



Search Medium

Write



# Coreference resolution with BERT-based Models

See how Bert-based models in Spark NLP can effortlessly resolve co-reference in your text data



Gursev Pirge · Following

Published in John Snow Labs · 7 min read · Mar 28



2



Photo by [David Kiriakidis](#) on [Unsplash](#)

*TL; DR: Coreference resolution is the task of identifying and linking all expressions within a text that refer to the same real-world entity, such as a person, object, or concept. Using Spark NLP, it is possible to perform many NLP applications, including text understanding, information extraction, and question answering.*

Coreference resolution is the task of **identifying and linking all expressions** within a text that refer to the same real-world entity, such as a person, object, or concept. In practical terms, coreference resolution involves analyzing a text and identifying **all expressions that refer to a specific entity**, such as “he,” “she,” “it,” or “they.” Once these expressions are identified, they are linked together to form a “coreference chain,” which represents all the different ways in which that entity is referred to in the text.

For example, given the sentence “John went to the store. He bought some groceries,” ; a coreference resolution model would identify that “John” and “He” both refer to the same entity and produce a cluster of coreferent mentions.

Coreference resolution is a complex task and it is used in a variety of applications, including information extraction, question answering, and machine translation. It is an important task in natural language processing (NLP), as it enables machines to accurately understand the meaning of a text and generate more human-like responses.

In this post, you will learn how to use Spark NLP to perform coreference resolution.

Let us start with a short Spark NLP introduction and then discuss the details of the coreference resolution techniques with some solid results.

## Introduction to Spark NLP

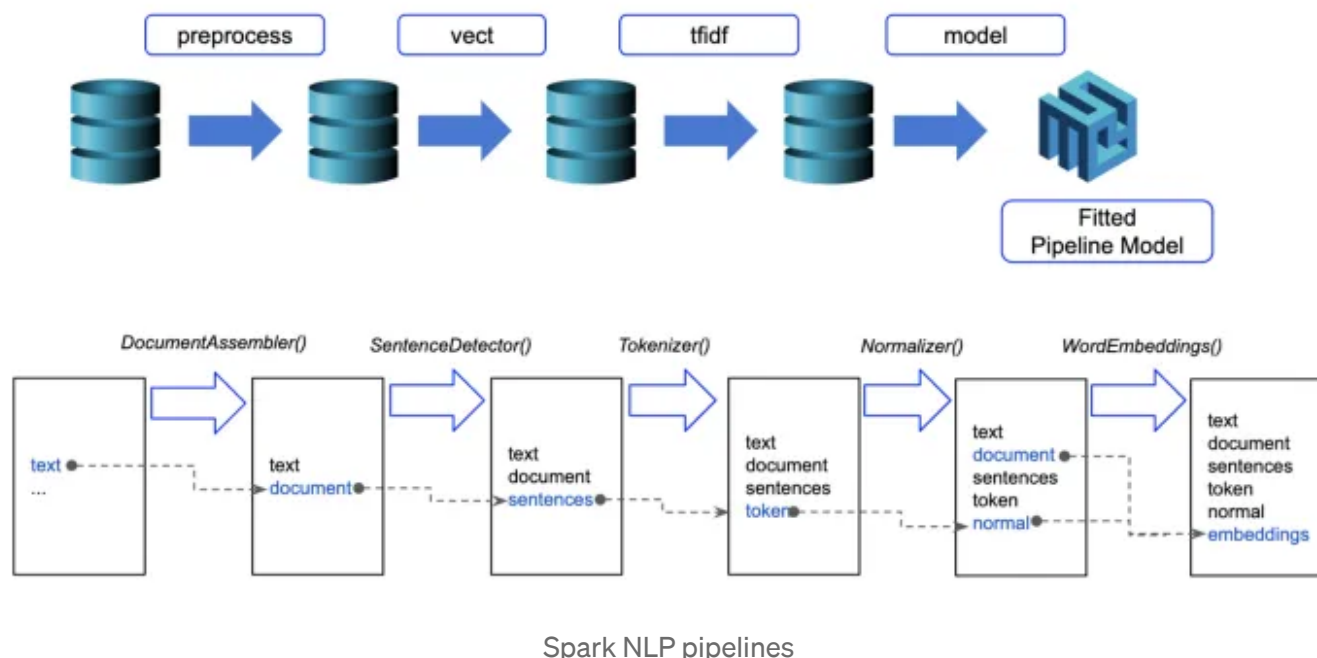
Spark NLP is an open-source library maintained by [John Snow Labs](#). It is built on top of Apache Spark and Spark ML and provides simple, performant & accurate NLP annotations for machine learning pipelines that can scale easily in a distributed environment.

