Member-only story

Understanding Named Entity Recognition: What Is It And How To Use It In Natural Language Processing?

D212digital · Follow

7 min read · Feb 13, 2023

Natural Language Processing (NLP) is a rapidly growing field of computer science, and Named Entity Recognition (NER) is one of its most powerful and versatile applications. This article dives into the details of NER — what it is, why it matters and how to use it in NLP projects. Get ready to learn everything you need to know about named entity recognition!

Introduction to Named Entity Recognition

Named Entity Recognition (NER) is a process of identifying and classifying named entities in text. Named entities can be people, locations, organizations, products, etc. NER is a part of Natural Language Processing (NLP) and can be used for various tasks such as information extraction, entity linking, and question answering.

One of the benefits of using NER is that it can help you structure unstructured data. For example, if you have a collection of news articles, you can use NER to automatically extract people, places, and organizations mentioned in the articles. This can be useful for tasks such as building knowledge graphs or performing market analysis.

NER can also be used for entity linking, which is the task of linking mentions of entities in text to entries in a knowledge base. For example, you could link mentions of “Barack Obama” in text to his Wikipedia entry. This can be useful for providing additional context about named entities or for disambiguating between multiple entities with the same name.

Finally, NER can be used as part of Question Answering systems. For example, if you ask a question like “Who was the President

What is Named Entity Recognition Technically?

Technically, the NER model typically goes through several stages of processing, including tokenization, part-of-speech tagging, and feature extraction. During the tokenization stage, the text is split into individual words or phrases. In the part-of-speech tagging stage, each word is assigned a part of speech such as noun, verb, or adjective. Finally, in the feature extraction stage, the model extracts relevant features such as the surrounding words, the position of the word within the sentence, and the part of speech.

Once the NER model has extracted the relevant features, it uses a machine learning algorithm to predict the named entity and its category. Common machine learning algorithms used for NER include conditional random fields, support vector machines, and neural networks.

There are many different NER systems available, both commercial and open-source. Some popular options include the Stanford Named Entity Recognizer (NER), Apache OpenNLP's NER models, and spaCy's NER model.

Applications of NER in Natural Language Processing

NER is used in a variety of applications within Natural Language Processing, including information extraction, question answering, and machine translation.

Information extraction is the task of extracting structured information from unstructured text. NER can be used to automatically extract people, locations, organizations, and other named entities from a text document. This information can then be used to populate a database or knowledge base.

Question answering systems are designed to answer questions posed in natural language. NER can be used to identify the entities mentioned in a question, which can then be used to retrieve relevant information from a knowledge base or database.

Machine translation is the task of automatically translating one natural language into another. NER can be used to identify named entities in the source language text, which can then be translated using a diction or glossary.

Steps of NER Process

Creating a NER (Named Entity Recognition) program can be a complex task that requires knowledge in Natural Language Processing, machine learning, and programming. Here are the basic steps to create a NER program:

1. Collect and annotate a dataset of text documents with named entities.
2. Preprocess the text data by tokenizing, normalizing, and vectorizing the text.
3. Split the dataset into training and testing sets.
4. Train a machine learning model (such as a Conditional Random Field or a Deep Learning model) on the annotated dataset.
5. Evaluate the model on the testing set and fine-tune the model if necessary.
6. Use the trained model to predict named entities on new, unseen text data.

There are many tools and libraries available that can help with each of these steps, such as NLTK, spaCy, and scikit-learn. Additionally, there are pre-trained models available that can be fine-tuned on your specific dataset to speed up the process.

Here is an example of how to perform NER using the spaCy library in Python:

```
import spacy
# Load the English language model
nlp = spacy.load("en_core_web_sm")

# Define a text for NER
text = "Apple is looking at buying U.K. startup for $1 billion"

# Process the text with the NLP model
doc = nlp(text)

# Print out named entities
for ent in doc.ents:
    print(ent.text, ent.label_)
```

This code loads the spaCy English language model, processes a text with the model, and then extracts named entities from the processed text using the doc.ents attribute. The output will display the named entities in the text, along with their labels. In the example text above, the output would be:

```
Apple ORG
U.K. GPE
$1 billion MONEY
```

where ORG refers to an organization, GPE refers to a geopolitical entity, and MONEY refers to a monetary value.

State-of-the-Art Techniques for NER

Named entity recognition is a difficult task due to the vast number of possible entities (people, locations, organizations, etc.) and the wide variety of ways in which they can be expressed in text. However, recent advances in machine learning have resulted in significant improvements in NER systems.

Some state-of-the-art techniques for NER include:

- Recurrent Neural Networks (RNNs): RNNs are a type of neural network that is well suited for modeling sequential data, such as sentences or paragraphs of text. RNNs can learn complex linguistic patterns and recognize named entities in text with high accuracy.
- Convolutional Neural Networks (CNNs): CNNs are a type of neural network that is well suited for modeling spatial data, such as images. CNNs have been shown to be effective at recognizing named entities in images and videos.
- Long Short-Term Memory Networks (LSTMs): LSTMs are a type of recurrent neural network

Examples of Current NER Systems and Use Cases

Some of the most popular current NER systems are:

- Google Cloud Natural Language API
- IBM Watson Natural Language Classifier

- Microsoft Azure Text Analytics
- Amazon Comprehend

Each of these has different strengths and weaknesses, so it's important to choose the right one for your specific needs. Here are some examples of how they can be used:

- Google Cloud Natural Language API can be used to analyze text from social media, websites, and documents. It offers support for multiple languages and can be used to extract entities like people, locations, organisations, and events.
- IBM Watson Natural Language Classifier can be used to classify texts into predefined categories. It offers good accuracy and supports a variety of languages.
- Microsoft Azure Text Analytics can be used to extract sentiment, key phrases, and entities from text. It offers support for multiple languages and is able to handle a large amount of data.
- Amazon Comprehend can be used to analyze text from social media, websites, and documents. It offers support for multiple languages and can be used to extract entities like people, locations, organisations, events, and Sentiment

Considerations for Working with NER

When developing a Named Entity Recognition (NER) system, there are several important considerations to keep in mind. Here are a few:

1. Data Quality: The quality and quantity of the data used to train the NER model are critical. Make sure the data is representative of the real-world

scenarios and is properly annotated.

2. Entity Types: Decide on the types of entities you want to recognize and label them consistently throughout your data. It is important to have a clear definition and understanding of what constitutes each entity type.
3. Feature Selection: Choose the right features that can help the model differentiate between entities and non-entities. These features could be based on the context, such as the words surrounding the entity, or the morphology of the entity itself.
4. Model Selection: Select the right type of machine learning algorithm for your task, such as a rule-based system, a statistical model, or a deep learning model. It is important to experiment with different models to find the one that best fits your data and use case.
5. Evaluation Metrics: Choose appropriate evaluation metrics to measure the performance of the NER system. Common evaluation metrics for NER include precision, recall, and F1 score.
6. Real-World Use: Finally, consider the real-world use case for your NER system. This includes factors such as the speed and efficiency of the system, its ability to handle new and unseen entities, and its ability to integrate with other systems or applications.

Finally, remember that NER can be resource-intensive, so you'll need to consider your processing power and runtime when deciding whether or not to use it.

Conclusion

I hope that this article has helped you better understand Named Entity Recognition and how it can be used within Natural Language Processing.

With the right tools in place, NER can save you time and give you reliable results while also allowing for more accurate predictions. Whether you're a business trying to analyse customer reviews quickly or an individual trying to get a grip on content from various sources, understanding how NER works is key to optimizing Natural Language Processing capabilities.

Thanks for reading.

David.

Python Programming | Naturallanguageprocessing | Named Entity Recognition | Deep Learning | Machine Learning

Written by D212digital

134 Followers

Self-motivated Developer — Specialising in NLP & NLU. Talks about Machine Learning, AI, Deep Learning

See more recommendations