Transparency Note: Custom Named Entity Recognition

Updated 02/02/2022

Table of Contents

What is a Transparency Note?

An AI system includes not only the technology, but also the people who will use it, the people who will be affected by it, and the environment in which it is deployed. Creating a system that is fit for its intended purpose requires an understanding of how the technology works, what its capabilities and limitations are, and how to achieve the best performance. Microsoft's Transparency Notes are intended to help you understand how our AI technology works, the choices system owners can make that influence system performance and behavior, and the importance of thinking about the whole system, including the technology, the people, and the environment. You can use Transparency Notes when developing or deploying your own system, or share them with the people who will use or be affected by your system.

Microsoft's Transparency Notes are part of a broader effort at Microsoft to put our AI Principles into practice. To find out more, see the Microsoft AI principles.

Introduction to Custom Named Entity Recognition

Custom Named Entity Recognition (custom NER) is a cloud-based API service for information extraction. The service applies machine-learning intelligence so you can build custom models for information extraction tasks.

Custom NER can be used to extract information from .txt files. For example, a financial institution might want to build an automated notification system to remind clients of their payments due. The organization uses custom NER to extract relevant information from loan agreements, such as the client name, loan amount, interest rate, and payment date. The system can further process the extracted entities to send a reminder to the client with the next payment date and amount due.

The basics of Custom Named Entity Recognition

Custom Named Entity Recognition enables its users to build custom machine learning models to extract domain-specific entities from unstructured text, such as contracts or financial documents.

By creating a custom NER project, developers can iteratively tag entities within the data, train, evaluate, and improve model performance before making it available for consumption. The quality of the tagged data greatly affects model performance. To simplify building and customizing your model, the service offers a custom web portal that can be accessed through Language Studio.

Custom NER terminology

The following terms are commonly used with this feature:

Term	Definition
Project	A *project* is a work area for building your custom ML models based on your data. Your project can only be accessed by you and others who have access to the Azure resource being used. Within a project you can tag entities within the data, build models, evaluate and improve models where necessary, and eventually deploy a model. You can have multiple models within your project, all built on the same dataset.
Model	A *model* is an object that is trained to do a certain task, in this case custom entity recognition. Models are trained by providing tagged data to learn from so they can later be used for recognition tasks.
Entity	An *entity* is a span of text that indicates a certain type of information. The text span can consist of one or more words. In the scope of custom NER, entities represent the information that the user wants to extract from the text. Developers tag entities within their data with the needed entities before passing it to the model for training. For example "Invoice number", "Start date", "Shipment number", "Birthplace", "Origin city", "Supplier name" or "Client address".

Example use cases

Here are some examples of when you might use custom NER:

- **Knowledge mining to enhance semantic search:** Search is foundational to any app that surfaces text content to users. Common scenarios include catalog or document search, retail product search, or knowledge mining for data science. Many enterprises across various industries want to build a rich search experience over private, heterogeneous content, which includes both structured and unstructured documents. As a part of their pipeline, developers can use custom NER for extracting entities from the text that are relevant to their industry. These entities can be used to enrich the indexing of the file for a more customized search experience.
- **Information extraction from unstructured text:** Many financial and legal organizations extract and normalize data from thousands of complex, unstructured text sources on a daily basis. Such sources include bank statements, legal agreements, or bank forms. For example, mortgage application data extraction done manually by human reviewers may take several days to extract. Automating these steps simplifies the process and saves cost, time, and effort.
- **Audit and compliance:** Instead of manually reviewing significantly long text files to audit and apply policies, IT departments in financial or legal enterprises can use custom NER to build automated solutions. These solutions can be helpful to enforce compliance policies, and set up necessary business rules based on knowledge mining pipelines that process structured and unstructured content.

Considerations when choosing a use case

Be aware of the following guidance when you use custom NER:

- **Avoid using custom NER for decisions that might have serious adverse impacts.** For example, avoid scenarios that include medical or health diagnosis based on extracted information from an individual's medical history form, or charging a user's bank account based on extracted values. Additionally, it's advisable to include human review of decisions that have the potential for serious impacts on individuals.

- **Avoid creating custom entities that extract unnecessary or sensitive information.** Avoid extracting sensitive user information if it's not required for your use case. For example, if your scenario requires extracting your user's city and country, create entities that extract only the city and country from a user's address instead of extracting the entire address.

Characteristics and limitations of Custom Named Entity Recognition

Performance of Custom Named Entity Recognition (NER) models varies based on the scenario and input data. The following sections help you understand key concepts about performance and evaluation of custom NER models.

Performance evaluation metrics

Reviewing model evaluation is an important step in the custom NER's model development life cycle. It helps developers to learn how well their model is performing, and gives them an idea about the expected performance when the model is used in production.

In the process of building a model, developers can define training and testing sets during tagging. Alternatively, these sets can be chosen at random during training. Either way, the training and testing sets are essential for training and evaluating custom NER models. The training set is used to train the custom machine learning model, while the test set is used as a blind set to evaluate model performance.

The model evaluation process is triggered after training is completed successfully. The evaluation process takes place by using the trained model to predict user-defined entities from the test set. The predicted entities are compared with the provided data tags and evaluation metric are calculated accordingly.

To calculate a model's evaluation the extracted entities are categorized into: true positives, false positives, or false negatives. The following table further explains these terms, given the following text example:

The first party of this contract is John Smith resident of Frederick, Nebraska. And the second party is Forrest Ray resident of Corona, New Mexico. There is also Fannie Thomas resident Colorado Springs, Colorado.

Category	**Correct/Incorrect**	**Definition**	**Example**
True positive	Correct	The model recognizes this entity, and this is same as it has been tagged.	John Smith and Fannie Thomas were correctly predicted as Person. Colorado Springs was correctly predicted as City.
False positive	Incorrect	The model extracts a word that is not an entity, or correctly extracts an entity but predicts the wrong type for it.	Forrest was incorrectly predicted as City while it should have been Person. party was incorrectly extracted when it should not have been.
False negative	Incorrect	The model doesn't return a result when an entity is present in the data.	Nebraska was not extracted by the model while it should have been predicted as State.

The preceding categories are then used to calculate precision, recall, and a score (called the *F1 score*). These are the metrics provided to developers as part of the service's model evaluation. Here are more details:

- **Precision:** The measure of how accurate your model is. It's the ratio between the true positives and all the positives. Out of predicted positive entities, precision reveals how many of them are actual positives (belong to the right entity type as predicted).
- **Recall:** The measure of the model's ability to extract actual positive entities. It's the ratio between the predicted true positives and the actually tagged positives. Out of actual tagged entities, recall reveals how many of them are predicted correctly.
- **F1 score:** F1 score is a function of the previous two metrics. You need it when you seek a balance between precision and recall.

Any custom NER model will have both false negative and false positive errors. Developers need to consider how each type of error will affect the overall system, and carefully think through the tradeoffs for each use case.

Depending on your scenario, precision or recall might be a more suitable metric for evaluating your model's performance. For example, if your scenario is about extracting the interest rate from a financial contract, extracting the wrong entity would result in incorrect calculations and improper accounting. In this case your system should be more sensitive to false positives, and precision would then be a more relevant metric for evaluation.

However, if your scenario is about extracting entities to enrich your search index, failing to extract an entity would result in worse indexing. In this case, the system should be more sensitive to false negatives, and recall would then be a more relevant metric for evaluation.

If you want to optimize for general purpose scenarios, or when precision and recall are both important, you can rely on the F1 score. Understand that evaluation scores are subjective to your scenario and acceptance criteria, and there's no absolute metric that works for all scenarios.

Best practices for enhancing system performance

The following points will help you improve system performance:

- **Understand service limitations:** There are some limits enforced on the user, such as the number of files and entities in your data or entity length. Learn more about system limitations.
- **Plan your schema:** Identify the entities that need to be extracted from the data. Developers need to plan the schema accordingly. Learn more about recommended practices.
- **Select training data:** The quality of training data is an important factor in model quality. Using diverse and real-life data that's similar to the data you expect in production is preferable. Make sure to include all layouts and formats of text that would be expected in production. Learn more about recommended practices.
- **Tag data accurately:** The quality of your tagged data is a key factor in model performance, as this is considered the "ground truth" from which the model learns. In the process of tagging your data, remember to:
    - Tag completely. When you're tagging an entity, make sure you tag all the instances in all your files.
    - Tag precisely and consistently. When you're tagging an entity, make sure that you only tag what you need to extract, and avoid adding unnecessary words. The precision, consistency and completeness of your tagged data are key factors to determining model performance.
- **Ensure you have adequate representation of tagged entities in your dataset**: Examine data distribution to make sure that all your entities are adequately represented. If a certain entity is tagged less frequently than the others, this means this entity is under-represented and probably won't be recognized properly by the model at runtime. In this case, make sure that you tag all the instances of this entity. If you have done this already, consider adding more files to your dataset, and add more tags for this entity.
- **Add more data for better model performance**. Generally, more tagged data leads to better models provided that your data is tagged accurately and consistently. We recommend having around 200 tagged instances per entity. If you are using multilingual dataset and notice low performance in certain languages during the evaluation process, consider adding more data in this language to your training set and retrain the model.
- **Review the evaluation and improve the model:** After the model is trained successfully, check the model evaluation and confusion matrix. This helps you understand where your model made errors, and learn about entities that aren't performing well. Review the test set and view the predicted and tagged entities side by side. You will get a better idea of the model's performance and whether any changes in the schema or the tags are necessary.

General guidelines

The following guidelines can you help you understand and improve performance in custom NER.

Understand confidence scores

After you have tagged data and trained your model, you will need to deploy it for use in a production environment. Deploying a model means making it available for use via Analyze API to extract entities from a given text. The API returns a JSON object that contains a list of extracted entities each with its start index, length, and confidence score.

The confidence score is a decimal number between zero (0) and one (1), and it serves as an indicator of how confident the system is with its prediction. A higher value indicates that the service is more confident that the result is accurate. The returned score is directly affected by the data tagged when building the custom model. If the user's input at extraction is similar to the data used in training, higher scores can be expected, and more

If a certain entity is usually extracted with low confidence score, the developer might want to examine tagged data in the training set, add more instances for this entity, and retrain the model.

Set confidence score thresholds

Developers might choose to make decisions in the system based on the confidence score the system returns. They might choose to set a certain threshold where entities with confidence scores higher or lower than this threshold are treated differently. The confidence score threshold can be adjusted based on the developer's scenario.

Different scenarios call for different approaches. If critical actions will be based on the extracted entities, developers can choose to set a higher threshold to ensure the accuracy of the predicted entities. In this case, developers are expected to get fewer false positives but more false negatives (thus, a higher precision).

If in your scenario no critical decision will be taken based on the extracted entity, a developer can accept a lower threshold in order to extract all entities from the incoming text. In this case, you'll get more false positives but fewer false negatives (thus, a higher recall).

It's very important to evaluate your system with the set thresholds by using real data. The system processes this data in production to determine the effects on precision and recall.

Different training sessions and changes in evaluation

Retraining the same model without any changes in tagged data will result in the same model output, and therefore the same evaluation scores. However, if you add or remove tags, the model performance will change. The evaluation scores can then be compared with the previous version of the model, provided that there were no new files added during tagging. Adding new files, or training a different model with random set splits, will lead to different files in train and test sets. In this case, you can't make a direct comparison to other models, because performance is calculated on different splits for test sets.

Review incorrect predictions to improve performance

After you've trained your model, you can review model evaluation details to identify areas of improvement. You can also review the confusion matrix to identify entities that are often mistakenly predicted, and see if anything can be done to improve model performance. If you notice that a specific entity is often extracted as another entity, it's likely that these two entities have similar semantic characteristics. You might need to rethink your schema, or you can add more tagged examples to your dataset to help the model identify these entities.

After you have viewed evaluation details for your model, you should improve your model. This enables you to view the predicted and tagged entities side by side, to know what went wrong during model evaluation.

If you find that a complex entity is repeatedly not extracted, consider breaking it down to simpler entities for easier extraction. For example, instead of creating an address entity that extracts the whole address at once, consider extracting the zip code, street name, city, and state separately, in different entities.

If an entity is predicted while it was not tagged in your data, this means that you need to review your tags. Be sure that all instances of an entity are properly tagged in all files. Learn more about tagging your data.

Performance varies across features and languages

Custom NER gives you the option to use data in multiple languages. You can have multiple files in your dataset of different languages. Also, you can train your model in one language and use it to query text in other languages. If you want to use the multilingual option, you have to enable this during project creation. If you notice low scores in a certain language, consider adding more data in this language to your training set. For more information, see Supported languages.

Learn more about responsible AI

Microsoft AI principles

Microsoft responsible AI resources

Microsoft Azure Learning courses on responsible AI

Learn more about Custom Named Entity Recognition

What is Custom Named Entity Recognition?

Contact us

Give us feedback on this document

About this document

Published: 02/02/2022

Last updated: 02/02/2022