Since its first release in July 2017, Spark NLP has grown in a full NLP tool, providing:

- A single unified solution for all your NLP needs
- Transfer learning and implementing the latest and greatest **SOTA** algorithms and models in NLP research
- The most widely used NLP library in industry (5 years in a row)
- The most scalable, accurate and fastest library in NLP history

Spark NLP comes with 14,500+ pretrained pipelines and models in more than 250+ languages. It supports most of the NLP tasks and provides modules that can be used seamlessly in a cluster.

Spark NLP processes the data using `Pipelines`, structure that contains all the steps to be run on the input data:



Each step contains an annotator that performs a specific task such as tokenization, normalization, and dependency parsing. Each annotator has input(s) annotation(s) and outputs new annotation.

An **annotator** in Spark NLP is a component that performs a specific NLP task on a text document and adds annotations to it. An annotator takes an input text document and produces an output document with additional metadata, which can be used for further processing or analysis. For example, a named entity recognizer annotator might identify and tag entities such as people, organizations, and locations in a text document, while a sentiment analysis annotator might classify the sentiment of the text as positive, negative, or neutral.

## Setup

To install Spark NLP in Python, simply use your favorite package manager (conda, pip, etc.). For example:

```
pip install spark-nlp  
pip install pyspark
```

For other installation options for different environments and machines, please check the [official documentation](#).

Then, simply import the library and start a Spark session:

```
import sparknlp  
  
# Start Spark Session  
spark = sparknlp.start()
```

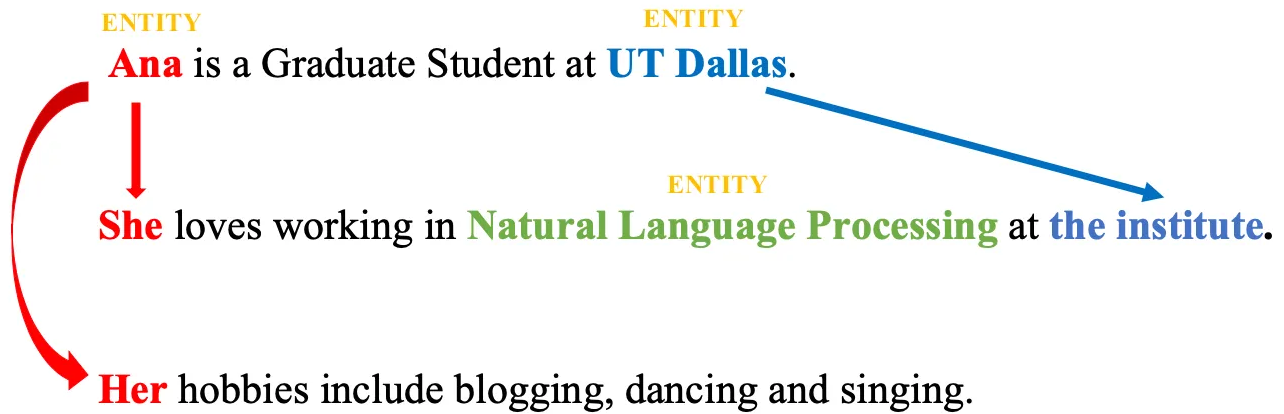
## Defining the Spark NLP Pipeline

The `SpanBertCoref` annotator expects `DOCUMENT` and `TOKEN` as input, and then will provide `DEPENDENCY` as output. Thus, we need the previous steps to generate those annotations that will be used as input to our annotator.

Spark NLP has the **pipeline approach** and the pipeline will include the necessary stages.

Please check [Unraveling Coreference Resolution in NLP!](#) for the examples and explanations below.

**First example** is for this text:



Here, “Ana”, “Natural Language Processing” and “UT Dallas” are possible entities.

“She” and “Her” are references to the entity “Ana” and “the institute” is a reference to the entity “UT Dallas”.

```
# Import the required modules and classes
from sparknlp.base import DocumentAssembler, Pipeline
from sparknlp.annotator import (
    SentenceDetector,
    Tokenizer,
    SpanBertCorefModel
)
import pyspark.sql.functions as F

# Step 1: Transforms raw texts to `document` annotation
document = DocumentAssembler() \
    .setInputCol("text") \
    .setOutputCol("document")

# Step 2: Sentence Detection
sentenceDetector = SentenceDetector() \
    .setInputCols("document") \
    .setOutputCol("sentences")

# Step 3: Tokenization
token = Tokenizer() \
    .setInputCols("sentences") \
    .setOutputCol("tokens")
```

```

.setContextChars(["(", ")", "?", "!", ".", ","])

# Step 4: Coreference Resolution
corefResolution= SpanBertCorefModel().pretrained("spanbert_base_coref")\
    .setInputCols(["sentences", "tokens"]) \
    .setOutputCol("corefs") \
    .setCaseSensitive(False)

# Define the pipeline
pipeline = Pipeline(stages=[document, sentenceDetector, token, corefResolution])

# Create the dataframe
data = spark.createDataFrame([["Ana is a Graduate Student at UT Dallas. She love

# Fit the dataframe to the pipeline to get the model
model = pipeline.fit(data)

