**Train and evaluate a language understanding model**

**Train your conversational language understanding model**

After you have completed [labeling your utterances](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/how-to/tag-utterances), you can start training a model. Training is the process where the model learns from your [labeled utterances](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/how-to/tag-utterances).

To train a model, start a training job. Only successfully completed jobs create a model. Training jobs expire after seven days, after this time you will no longer be able to retrieve the job details. If your training job completed successfully and a model was created, it won't be affected by the job expiring. You can only have one training job running at a time, and you can't start other jobs in the same project.

The training times can be anywhere from a few seconds when dealing with simple projects, up to a couple of hours when you reach the [maximum limit](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/service-limits) of utterances.

Model evaluation is triggered automatically after training is completed successfully. The evaluation process starts by using the trained model to run predictions on the utterances in the testing set, and compares the predicted results with the provided labels (which establishes a baseline of truth).

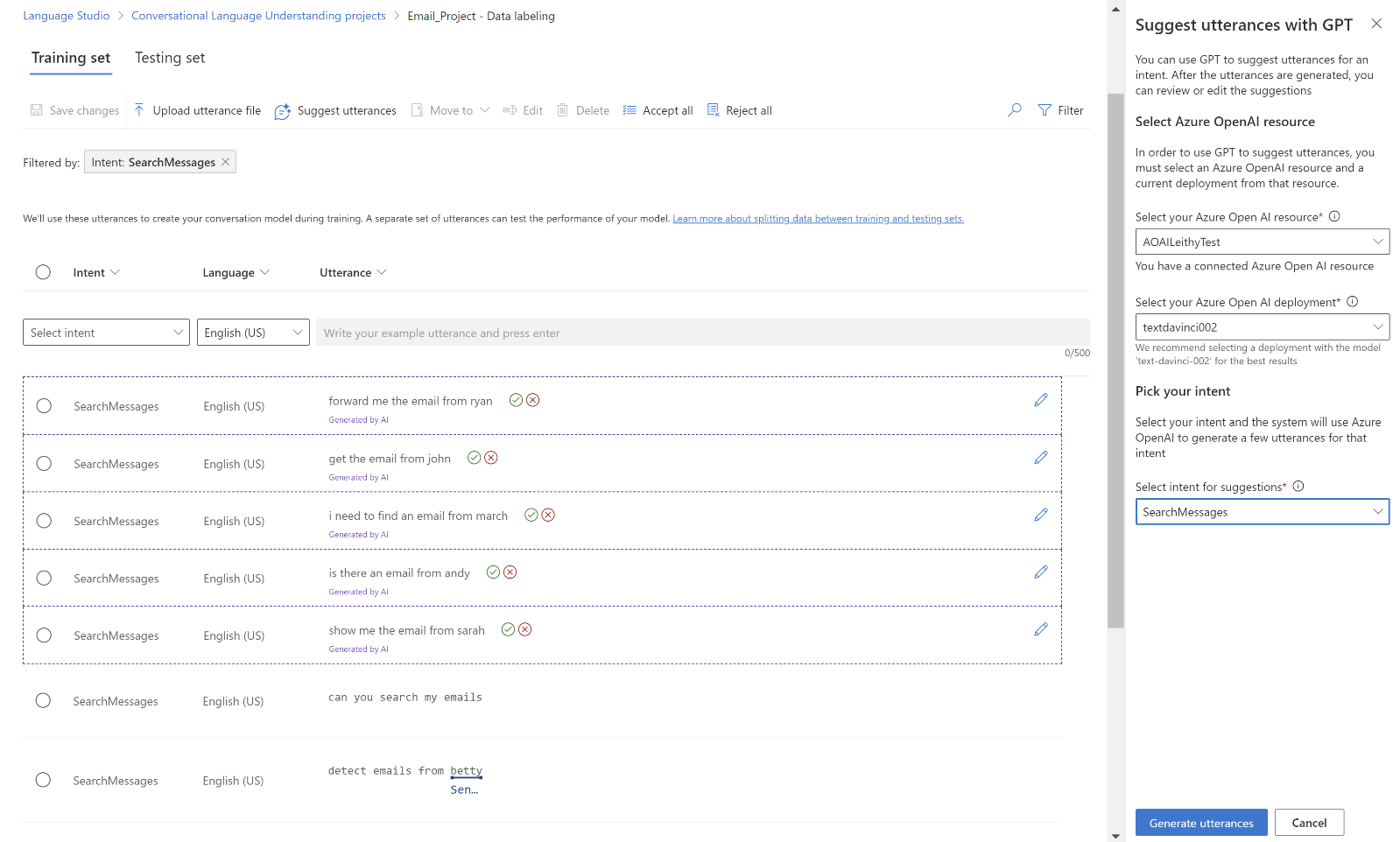
**Balance training data**

When it comes to training data, try to keep your schema well balanced. Including large quantities of one intent and very few of another results in a model that's biased toward particular intents.

To address this scenario, you might need to downsample your training set. Or you might need to add to it. To downsample, you can:

* Get rid of a certain percentage of the training data randomly.
* Analyze the dataset and remove overrepresented duplicate entries, which is a more systematic manner.

To add to the training set, in Language Studio, on the **Data labeling** tab, select **Suggest utterances**. Conversational Language Understanding sends a call to [Azure OpenAI](https://learn.microsoft.com/en-us/azure/ai-services/openai/overview) to generate similar utterances.

[](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/media/suggest-utterances.png#lightbox)

You should also look for unintended "patterns" in the training set. For example, look to see if the training set for a particular intent is all lowercase or starts with a particular phrase. In such cases, the model you train might learn these unintended biases in the training set instead of being able to generalize.

We recommend that you introduce casing and punctuation diversity in the training set. If your model is expected to handle variations, be sure to have a training set that also reflects that diversity. For example, include some utterances in proper casing and some in all lowercase.

**Data splitting**

Before you start the training process, labeled utterances in your project are divided into a training set and a testing set. Each one of them serves a different function. The **training set** is used in training the model, this is the set from which the model learns the labeled utterances. The **testing set** is a blind set that isn't introduced to the model during training but only during evaluation.

