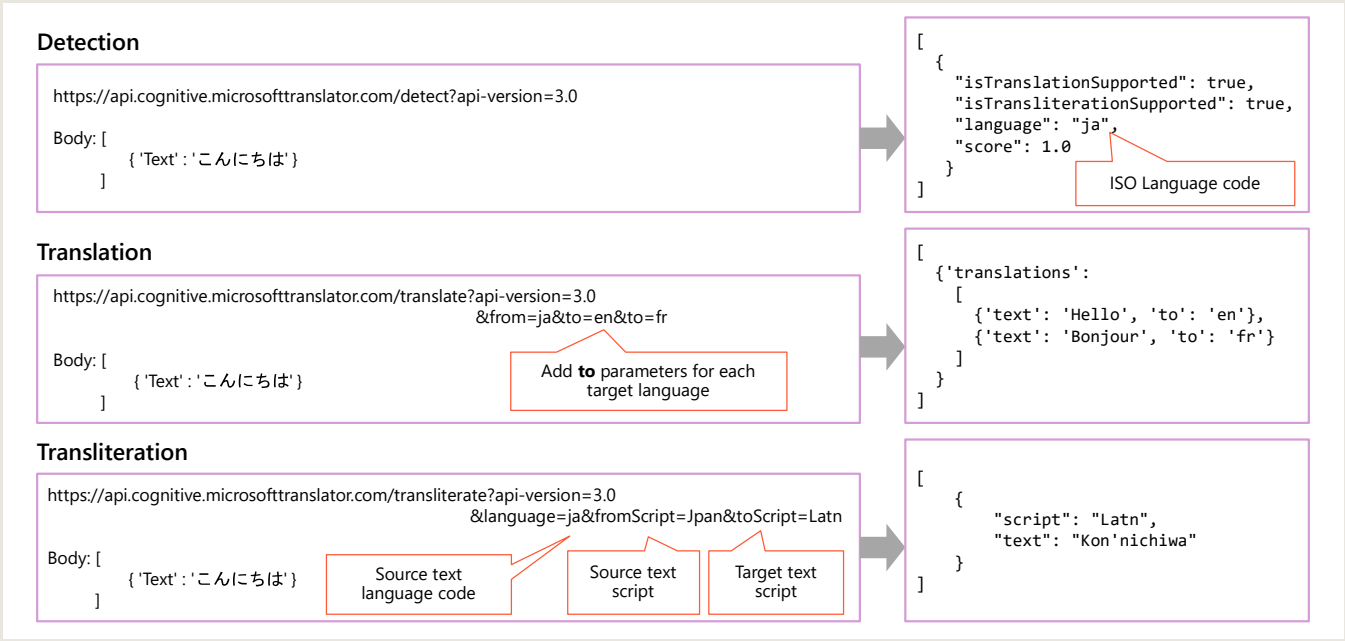
Multilingual text translation REST API

- Language *detection*
- One-to-many *translation*
- Script *transliteration*



Note that the SDKs are in preview at this time – the REST API is available publicly.

Detection, Translation, and Transliteration



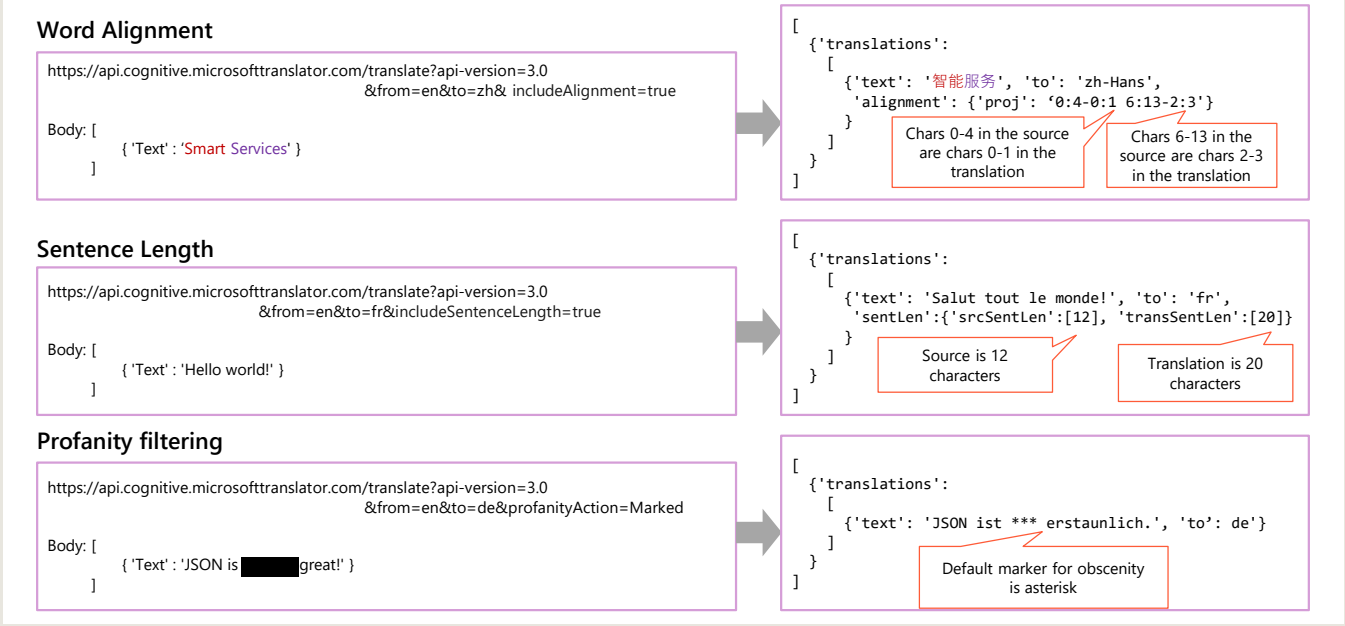
Bilden visar **Detection, Translation och Transliteration** med Microsoft Translator API.

1. **Detection** – Identifierar språk från en text (こんにちは → "ja" för japanska).
2. **Translation** – Översätter texten till andra språk (こんにちは → "Hello" på engelska, "Bonjour" på franska).
3. **Transliteration** – Konverterar textens skriftsystem till ett annat (こんにちは → "Kon'nichiwa" i latinskt alfabet).

- **API-anropen** använder **query-parametrar** (from, to, script) för att styra utdata.
- **Resultaten** returneras som JSON-svar.

You can find the full set of supported languages at https://api.cognitive.microsofttranslator.com/languages?api-version=3.0

Translation Options



Bilden visar **Translation Options** i Microsoft Translator API, med tre inställningar:

1.Word Alignment – Spårar hur ord i källtexten matchas mot översättningen. Exempel: "Smart Services" till **智能服务**, där teckenpositionerna anges i alignment.

2.Sentence Length – Returnerar längden på både käll- och översättningstext. Exempel: "Hello world!" blir "Salut tout le monde!", där källtexten har 12 tecken och översättningen 20.

3.Profanity Filtering – Maskerar svordomar med ***. Exempel: "JSON is [svordom] great!" blir "JSON ist *** erstaunlich." på tyska.