```

Let us transform in order to get a prediction and determine the related entities:

```
model.transform(data).selectExpr("explode(corefs) AS coref").selectExpr("coref.r
```

token	metadata
ana	{head.sentence -> -1, head -> ROOT, head.begin -> -1, head.end -> -1, sentence -> 0}
she	{head.sentence -> 0, head -> ana, head.begin -> 0, head.end -> 2, sentence -> 1}
her	{head.sentence -> 0, head -> ana, head.begin -> 0, head.end -> 2, sentence -> 2}
ut dallas	{head.sentence -> -1, head -> ROOT, head.begin -> -1, head.end -> -1, sentence -> 0}
the institute	{head.sentence -> 0, head -> ut dallas, head.begin -> 29, head.end -> 37, sentence -> 1}

The dataframe shows the extracted entities and their metadata.

## One-liner alternative

In October 2022, John Snow Labs released the open-source `johnsnowlabs` library that contains all the company products, open-source and licensed, under one common library. This simplified the workflow especially for users working with more than one of the libraries (e.g., Spark NLP + Healthcare NLP). This new library is a wrapper on all John Snow Lab's libraries, and can be installed with `pip`:

```
pip install johnsnowlabs
```

Please check the [official documentation](#) for more examples and usage of this library. To run coreference resolution with one line of code, we can simply:

```
# Import the NLP module which contains Spark NLP and NLU libraries
from johnsnowlabs import nlp

sample_text= "Ana is a Graduate Student at UT Dallas. She loves working in
Natural Language Processing at the Institute. Her hobbies include blogging,
dancing and singing."

# Returns a pandas Data Frame, we select the desired columns
nlp.load('en.coreference.spanbert').predict(sample_text, output_level='sentence')
```

	coref	coref_head	coref_head_begin	coref_head_end	coref_head_origin_sentence	coref_origin_sentence	sentence
0	[Ana, She, Her]	[ROOT, Ana, Ana]	[-1, 0, 0]	[-1, 2, 2]	[-1, 0, 0]	[0, 1, 2]	Ana is a Graduate Student at UT Dallas.
0	[Ana, She, Her]	[ROOT, Ana, Ana]	[-1, 0, 0]	[-1, 2, 2]	[-1, 0, 0]	[0, 1, 2]	She loves working in Natural Language Processi...
0	[Ana, She, Her]	[ROOT, Ana, Ana]	[-1, 0, 0]	[-1, 2, 2]	[-1, 0, 0]	[0, 1, 2]	Her hobbies include blogging, dancing and sing...

The resulting dataframe produced by the one-liner model.

The reason for the difference between the one-liner's results and the previous results is, here the model's case sensitivity was ON and did not



detect ‘the Institute’.

The one-liner is based on default models for each NLP task. Depending on your requirements, you may want to use the one-liner for simplicity or customizing the pipeline to choose specific models that fit your needs.

NOTE: when using only the `johnsnowlabs` library, make sure you initialize the spark session with the configuration you have available. Since some of the libraries are licensed, you may need to set the path to your license file. If you are only using the open-source library, you can start the session with `spark = nlp.start(nlp=False)`. The default parameters for the start function includes using the licensed Healthcare NLP library with `nlp=True`, but we can set that to `False` and use all the resources of the open-source libraries such as Spark NLP, Spark NLP Display, and NLU.

**Second example** is much longer and more complicated.

The paragraph involves a person and a company’s names mentioned in multiple ways and the model was able to detect them all.