After the model is trained successfully, the model can be used to make predictions from the utterances in the testing set. These predictions are used to calculate [evaluation metrics](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/concepts/evaluation-metrics). It is recommended to make sure that all your intents and entities are adequately represented in both the training and testing set.

Conversational language understanding supports two methods for data splitting:

* **Automatically splitting the testing set from training data**: The system will split your tagged data between the training and testing sets, according to the percentages you choose. The recommended percentage split is 80% for training and 20% for testing.

**Note :** If you choose the **Automatically splitting the testing set from training data** option, only the data assigned to training set will be split according to the percentages provided.

* **Use a manual split of training and testing data**: This method enables users to define which utterances should belong to which set. This step is only enabled if you have added utterances to your testing set during [labeling](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/how-to/tag-utterances).

**Training modes**

CLU supports two modes for training your models

* **Standard training** uses fast machine learning algorithms to train your models relatively quickly. This is currently only available for **English** and is disabled for any project that doesn't use English (US), or English (UK) as its primary language. This training option is free of charge. Standard training allows you to add utterances and test them quickly at no cost. The evaluation scores shown should guide you on where to make changes in your project and add more utterances. Once you’ve iterated a few times and made incremental improvements, you can consider using advanced training to train another version of your model.
* **Advanced training** uses the latest in machine learning technology to customize models with your data. This is expected to show better performance scores for your models and will enable you to use the [multilingual capabilities](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/language-support#multi-lingual-option) of CLU as well. Advanced training is priced differently. See the [pricing information](https://azure.microsoft.com/pricing/details/cognitive-services/language-service) for details.

Use the evaluation scores to guide your decisions. There might be times where a specific example is predicted incorrectly in advanced training as opposed to when you used standard training mode. However, if the overall evaluation results are better using advanced, then it is recommended to use your final model. If that isn’t the case and you are not looking to use any multilingual capabilities, you can continue to use model trained with standard mode.

**Note :** You should expect to see a difference in behaviors in intent confidence scores between the training modes as each algorithm calibrates their scores differently.