- **Parametrar** (includeAlignment, includeSentenceLength, profanityAction) används för att styra API:ts beteende.

https://learn.microsoft.com/azure/ai-services/translator/reference/v3-0-translate#request-parameters
https://learn.microsoft.com/azure/ai-services/translator/word-alignment
https://learn.microsoft.com/azure/ai-services/translator/profanity-filtering

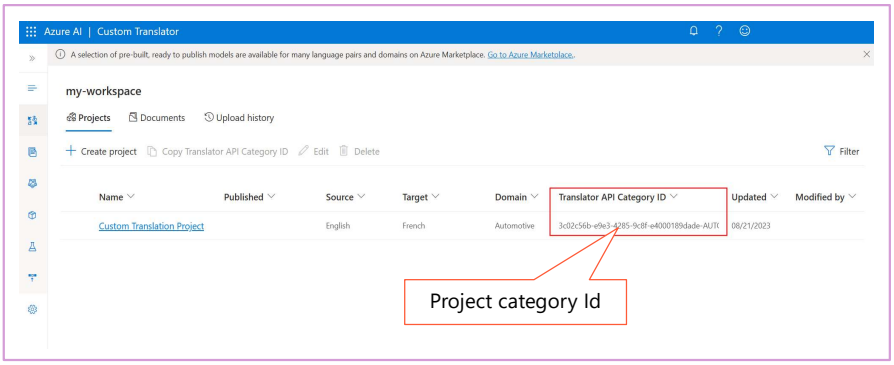
Custom Translation

Create a custom translation model

1. Use the Custom Translator portal
2. Link a workspace to your Azure AI Translator resource
3. Create a project
4. Upload training data files
5. Train a model

Call your model through the Translator API

- Specify a **category** parameter with the project category Id



Bilden beskriver **Custom Translation** i Azure AI Translator.

Processen för att skapa en anpassad översättningsmodell:

1.Använd Custom Translator-portalen.

2.Koppla ett workspace till din Azure AI Translator-resurs.

3.Skapa ett projekt för översättning.

4.Ladda upp träningsdata (t.ex. tidigare översatta texter).

5.Träna modellen för att anpassa översättningar efter behov.

Användning via Translator API:

•Ange en **category-parameter** med **Project Category ID** (visas i gränssnittet).

•Detta gör att API:et använder den tränade modellen istället för standardöversättning.

- **Fördel:** Anpassad översättning för specifika domäner (t.ex. bilindustri, medicin).

A custom translation model is useful when you need to define translations for organization or industry-specific terms that aren't defined in the default Translator model.

For more details, see the Quickstart at https://learn.microsoft.com/azure/ai-services/Translator/custom-translator/quickstart

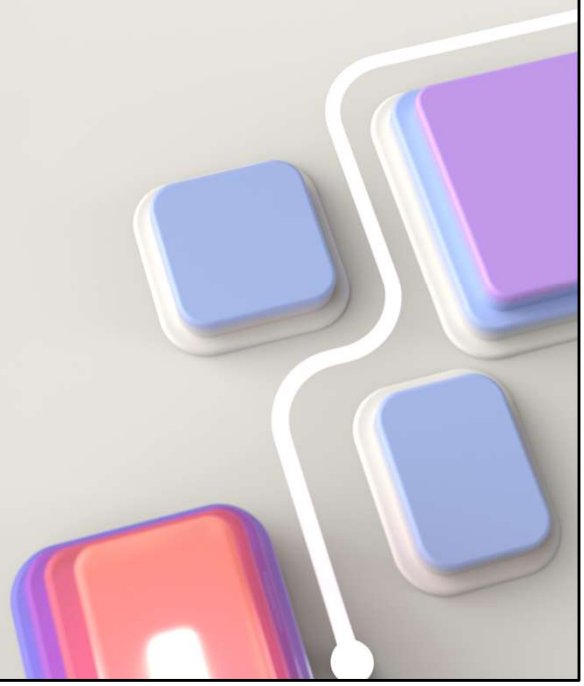
Demo – Translate Text

Detect language

Translate text

Instructor demo, can use lab in repo: mslearn-ai-language/Instructions/Exercises at main · MicrosoftLearning/mslearn-ai-language (github.com) for translation

Build a question answering solution



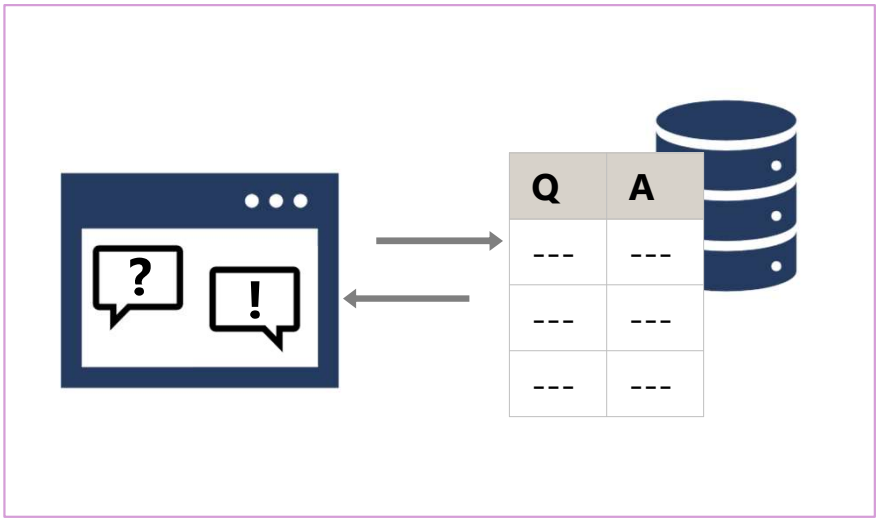
Learning Objectives

After completing this module, you will be able to:

1. Describe the question answering capabilities of Azure AI Language.
2. Describe the differences between question answering and conversational language understanding.
3. Create a knowledge base.
4. Implement multi-turn conversation.
5. Test and publish a knowledge base.
6. Consume a published knowledge base.
7. Implement active learning.

Introduction to Question Answering

- **Knowledge base of question and answer pairs with natural language understanding**
- **Published as a REST endpoint for applications to consume**
- **Available through language specific SDKs**



Bilden beskriver **Question Answering** i Azure AI.

Vad är det?

En tjänst som använder en **kunskapsbas** med frågor och svar för att förstå naturligt språk och ge **relevanta svar**.

Hur fungerar det?

1.Bygg en kunskapsbas

1. Innehåller **frågor och svar** som kan användas av systemet.

2.Publicera som en REST API-endpoint

1. Applikationer kan **fråga API:t** och få **automatiska svar** baserade på kunskapsbasen.

3.Använd genom SDKs

1. Färdiga språkpaket gör det **enkelt att integrera** i olika applikationer.

Exempel

🟢 **Fråga:** *"Hur återställer jag mitt lösenord?"*
🔵 **Svar från API:t:** *"Gå till inställningar och klicka på 'Glömt lösenord'."*
🚀 **Perfekt för chatbots, FAQ-system och automatiserade kundsupportlösningar!**

Question Answering vs Language Understanding

Question answering

- User submits a question, expecting an answer
- Service uses a layered ranking approach with a natural language processing re-ranking model to match the question to an answer in the knowledge base
- Response is a static answer to a known question
- Client application presents the answer to the user

Language understanding

- User submits an utterance, expecting an appropriate response or action
- Service uses natural language understanding to interpret the utterance, match it to an intent, and identify entities
- Response indicates the most likely intent and referenced entities
- Client application is responsible for performing appropriate action based on the detected intent

The custom question answering system uses a layered ranking approach. The data is stored in Azure search, which also serves as the first ranking layer. The top results from Azure search are then passed through custom question answering's NLP re-ranking model to produce the final results and confidence score.

Creating a Knowledge Base

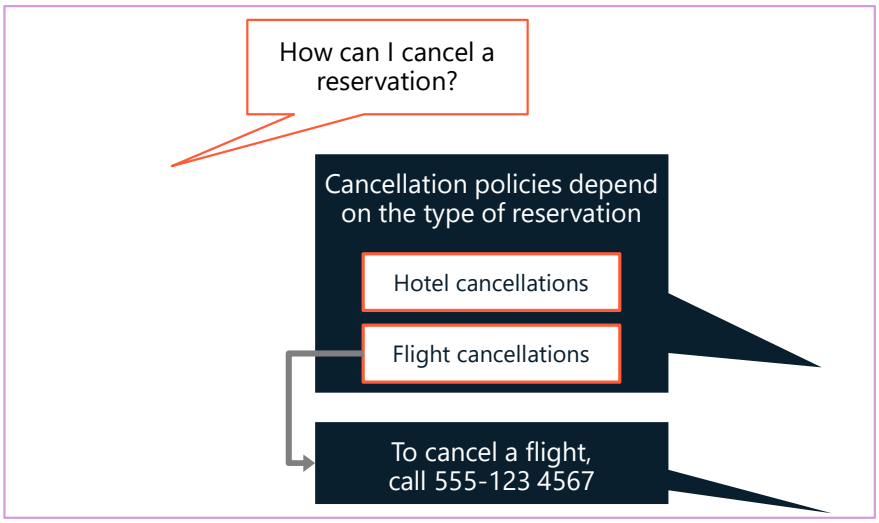
Use the Language Studio portal

1. Create an **Azure AI Language service** resource in your Azure subscription
2. In Language Studio, select your Azure AI Language resource and **create a Custom question answering** project.
3. Populate the knowledge base:
   - Import from existing FAQ web page
   - Upload document files
   - Add pre-defined "chit-chat" pairs
5. Create the knowledge base and edit question and answer pairs

Multi-turn conversation

Add follow-up prompts to define multi-turn exchanges

- Can reference existing question and answer pairs
- Can be restricted to follow-up responses only



Testing and publishing a Knowledge Base

Test interactively in Language Studio

- Inspect results to see confidence scores
- Add alternative phrases to improve scores as necessary

Publish the trained knowledge base

- Creates an HTTP REST-based endpoint for client apps to consume
- Published knowledge base can be used with SDKs within your app

Creating client apps

- REST Interface or SDKs
- Submit questions to the endpoint

```
{
  "question": "What do I need to do to
cancel a reservation?",
  "top": 2,
  "scoreThreshold": 20,
  "strictFilters": [
    {
      "name": "category",
      "value": "api"
    }
  ]
}
```

```
{
  "answers": [
    {
      "score": 27.74823341616769,
      "id": 20,
      "answer": "Call us on 555 123 4567 to
cancel a reservation.",
      "questions": [
        "How can I cancel a reservation?"
      ],
      "metadata": [
        {
          "name": "category",
          "value": "api"
        }
      ]
    }
  ]
}
```

Confidence score

Answer text

Best question match

Bilden visar hur en **klientapp** kan använda en **REST API** eller **SDK** för att skicka frågor och få svar från en **kunskapsbas**.

Hur det fungerar:

1.Frågan skickas till API:t

1. Exempel: "*What do I need to do to cancel a reservation?*"
2. Inkluderar **filter och tröskelvärde** (scoreThreshold: 20 för att bara få relevanta svar).

2.API:t returnerar ett svar

1. **Bästa matchade fråga:** "*How can I cancel a reservation?*"
2. **Svar:** "*Call us on 555 123 4567 to cancel a reservation.*"
3. **Confidence Score** anger hur säker modellen är på svaret.

Används för att bygga FAQ-bots, chatbots och kundtjänstsystem!

Här betyder "api" att **frågan och svaret hör till en kategori relaterad till API:er**. Detta kan syfta på frågor om **API-användning, teknisk dokumentation eller integrationer**.

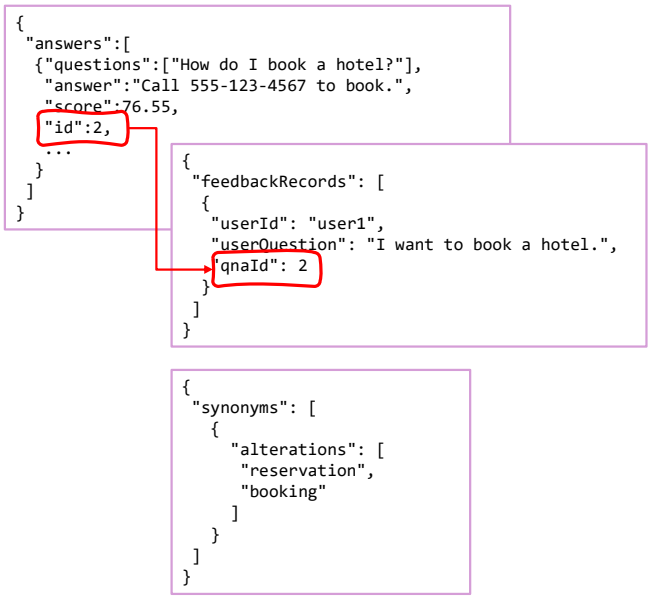
Improving Question Answering Performance

Enable *Active Learning* to suggest alternatives when multiple questions have similar scores for user input

- **Implicit**: The service identifies potential alternative phrases for questions; and presents suggestions in the Language Studio. Periodically review and accept/reject the suggestions.
- **Explicit**: The service returns multiple possible question matches to the user, and the user identifies the correct one. The client app then uses the API to submit feedback items, identifying the correct answer.

Create *Synonyms* for terms with the same meaning

- Add synonyms to the knowledge base through the API or Language Studio interface.



Bilden visar hur man kan förbättra Question Answering-prestanda med Active Learning och synonymer.

1. Active Learning (Lär sig av användaren)

📌 Implicit – Systemet föreslår alternativa frågor automatiskt, som kan granskas och godkännas i Language Studio.

📌 Explicit – Användaren väljer bästa frågematchning och skickar feedback via API.

🔹 Exemplet visar en användare som frågar:
"*I want to book a hotel.*"
API kopplar det till en befintlig fråga "*How do I book a hotel?*" och identifierar rätt svar via qnaId: 2.

2. Synonymer för bättre förståelse

📌 Synonymer (t.ex. *reservation = booking*) kan läggas till för att förbättra sökresultaten.

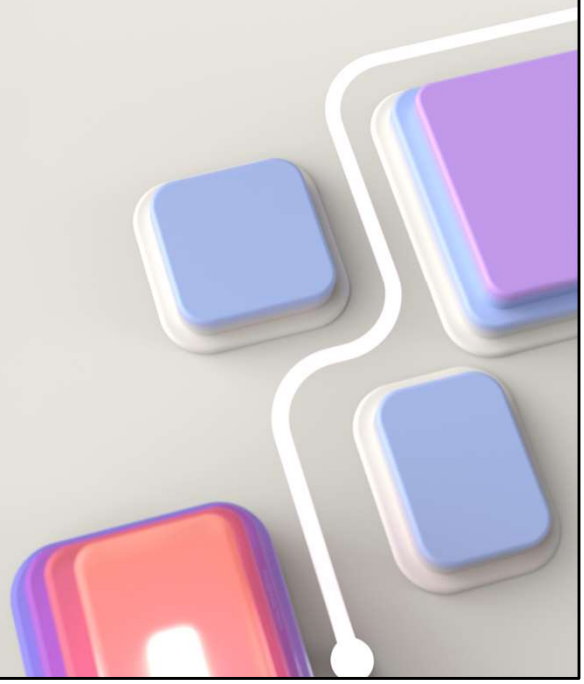
🚀 Effekt: Systemet blir smartare över tid och kan förstå fler sätt att ställa samma fråga!

Exercise – Create a Question Answering solution

Create and edit a knowledge base

Train, test, and deploy the knowledge base

Build a conversational language understanding app



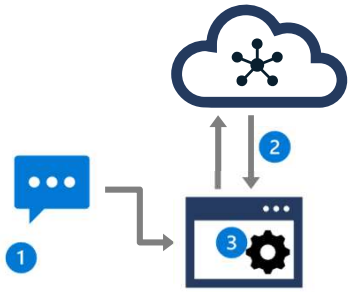
Learning objectives

After completing this module, you will be able to:

1. Provision an Azure AI Language resource
2. Define intents, entities, and utterances
3. Use patterns to differentiate similar utterances and use pre-built entity components
4. Train, test, publish, and review a model
5. Describe Azure AI Language Understanding features

Introduction to language understanding

1. An app accepts natural language input from a user
2. A language model is used to determine semantic meaning (the user's *intent*)
3. The app performs an appropriate action



Natural Language Processing (NLP) requires a language model to interpret user input

Often this activity is referred to as *natural language understanding* (NLU)

Conversational language understanding (CLU) is an Azure service to enable you to build natural language understanding component to be used in an end-to-end conversational application.

Intents and utterances

To train a language understanding model:

- Specify *utterances* that represent expected natural language input
- Map utterances to *intents* that assign semantic meaning

Utterance	Intent
What time is it?	GetTime
Tell me the time.	
What is the weather forecast?	GetWeather
Do I need an umbrella?	
Turn the light on.	TurnOnDevice
Switch on the fan.	
Hello	None

Entities

Define *entities* to add specific context to intents

Utterance	Intent	Entities
What is the time?	GetTime	
What time is it in London?	GetTime	Location (London)
What's the weather forecast for Paris?	GetWeather	Location (Paris)
Will I need an umbrella tonight?	GetWeather	Time (tonight)
What's the forecast for Seattle tomorrow?	GetWeather	Location (Seattle), Time (tomorrow)
Turn the light on.	TurnOnDevice	Device (light)
Switch on the fan.	TurnOnDevice	Device (fan)

Entity types:

Learned	List	Prebuilt
Machine learned through training	Term in a defined list	Common types like numbers and date/times

Prebuilt entity components

Prebuilt components automatically predict common types from utterances:

Quantities

- Age, Number, Percentage, Currency, others…

Datetime

- "June 23, 1976", "7 AM", "6:49 PM", "Tomorrow at 7 PM", "Next Week".

Email

- "user@contoso.com"

Phone number

- US Phone Numbers such as "+1 123 456 7890" or "(123)456-7890".

URL

- "https://learn.microsoft.com/"

Azure AI Language service capabilities

Features fall into two categories:

Preconfigured features – Can be used without labeling or training:

- Summarization
- Named entity recognition
- PII detection
- Key phrase detection
- Sentiment analysis
- Language detection

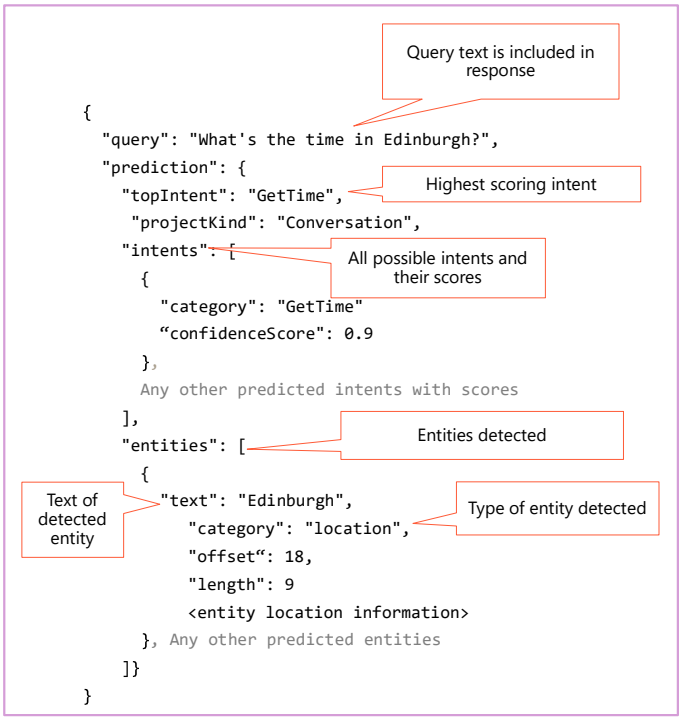
Learned features – Require labeling, training, and deploying to utilize

- Conversational language understanding
- Custom named entity recognition
- Custom text classification
- Question answering

Processing predictions

Submit a request to a published slot, specifying:

- **Kind –** Indicates which language feature you're requesting. For example, **kind** is defined as *Conversation* for conversational language understanding, or *EntityRecognition* to detect entities
- **Parameters –** Indicates the values for various input parameters. These parameters vary depending on the feature.
- **Analysis input –** Specifies the input documents or text strings to be analyzed by the Azure AI Language service.



Bilden visar hur en förfrågan skickas till en språkmodell för att **bearbeta och tolka en fråga**.

Hur det fungerar

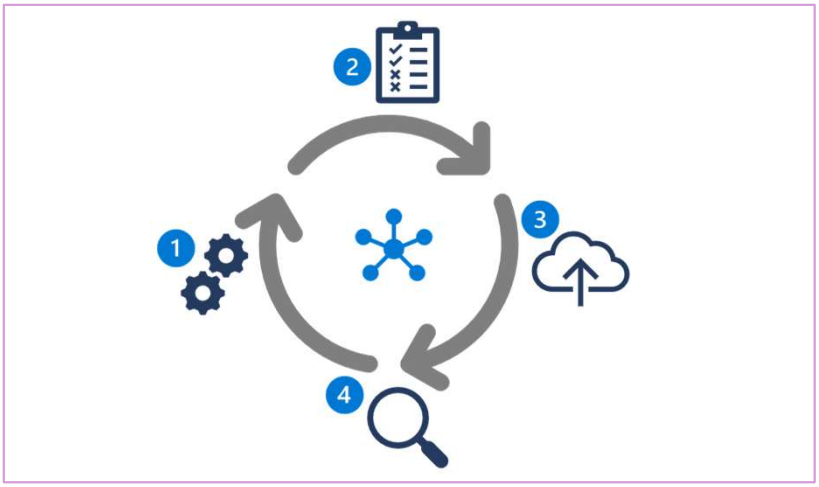
📌 **Förfrågan**: En användare frågar *"What's the time in Edinburgh?"*
📌 **Intent** (*avsikt*): Modellen tolkar frågan och matchar den mot **GetTime** (med högsta sannolikhet 0.9).
📌 **Entities** (*entiteter*): Modellen identifierar att *"Edinburgh"* är en **plats**.

Nyckelbegrepp

🔹 **Kind** – Anger vilken typ av språkmodell som används (t.ex. konversation eller entity detection).
🔹 **Parameters** – Specificerar inställningar för förfrågan.
🔹 **Analysis Input** – Själva texten som analyseras av **Azure AI Language Service**.
🚀 **Resultat:** API:t förstår både **avsikt** (hitta tid) och **entitet** (Edinburgh), vilket gör att ett program kan ge rätt svar.

Training, testing, publishing, and reviewing

1. Train a model to learn intents and entities from sample utterances
2. Test the model interactively or using a testing dataset with known labels
3. Deploy a trained model to a public endpoint so client apps can use it
4. Review predictions and iterate on utterances to train your model



Exercise – Create a conversational language understanding app

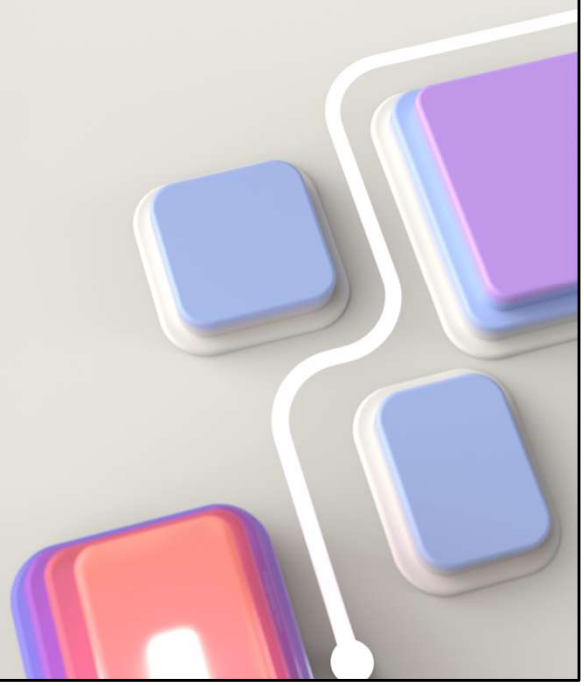
Create intents

Create entities

Test and publish a language model

Query your model from a client app

Custom classification and named entity extraction



Learning Objectives

After completing this module, you will be able to:

1. Label documents, train and deploy models for custom classification
2. Understand model performance and see where to improve your model
3. Use your custom model in an app

Custom Text Classification

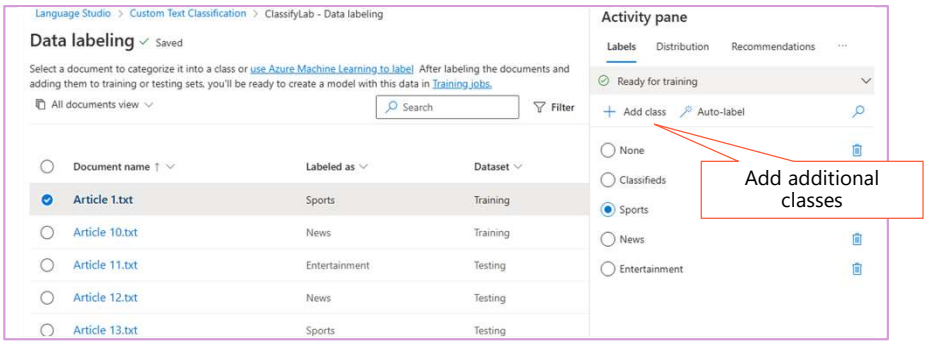
Assign custom labels to documents

1. Connect to documents in Azure
2. Define class labels to assign to your documents
3. Label documents
4. Train your model

Call your model through the Language API

- Specify project and deployment name

Can be single label or multi label projects



Bilden visar **Custom Text Classification** i Azure, där du kan träna en AI-modell för att automatiskt kategorisera textdokument.

Hur det fungerar

📌 **1. Koppla dokument** – Ladda upp och anslut dokument till Azure.
📌 **2. Skapa klasser** – Definiera etiketter (t.ex. *Sport*, *Nyheter*, *Underhållning*).
📌 **3. Märk dokument** – Tilldela dokument till rätt kategori.
📌 **4. Träna modellen** – Lär AI:n att känna igen klassificeringar.

Användning

🚀 Modellen kan sedan användas via **Language API**, där du skickar text och får tillbaka en klassificering.
🔹 **Enkelklass** (ett dokument har en etikett) eller **flerklass** (flera etiketter per dokument).

- 🔹 **Enkelklass** – Varje dokument tilldelas **endast en etikett** (t.ex. *Sport*).
- 🔹 **Flerklass** – Ett dokument kan ha **flera etiketter** (t.ex. *Sport* och *Nyheter* samtidigt).

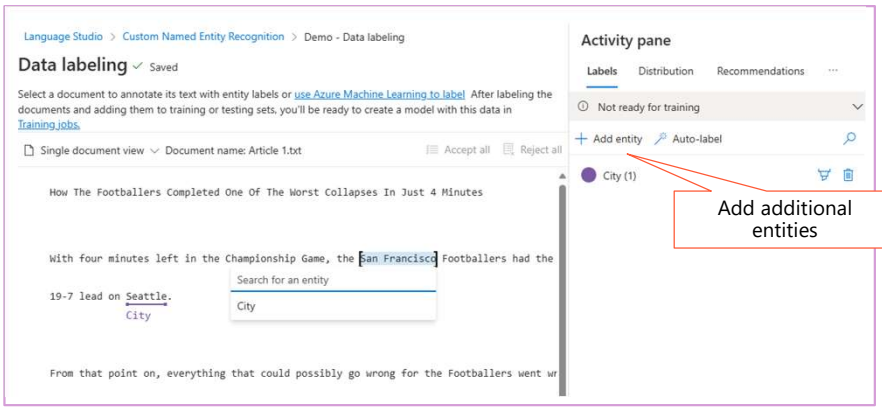
Custom Named Entity Recognition

Assign custom labels to entities in your documents

1. Connect to documents in Azure
2. Define entity labels to assign to your documents
3. Label documents completely and consistently
4. Train your model

Call your model through the Language API

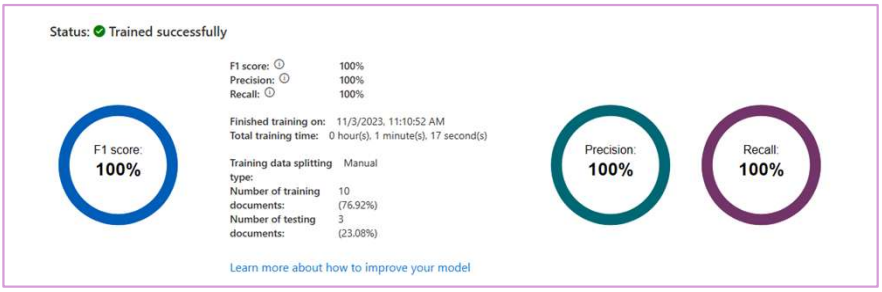
- Specify project and deployment name



Note that both of these custom models are similar, just vary a bit in how they're labelled and how they are used

Review and improve a model

1. Train a model to teach labels or entities
2. Review model performance to determine how to improve performance, including Confusion matrix
3. Determine what cases need to be added to your training data
4. Retrain your model with new data included, and repeat as necessary



Bildens model performance-mått:

•F1 Score (100%) – Balanserar precision och recall för att mäta modellens övergripande prestanda.

•Precision (100%) – Andelen förutsägelser som var korrekta av alla positiva resultat. Hög precision = få falska positiva.

•Recall (100%) – Andelen faktiska positiva fall som modellen lyckades identifiera. Hög recall = få falska negativa.

Ett perfekt resultat (100%) kan indikera en liten eller enkel dataset.

Demo – Extract custom entities

Create a custom named entity recognition project

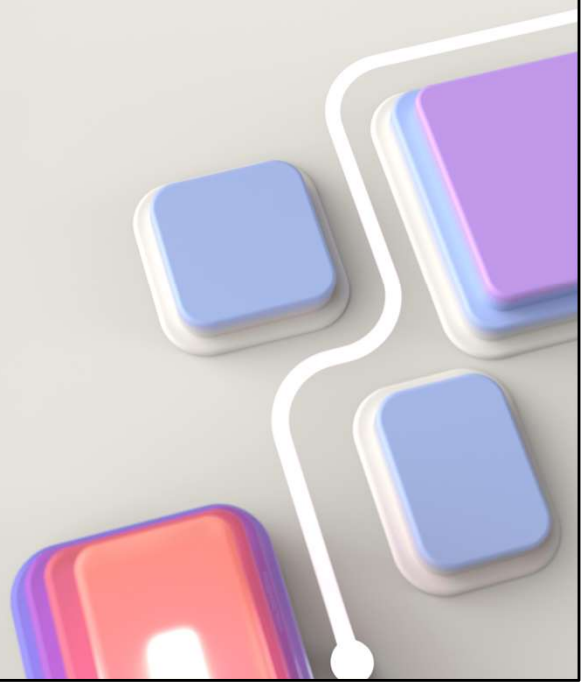
Create and label entities

Test and publish a custom model

Query your model from a client app

Instructor demo, can use lab in the repo: mslearn-ai-language/Instructions/Exercises at main · MicrosoftLearning/mslearn-ai-language (github.com) for text classification

Speech recognition, translation and synthesis



Learning Objectives

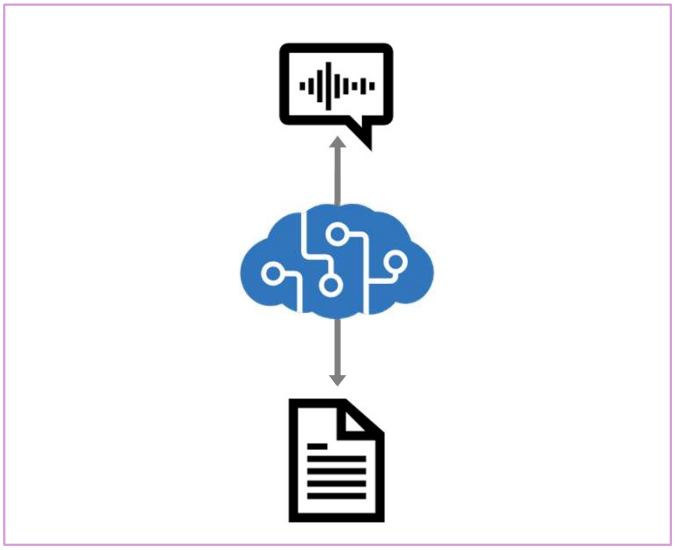
After completing this module, you will be able to:

1. Provision an Azure resource for the Azure AI Speech service
2. Use the Speech to text API to implement speech recognition
3. Use the Text to speech API to implement speech synthesis
4. Configure audio format and voices
5. Use Speech Synthesis Markup Language (SSML)

The Speech Service

Speech APIs

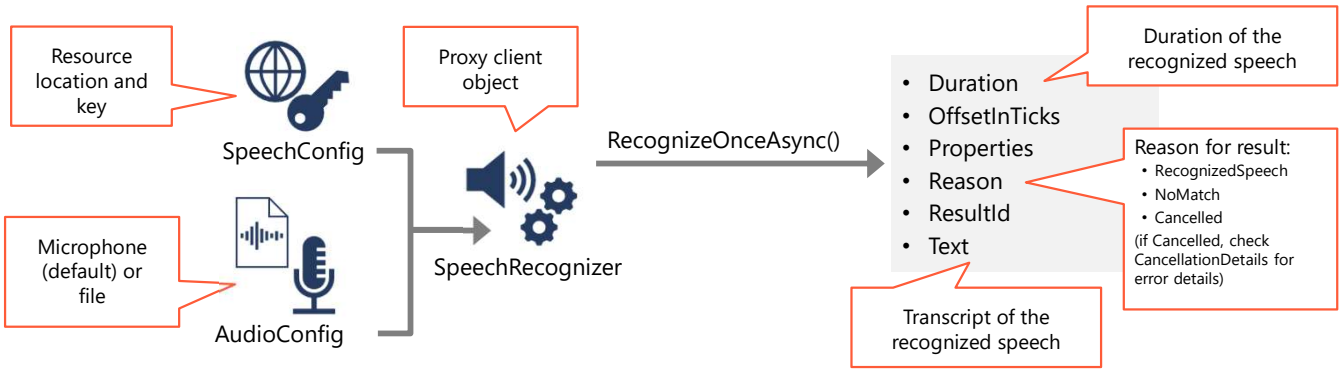
- Speech-to-Text API (speech recognition)
- Text-to-Speech API (speech synthesis)
- Speech Translation API
- Speaker Recognition API
- Intent Recognition (uses conversational language understanding)



In this lesson, we'll focus on speech-to-text and text-to-speech.

We'll cover speech translation in the next lesson of this module, and we'll look at Intent Recognition in a later module.

Speech-to-Text



Two REST APIs:

- Speech-to-text API – Used by Azure AI Speech SDK – preferred for most scenarios
- Speech-to-text Short Audio API – Useful for short (up to 60s) of audio

Azure AI Speech SDK (.NET, Python, JavaScript, etc.)

There are two REST APIs, one specifically for short bursts of speech, and the other for batch and custom speech recognition.

In most cases, you'll likely use one of the SDKs, which offer a consistent programming pattern for working with speech.

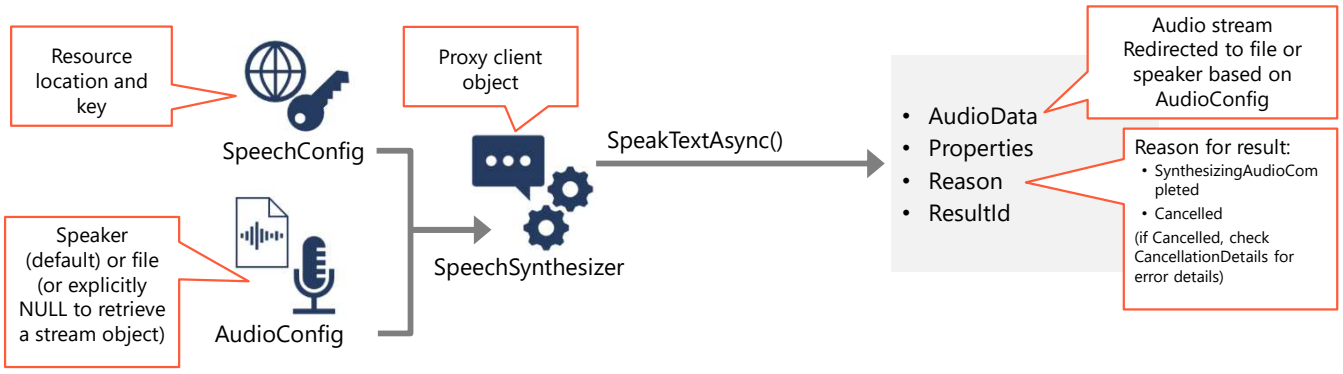
The slide shows a graphical representation of how to use SDK classes to recognize text.

The approach is similar for C#, Python, or JavaScript:

1. Create a SpeechConfig object with the key and region for your Azure AI Speech resource
2. Create an AudioConfig object indicating the input source (mic or audio file)
3. Use the config objects to create a SpeechRecognizer client object
4. Call a method to send a request to the service - in this case, the RecognizeOnceAsync method is used to recognize a single speech input using a non-blocking (asynchronous) call.
5. The results of the call include several details, the most important of which are:
   - Duration – (the duration of the speech in seconds)
   - Reason (the reason for the returned results (NoMatch means that the calls succeeded, but no speech was detected, Cancelled usually means something went wrong.)
   - Text (if the result is RecognizedText, this contains a transcription of the spoken input.

There are other methods you can use to perform batch transcription, submit multiple files for simultaneous transcription, and so on. For details, explore the SDK documentation for your preferred language.

Text-to-Speech



Two REST APIs:

- Text-to-speech API – Suitable for most scenarios
- Batch synthesis API – Convert large volumes of text to audio files

Azure AI Speech SDK (.NET, Python, JavaScript, etc.)

Audio Format and Voices

Audio Format

Select an audio format to specify:

- Audio file type
- Sample-rate
- Bit-depth

Voices

- Standard voices: Synthetic voices created from audio samples
- Neural voices: Natural sounding voices created using deep neural networks

```
speechConfig.SetSpeechSynthesisOutputFormat(SpeechSynthesisOutputFormat.Riff24Khz16BitMonoPcm);

speechConfig.SpeechSynthesisVoiceName = "en-GB-George";
```


Denna bild förklarar hur man hanterar **ljudformat** och **röster** vid talsyntes i Azure.

Ljudformat

För att generera ljud behöver du specificera:

•**Filtyp** (t.ex. WAV, MP3)

•**Samplingsfrekvens** (hur många gånger per sekund ljud samplas, t.ex. 24 kHz)

•**Bitdjup** (ljudkvalitet, t.ex. 16-bit)

I koden sätts formatet med:

speechConfig.SetSpeechSynthesisOutputFormat(SpeechSynthesisOutputFormat.Riff24Khz16BitMonoPcm); Detta väljer **24 kHz, 16-bit mono PCM (WAV-format)**.

Röster

Två typer av syntetiska röster:

•**Standardröster** – baserade på inspelade ljudprover.

•**Neurala röster** – mer naturliga, skapade med **neurala nätverk**.