"I had no idea I was getting in so deep," says Mr. Kaye, who founded Justin in 1982. Mr. Kaye had sold Capetronic Inc., a Taiwan electronics Maker, and retired, only to find he was bored. With Justin, he began selling toys and electronics made mostly in Hong Kong, beginning with Mickey Mouse radios. The company has grown -- to about 40 employees, from four initially, Mr. Kaye says. Justin has been profitable since 1986, adds the official, who shares [his] office... (nw/wsj/2418)

We will use the same model, but feed the text above:

```
data_2 = spark.createDataFrame([[""" "I had no idea I was getting in so deep," s
model = pipeline.fit(data_2)

model.transform(data_2).selectExpr("explode(corefs) AS coref").selectExpr("coref
```

token	metadata
I	{head.sentence -> -1, head -> ROOT, head.begin -> -1, head.end -> -1, sentence -> 0}
I	{head.sentence -> 0, head -> I, head.begin -> 2, head.end -> 2, sentence -> 0}
Mr . Kaye , who founded Justin in 1982	{head.sentence -> 0, head -> I, head.begin -> 2, head.end -> 2, sentence -> 0}
Mr . Kaye	{head.sentence -> 0, head -> I, head.begin -> 2, head.end -> 2, sentence -> 1}
he	{head.sentence -> 0, head -> I, head.begin -> 2, head.end -> 2, sentence -> 1}
he	{head.sentence -> 0, head -> I, head.begin -> 2, head.end -> 2, sentence -> 2}
Mr Kaye	{head.sentence -> 0, head -> I, head.begin -> 2, head.end -> 2, sentence -> 3}
Justin	{head.sentence -> -1, head -> ROOT, head.begin -> -1, head.end -> -1, sentence -> 0}
Justin	{head.sentence -> 0, head -> Justin, head.begin -> 70, head.end -> 75, sentence -> 2}
The company	{head.sentence -> 0, head -> Justin, head.begin -> 70, head.end -> 75, sentence -> 3}
Justin	{head.sentence -> 0, head -> Justin, head.begin -> 70, head.end -> 75, sentence -> 4}

The dataframe shows the extracted entities and their metadata.

For additional information, please consult the following references:

- Documentation : [SpanBertCoref](#)
- Python Docs : [SpanBertCoref](#)
- Scala Docs : [SpanBertCoref](#)
- For extended examples of usage, see the [Spark NLP Workshop repository](#).
- Academic Reference Paper: [SpanBERT: Improving Pre-training by Representing and Predicting Spans](#)
- John Snow Labs [SpanBertCoref Model](#)

## Conclusion

SpanBertCoref annotator of Spark NLP is a coreference resolution model based on [SpanBert](#), which identifies expressions which refer to the same entity in a text.

Coreference resolution models produce a mapping of all the expressions in a text that refer to the same real-world entity. Coreference resolution can be a challenging task, particularly in cases where there are multiple potential referents for a given expression, or when the referent is implicit or ambiguous.

## More from the list: "NLP"

Curated by Himanshu Birla



Jon Gi... in Towards Data ...

### Characteristics of Word Embeddings

★ · 11 min read · Sep 4, 2021



Jon Gi... in Towards Data ...

### The Word2vec Hyperparameters

★ · 6 min read · Sep 3, 2021



Jon Gi... in

### The Word2vec

★ · 15 min read



[View list](#)



## Written by Gursev Pirge


98 Followers · Writer for John Snow Labs

Following



## More from Gursev Pirge and John Snow Labs




 Gursev Pirge

## Performance Comparison of Multi-Class Classification Algorithms

This article comprises the application and comparison of supervised multi-class...

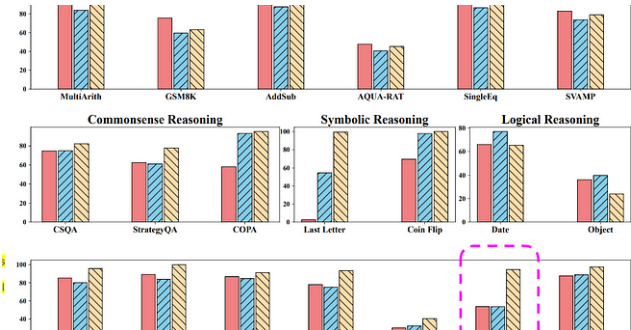
7 min read · Dec 29, 2020


 149









 Veysel Kocaman

 in John Snow Labs

## Can We Trust ChatGPT and LLMs in Information Retrieval Tasks ?

LLMs are great productivity tool. They don't lie, and they don't tell the truth either. They...