**Train model using Language Studio**

1. Select **Train model** from the left side menu.

A computer screen shot of a box

Description automatically generated

1. Select **Start a training job** from the top menu.

A screenshot of a computer

Description automatically generated

1. Select **Train a new model** and enter a new model name in the text box. Otherwise, to replace an existing model with a model trained on the new data, select **Overwrite an existing model** and then select an existing model. Overwriting a trained model is irreversible, but it won't affect your deployed models until you deploy the new model.

A screenshot of a computer

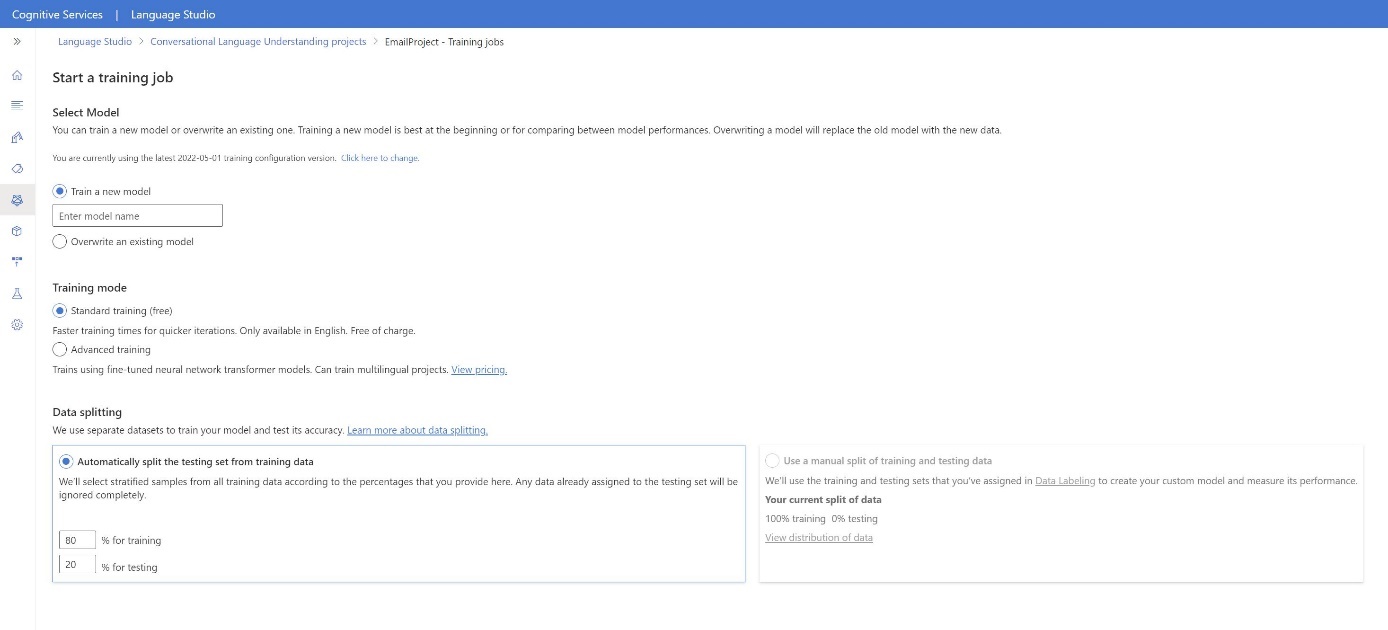
Description automatically generated

1. Select training mode. You can choose **Standard training** for faster training, but it is only available for English. Or you can choose **Advanced training** which is supported for other languages and multilingual projects, but it involves longer training times. Learn more about [training modes](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/how-to/train-model#training-modes).

A screenshot of a computer

Description automatically generated

1. Select a [data splitting](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/how-to/train-model#data-splitting) method. You can choose **Automatically splitting the testing set from training data** where the system will split your utterances between the training and testing sets, according to the specified percentages. Or you can **Use a manual split of training and testing data**, this option is only enabled if you have added utterances to your testing set when you [labeled your utterances](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/how-to/tag-utterances).
2. Select the **Train** button.

[](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/media/train-model.png#lightbox)

A screenshot of a computer

Description automatically generated

1. Select the training job ID from the list. A panel will appear where you can check the training progress, job status, and other details for this job.

A screenshot of a computer

Description automatically generated

**Note**

* + Only successfully completed training jobs will generate models.
  + Training can take some time between a couple of minutes and couple of hours based on the count of utterances.
  + You can only have one training job running at a time. You can't start other training jobs within the same project until the running job is completed.