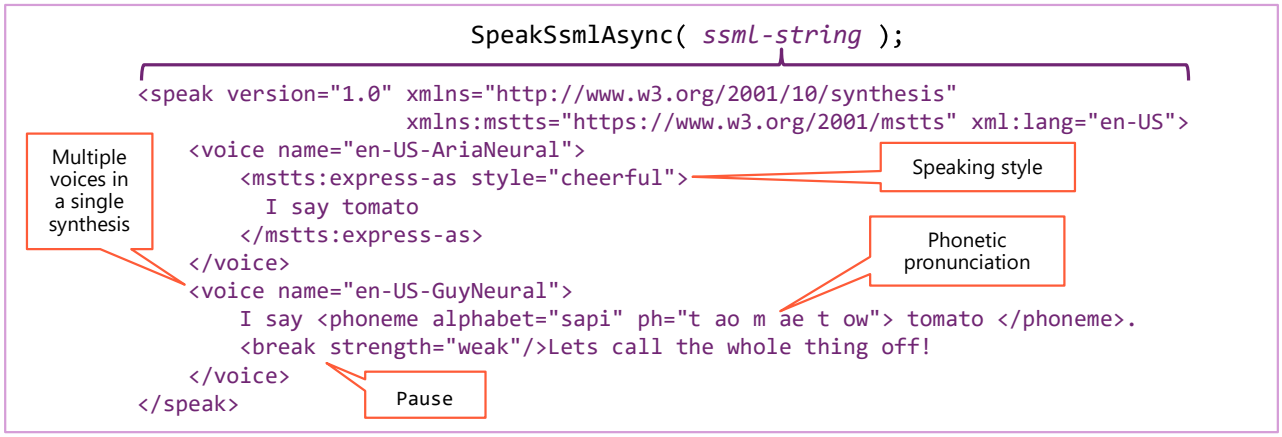
I koden väljs en brittisk engelsk röst:

speechConfig.SpeechSynthesisVoiceName = "en-GB-George"; Detta betyder att **rösten "George" med brittisk accent används för talsyntes**.

Speech Synthesis Markup Language (SSML)

XML-based language with customization options:

- Speaking styles (Neural voices only)
- Pauses and silence
- Phonemes (phonetic pronunciations)
- Prosody (speaking pitch, range, rate, etc.)
- "say-as" (number, date, time, address, etc.)
- Insert recorded speech or background audio



Denna bild förklarar **Speech Synthesis Markup Language (SSML)**, ett XML-baserat språk som används för att styra **text-till-tal (TTS)** i Azure.

Vad gör SSML?

SSML låter dig anpassa hur text uttalas genom att styra **röst**, **intonation**, **pauser**, och **fonetisk uttal**.

Kodförklaring

•<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xmlns:mstts="https://www.w3.org/2001/mstts" xml:lang="en-US"> **<speak>**: Definierar en SSML-struktur.

•**xmlns**: Anger XML-standarden för tal.

•**xml:lang="en-US"**: Anger att språket är engelska (USA).

Flera röster i samma syntes

•<voice name="en-US-AriaNeural"> <mstts:express-as style="cheerful"> I say tomato </mstts:express-as> </voice> **<voice name="en-US-AriaNeural">**: Väljer rösten "Aria" med en **neural röstmodell**.

•**<mstts:express-as style="cheerful">**: Ställer in en **glad ton** vid uttal.

•<voice name="en-US-GuyNeural"> I say <phoneme alphabet="sapi" ph="t ao m ae t ow"> tomato </phoneme>. <break strength="weak"/> Lets call the whole thing off! </voice> **<phoneme>**: Styr det fonetiska uttalet av "tomato".

•**<break strength="weak"/>**: Lägger in en kort **paus** innan nästa mening.

Varför använda SSML?

•Möjliggör **naturligare talsyntes** genom att anpassa uttal och intonation.

•**Flera röster** kan användas i samma syntes.

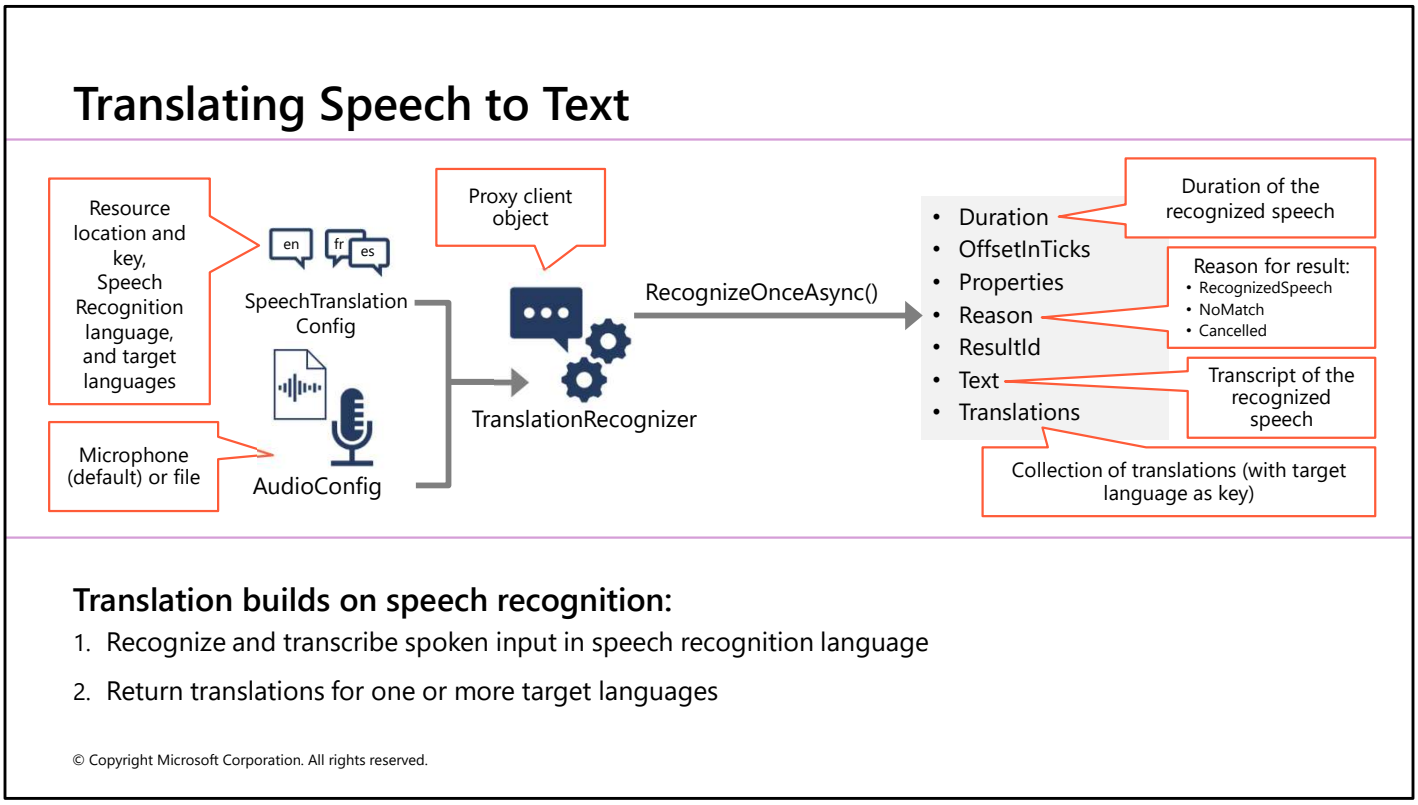
•**Pauser** och **fonetiska justeringar** förbättrar tydlighet och uttryck.

Perfekt för applikationer som kräver **realistisk talsyntes**, t.ex. virtuella assistenter och ljudböcker.

Exercise – Recognize and Synthesize Speech

Recognize Speech

Synthesize Speech



Denna bild visar hur **Azure Speech-to-Text och översättning** fungerar genom **SDK-klasser**.

Översikt av processen

1.Konfiguration av tjänsten

1. **SpeechTranslationConfig**: Anger **resursnyckel**, **region**, **igenkänningsspråk** och **målöversättningsspråk**.
2. **AudioConfig**: Definierar ljudkällan (mikrofon eller ljudfil).

2.Användning av översättningsigenkänning

1. **TranslationRecognizer** används som en klient för att hantera taligenkänning och översättning.
2. **RecognizeOnceAsync()** är en **asynkron metod** som gör en engångsigenkänning av tal.

3.Resultat från igenkänningen

1. **Duration**: Hur länge det inspelade talet varade.
2. **Reason**: Orsaken till resultatet:
    1. **RecognizedSpeech**: Talet har igenkänts korrekt.
    2. **NoMatch**: Ingen taligenkänning hittades.
    3. **Cancelled**: Något gick fel.
3. **Text**: Den transkriberade texten.
4. **Translations**: En samling av översättningar, där varje målöversättning har ett språk som nyckel.

Hur detta används i kod

•Skapa ett **SpeechTranslationConfig-objekt** med nyckel, region och språk.
•Definiera en **AudioConfig** för ljudinmatning.
•Använd dessa konfigurationer för att skapa en **TranslationRecognizer**.
•Kör **RecognizeOnceAsync()** för att känna igen och översätta tal.

Alternativa metoder

•**Batch-transkription**: Bearbeta flera ljudfiler samtidigt.
•**Anpassad taligenkänning**: Träna modellen för att förstå domänspecifikt språk.

Kort sagt, detta visar hur talsyntes och översättning automatiseras genom Azure AI Speech SDK!

There are two APIs, one specifically for short bursts of speech, and the other for batch and custom speech recognition.

In most cases, you'll likely use one of the SDKs, which offer a consistent programming pattern for working with speech.

The slide shows a graphical representation of how to use SDK classes to recognize text.

The approach is similar for C#, Python, or JavaScript:

1. Create a SpeechTranslationConfig object with the key and region for your Azure AI Speech resource
2. Use the SpeechTranslationConfig object to specify the speech recognition language (the language in which the input speech is spoken) and the target languages into which it should be translated.
3. Create an AudioConfig object indicating the input source (mic or audio file)
4. Use the config objects to create a SpeechRecognizer client object
5. Call a method to send a request to the service - in this case, the RecognizeOnceAsync method is used to recognize a single speech input using a non-blocking (asynchronous) call.
6. The results of the call include several details, the most important of which are:
    - Duration – (the duration of the speech in seconds)
    - Reason (the reason for the returned results (NoMatch means that the calls succeeded, but no speech was detected, Cancelled usually means something went wrong.)
    - Text (if the result is RecognizedText, this contains a transcription of the spoken input.

There are other methods you can use to perform batch transcription, submit multiple files for simultaneous transcription, and so on. For details, explore the SDK documentation for your preferred language.

Synthesizing Translations as Speech

Event-based synthesis

- Only supported for 1:1 translation (single target language)
- Specify desired voice in the **TranslationConfig**
- Use the **Synthesizing** event to retrieve audio stream
- Create an event handler
- Use Result.GetAudio() to retrieve byte stream

Manual synthesis

- Use for multiple target languages
- Translate to text then use Text-to-Speech API to synthesize each translation in the results

Den här bilden förklarar hur **översatt text kan omvandlas till tal** med hjälp av **Azure Speech Services**. Det finns två huvudsakliga metoder: **Event-driven syntes** och **Manuell syntes**.

1. Event-based synthesis (Händelsedriven syntes)

Detta tillvägagångssätt används **när endast ett mål språk ska genereras (1:1-översättning)**. Det innebär att översättningen direkt konverteras till tal utan att först sparas som text.

Steg för event-baserad syntes:

1.Ange önskad röst i TranslationConfig

1. Du väljer en specifik röst som ska användas för talgenereringen.

2.Använd Synthesizing-eventet för att hämta ljudströmmen

1. Systemet genererar tal i realtid, vilket gör det möjligt att strömma ljudet istället för att vänta på att hela översättningen ska bearbetas.

3.Skapa en händelsehanterare

1. En event handler används för att lyssna på **Synthesizing-eventet**, vilket gör att ljud kan hämtas och spelas upp direkt.

4.Använd Result.GetAudio() för att hämta ljud som en byte-ström

1. Med denna metod kan du hämta ljudet och vidare hantera det, t.ex. spela upp det eller spara det till en fil.

🔹 **Fördelar:**
✅ Snabb, eftersom ljudet genereras i realtid.
✅ Lämplig för live-översättning i samtal eller realtidsapplikationer.
✅ Kräver ingen extra konvertering från text till tal.

2. Manual synthesis (Manuell syntes)

Detta tillvägagångssätt används när **översättningen ska ske till flera språk samtidigt**. Istället för att direkt generera ljud från översättningen, delas processen upp i två steg:

1.Översätt tal till text

1. Systemet genererar textbaserade översättningar för flera språk.

2.Använd Text-to-Speech API för att skapa tal för varje språk

1. Varje översättning skickas till Azure **Text-to-Speech API**, som genererar ljud för varje språk separat.

🔹 **Fördelar:**
✅ Möjlighet att hantera **flera språk samtidigt**.
✅ Flexibilitet att **manipulera och redigera textöversättningen** innan tal skapas.
✅ Bättre för **asynkrona applikationer** där realtidsströmning inte är nödvändig.

Sammanfattning

•**Event-based synthesis** = Snabbt och realtidsbaserat, men bara för **ett** mål språk.

•**Manual synthesis** = Möjliggör **flera språk**, men kräver ett extra steg där text först genereras och sedan omvandlas till tal.

Valet mellan metoderna beror på applikationens krav! 😊

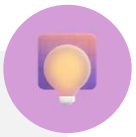
Extended interactive exercises

Custom text classification

Translate speech

https://aka.ms/azure-ai-language-lp

Knowledge check



1 **Which object should you use to specify that the speech input to be transcribed to text is in an audio file?**

- ☐ SpeechConfig
- ☑ AudioConfig
- ☐ SpeechRecognizer

2 **You have analyzed text that contains the word "Paris". How might you determine if this word refers to the French city or the character in Homer's *The Iliad*?**

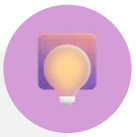
- ☐ Use the Azure AI Language service to extract key phrases.
- ☐ Use the Azure AI Language service to analyze sentiment.
- ☑ Use the Azure AI Language service to extract linked entities.

3 **When translating speech, in which cases can you can use the Synthesizing event to synthesize the translations and speech?**

- ☑ Only when translating to a single target language
- ☐ Only when translating to multiple target languages
- ☐ When translating to one or more target languages

Use the slide animation to reveal the correct answers.
1 b 2 c 3 a

Knowledge check



4 **Your app must interpret a command to book a flight to a specified city, such as "Book a flight to Paris." How should you model the city element of the command?**

- ☐ As an intent.
- ☐ As an utterance.
- ☑ As an entity.

5 **Your language model needs to detect an email when present in an utterance. What is the simplest way to extract that email?**

- ☐ Use Regular Expression entities.
- ☑ Use Prebuilt entity components
- ☐ Use Learned entity components.

6 **How should you create an application that monitors the comments on your company's web site and flags any indication that customers are unhappy?**

- ☐ Use the Azure AI Translator service to detect profanities in comments.
- ☑ Use the Azure AI Language service to perform sentiment analysis of the comments.
- ☐ Use the Azure AI Language service to extract named entities from the comments

Use the slide animation to reveal the correct answers.
4 c 5 b 6 b

Learning Path Recap

In this learning path, we learned to:

- Analyze and translate text
- Build a conversational language understanding model
- Build a question answering solution
- Speech recognition, synthesis, and translation
- Connect an app to Azure AI Language resources

Microsoft