6 min read · Apr 21


 72

 1





Category	Company	Found	Funding (06/2023)	LLMs	License	Serving
Leading LLM Companies	OpenAI	2015	\$11.3B	GPT-3.5, GPT-4 (ChatGPT)	Licensed	API
	Anthropic	2021	\$1.5B	Claude	Licensed	API
	Cohere	2019	\$435M	Command	Licensed	API
	Cerebras	2015	\$715M	Cerebras-GPT	Open Source	Local
	Alph Alpha	2019	\$31M	Luminous	Licensed	API
	Ai21 Labs	2017	\$118.5M	Jurassic-2	Licensed	API
	Google	1998	-	PALM-2 (Bard)	Licensed	API
Big Commercial Players	Meta	2004	-	LLaMA	Open Source (nonCommercial)	Local
	Databricks	2013	\$3.5B	Dolly	Open Source	Local
	HuggingFace	2017	\$160M	HuggingChat, Bloom, StarCoder	Open Source	Local
Others	StabilityAI	2019	\$114M	StableLM	Open Source	Local
	EleutherAI	2020	-	GPT-J, GPT-NeoX	Open Source	Local
	MosaicML	2021	\$64M	MPT-7B	Open Source	Local
	H2O.ai	2011	\$251M	-	-	-
	LMSYS	2011	-	Vicuna	Open Source	Local

 Veysel Kocaman

 in John Snow Labs

## Beyond OpenAI in Commercial LLM Landscape

Exploring the Innovators and Challengers in the Commercial LLM Landscape beyond...

23 min read · Jul 21

 134









 Gursev Pirge

 in John Snow Labs


## Named Entity Recognition (NER) with Python at Scale


Using Spark NLP in Python to identify named entities in texts at scale..

11 min read · May 13









[See all from Gursev Pirge](#)[See all from John Snow Labs](#)

## Recommended from Medium



Guillaume Michel in Kensho Blog

### Kensho Classify: The Solution to Common Challenges of Text...

Text classification is widely used in many industries and often serves as a pillar for mo...

4 min read · Sep 7



7



essed his claim to be the greatest player of all time after another performanc

```
is:
ted: {entity['word']], Entity Label: {entity['entity_group']], Confidence sco
```

Jokovic, Entity Label: PER, Confidence score: 0.9974638223648071  
Open, Entity Label: MISC, Confidence score: 0.9965554475784302  
Entity Label: LOC, Confidence score: 0.9993627667427063  
ntity Label: MISC, Confidence score: 0.9981368780136108  
Nadal, Entity Label: PER, Confidence score: 0.9987477660179138  
Entity Label: MISC, Confidence score: 0.9151148796081543



Seffa B

### Named Entity Recognition with Transformers: Extracting Metadata

3 min read · Jun 12

## Lists



### Natural Language Processing

669 stories · 283 saves



### The New Chatbots: ChatGPT, Bard, and Beyond

13 stories · 133 saves





## New\_Reading\_List

174 stories · 133 saves



## Icon Design

30 stories · 107 saves



Zahrizhal Ali

## Crafting Your Custom Text-to-Speech Model.

Welcome to a wild journey where code and comedy collide! In this blog, we're going to...

11 min read · Jun 2



200



2



FHIRFLY

## Transforming Healthcare: ClinicalBERT and UMLSBERT's Rol...

The healthcare sector has always been an intricate orchestra of challenges, especially i...

3 min read · Jul 1



1



Build	Stable (2.0.0)		Preview (Nightly)	
	Linux	Mac	Windows	
	Conda	Pip	LibTorch	Source
	Python		C++/Java	
Platform	CUDA 11.7	CUDA 11.8	ROCm 5.4.2	CPU
Command:	<pre>pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu117</pre>			



Bluetick Consultants

## The Future of Automatic Speech Recognition—Whisper

Introduction

4 min read · Jun 2

Artificial Intelligence (AI [ORG](#)), an ever-evolving field, has witnessed remarkable growth since its inception. Dating back to [the Dartmouth Conference](#) [ORG](#) in [1956](#) [DATE](#), [AI](#) [ORG](#) has emerged as a multidisciplinary domain encompassing machine learning, natural language processing ([NLP](#) [ORG](#)), computer vision, and robotics. Recent breakthroughs, like the introduction of deep learning techniques in [the early 2010s](#) [DATE](#), have accelerated [AI](#) [ORG](#) advancements. Tech giants like [Google](#) [ORG](#), [IBM](#) [ORG](#), and [Microsoft](#) [ORG](#) have invested heavily in [AI](#) [ORG](#) research and development. Significant milestones include the landmark victory of [IBM](#) [ORG](#)'s [Deep Blue](#) [ORG](#) over [Garry Kasparov](#) [PERSON](#) in [1997](#) [DATE](#) and the emergence of voice assistants like [Apple](#) [ORG](#)'s Siri in [2011](#) [DATE](#). [AI](#) [ORG](#) continues to shape industries across healthcare, finance, and transportation, fueling innovation and transforming the way we live and work.



Sahithi Reddy

## Named Entity Recognition using Python and Spacy.

NER, short for Named Entity Recognition, is an NLP method that detects and categorizes...

4 min read · Jun 11



24



See more recommendations