A screenshot of a computer

Description automatically generated

* + The machine learning used to train models is regularly updated. To train on a previous [configuration version](https://learn.microsoft.com/en-us/azure/ai-services/language-service/concepts/model-lifecycle), select **Select here to change** from the **Start a training job** page and choose a previous version.

**Evaluation metrics for conversational language understanding models**

Model evaluation is triggered automatically after training is completed successfully. The evaluation process starts by using the trained model to predict user-defined intents and entities for utterances in the test set. Then the process compares them with the provided tags to establish a baseline of truth. The results are returned so that you can review the model's performance. For evaluation, conversational language understanding uses the following metrics:

* **Precision**: Measures how precise or accurate your model is. It's the ratio between the correctly identified positives (true positives) and all identified positives. The precision metric reveals how many of the predicted classes are correctly labeled.

Precision = #True\_Positive / (#True\_Positive + #False\_Positive)

* **Recall**: Measures the model's ability to predict actual positive classes. It's the ratio between the predicted true positives and what was tagged. The recall metric reveals how many of the predicted classes are correct.

Recall = #True\_Positive / (#True\_Positive + #False\_Negatives)

* **F1 score**: The F1 score is a function of precision and recall. It's needed when you seek a balance between precision and recall.

F1 Score = 2 \* Precision \* Recall / (Precision + Recall)

Precision, recall, and the F1 score are calculated for:

* Each entity separately (entity-level evaluation).
* Each intent separately (intent-level evaluation).
* For the model collectively (model-level evaluation).

The definitions of precision, recall, and evaluation are the same for entity-level, intent-level, and model-level evaluations. However, the counts for *true positives*, *false positives*, and *false negatives* can differ. For example, consider the following text.

**Example**

* Make a response with "thank you very much."
* Reply with saying "yes."
* Check my email please.
* Email to Cynthia that dinner last week was splendid.
* Send an email to Mike.

The intents used are Reply, sendEmail, and readEmail. The entities are contactName and message.

The model could make the following predictions:

A screenshot of a computer

Description automatically generated

**Intent-level evaluation for Reply intent**

**A screenshot of a computer

Description automatically generated**

Precision = #True\_Positive / (#True\_Positive + #False\_Positive) = 1 / (1 + 1) = 0.5

Recall = #True\_Positive / (#True\_Positive + #False\_Negatives) = 1 / (1 + 1) = 0.5

F1 score = 2 \* Precision \* Recall / (Precision + Recall) = (2 \* 0.5 \* 0.5) / (0.5 + 0.5) = 0.5

**Intent-level evaluation for sendEmail intent**

**A screenshot of a computer

Description automatically generated**

Precision = #True\_Positive / (#True\_Positive + #False\_Positive) = 1 / (1 + 1) = 0.5

Recall = #True\_Positive / (#True\_Positive + #False\_Negatives) = 1 / (1 + 1) = 0.5

F1 score = 2 \* Precision \* Recall / (Precision + Recall) = (2 \* 0.5 \* 0.5) / (0.5 + 0.5) = 0.5

**Intent-level evaluation for readEmail intent**

A screenshot of a computer

Description automatically generated

Precision = #True\_Positive / (#True\_Positive + #False\_Positive) = 1 / (1 + 0) = 1

Recall = #True\_Positive / (#True\_Positive + #False\_Negatives) = 1 / (1 + 0) = 1

F1 score = 2 \* Precision \* Recall / (Precision + Recall) = (2 \* 1 \* 1) / (1 + 1) = 1

**Entity-level evaluation for contactName entity**

A screenshot of a computer

Description automatically generated

Precision = #True\_Positive / (#True\_Positive + #False\_Positive) = 1 / (1 + 0) = 1

Recall = #True\_Positive / (#True\_Positive + #False\_Negatives) = 1 / (1 + 1) = 0.5

F1 score = 2 \* Precision \* Recall / (Precision + Recall) = (2 \* 1 \* 0.5) / (1 + 0.5) = 0.67

**Entity-level evaluation for message entity**

**A screenshot of a phone

Description automatically generated**

Precision = #True\_Positive / (#True\_Positive + #False\_Positive) = 2 / (2 + 1) = 0.67

Recall = #True\_Positive / (#True\_Positive + #False\_Negatives) = 2 / (2 + 1) = 0.67

F1 Score = 2 \* Precision \* Recall / (Precision + Recall) = (2 \* 0.67 \* 0.67) / (0.67 + 0.67) = 0.67

**Model-level evaluation for the collective model**

**A screenshot of a computer

Description automatically generated**

Precision = #True\_Positive / (#True\_Positive + #False\_Positive) = 6 / (6 + 3) = 0.67

Recall = #True\_Positive / (#True\_Positive + #False\_Negatives) = 6 / (6 + 4) = 0.60

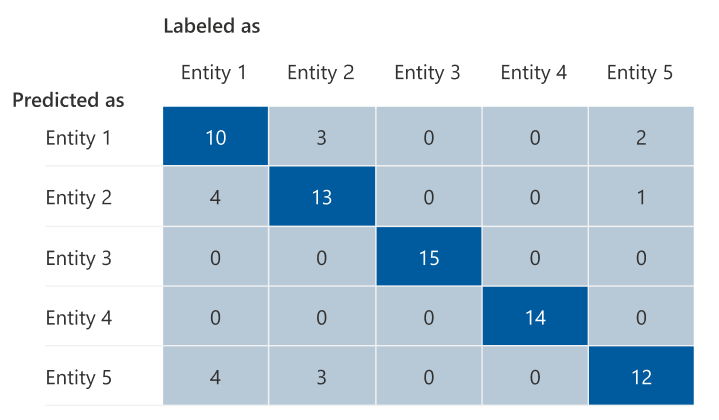
F1 score = 2 \* Precision \* Recall / (Precision + Recall) = (2 \* 0.67 \* 0.60) / (0.67 + 0.60) = 0.63

**Confusion matrix**

A confusion matrix is an N x N matrix used for model performance evaluation, where N is the number of entities or intents. The matrix compares the expected labels with the ones predicted by the model. The matrix gives a holistic view of how well the model is performing and what kinds of errors it's making.

You can use the confusion matrix to identify intents or entities that are too close to each other and often get mistaken (ambiguity). In this case, consider merging these intents or entities together. If merging isn't possible, consider adding more tagged examples of both intents or entities to help the model differentiate between them.

The highlighted diagonal in the following image shows the correctly predicted entities, where the predicted tag is the same as the actual tag.

**[](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/media/confusion-matrix-example.png#lightbox)**

You can calculate the intent-level or entity-level and model-level evaluation metrics from the confusion matrix:

* The values in the diagonal are the true positive values of each intent or entity.
* The sum of the values in the intent or entities rows (excluding the diagonal) is the false positive of the model.
* The sum of the values in the intent or entities columns (excluding the diagonal) is the false negative of the model.

Similarly:

* The true positive of the model is the sum of true positives for all intents or entities.
* The false positive of the model is the sum of false positives for all intents or entities.
* The false negative of the model is the sum of false negatives for all intents or entities.

**Guidance**

After you train your model, you see some guidance and recommendations on how to improve the model. We recommend that you have a model covering every point in the guidance section.

* **Training set has enough data**: When an intent or entity has fewer than 15 labeled instances in the training data, it can lead to lower accuracy because the model isn't adequately trained on that intent. In this case, consider adding more labeled data in the training set. You should only consider adding more labeled data to your entity if your entity has a learned component. If your entity is defined only by list, prebuilt, and regex components, this recommendation doesn't apply.
* **All intents or entities are present in test set**: When the testing data lacks labeled instances for an intent or entity, the model evaluation is less comprehensive because of untested scenarios. Consider having test data for every intent and entity in your model to ensure that everything is being tested.
* **Unclear distinction between intents or entities**: When data is similar for different intents or entities, it can lead to lower accuracy because they might be frequently misclassified as each other. Review the following intents and entities and consider merging them if they're similar. Otherwise, add more examples to better distinguish them from each other. You can check the Confusion matrix tab for more guidance. If you're seeing two entities constantly being predicted for the same spans because they share the same list, prebuilt, or regex components, make sure to add a *learned* component for each entity and make it *required*.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated