How to make an effective coreference resolution model

How to improve upon ready-to-use coreference resolution libraries

Marta Maślankowska · Following

Published in Towards Data Science · 9 min read · Apr 6, 2021

Written by Marta Maślankowska and Paweł Mielniczuk.

Photo by Dariusz Sankowski on Unsplash

Introduction

In this article, we present how to improve AllenNLP's coreference resolution model in order to achieve a more coherent output. We also introduce a couple of ensemble strategies to make use of both Huggingface and AllenNLP models at the same time.

In short, coreference resolution (CR) is an NLP task that aims to replace all ambiguous words in a sentence so that we get a text that doesn't need any extra context to be understood. If you need a refresher on some basic concepts, refer to our introductory article.

Here, we focus mainly on **improving how the libraries resolve found clusters**. If you are interested in a detailed explanation of the most common libraries for CR (e.g. what *AllenNLP* or *Huggingface* are), and our motivations feel free to check it out.

Ready-to-use yet incomplete

Both Huggingface and AllenNLP coreference resolution models would be a great addition to many projects. However, we've found several drawbacks (described in detail in the previous article) that made us doubt whether we truly wanted to implement those libraries in our system. The most substantial problem isn't the inability to find acceptable clusters but the last step of the whole process — resolving coreferences in order to obtain an unambiguous text.

This made us think that there might be something we can do about it. We've decided to implement several minor changes that result in a significant improvement of the final text. As AllenNLP seems to find more clusters, which are often better, we settled on a solution that focuses more on this model.

Overview of the proposed improvements

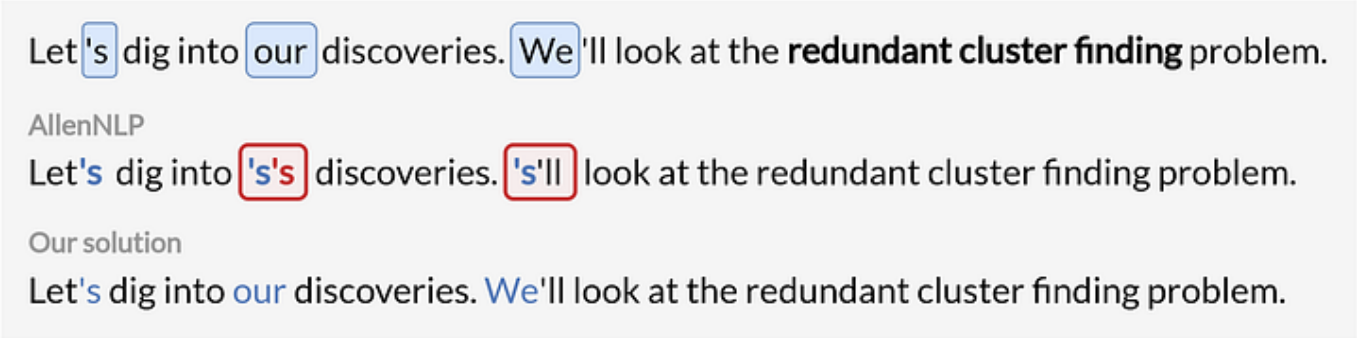
We've decided to consider AllenNLP as our main model, and utilize Huggingface as more of a reference while using it mostly as a refinement to AllenNLP output. Our solution consists of:

1. improving AllenNLP's method of replacing coreferences, based on the clusters already obtained by the model,
2. introducing several strategies that combine both models' outputs (clusters) into a single enhanced result.

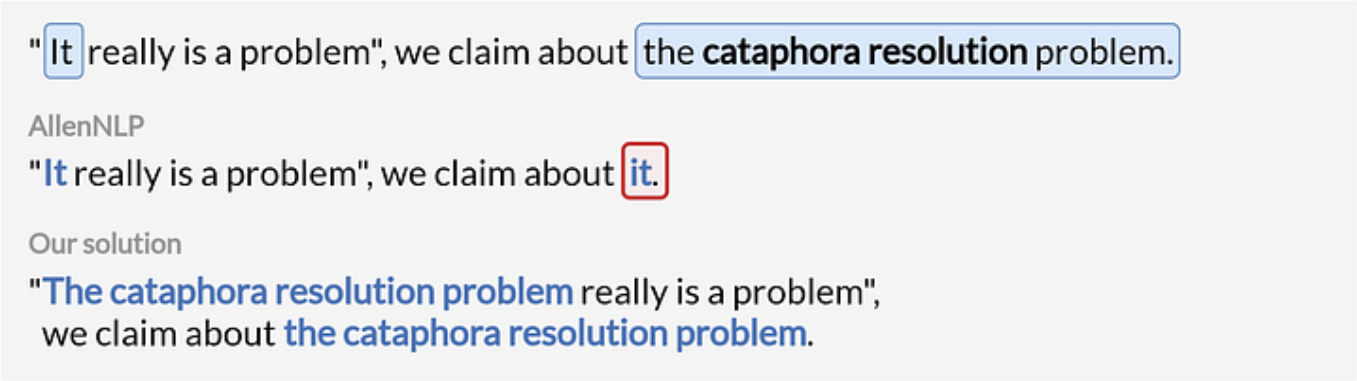
To refine the replacement of coreferences there are several problematic areas that can be somewhat easily improved upon:

- lack of a meaningful mention in a cluster that could become its head (a span with which we replace all other mentions in a given cluster)
- treating the first span as a head of a cluster (which is problematic especially with cataphora),
- various complications, and nonsense outputs as a result of nested mentions.

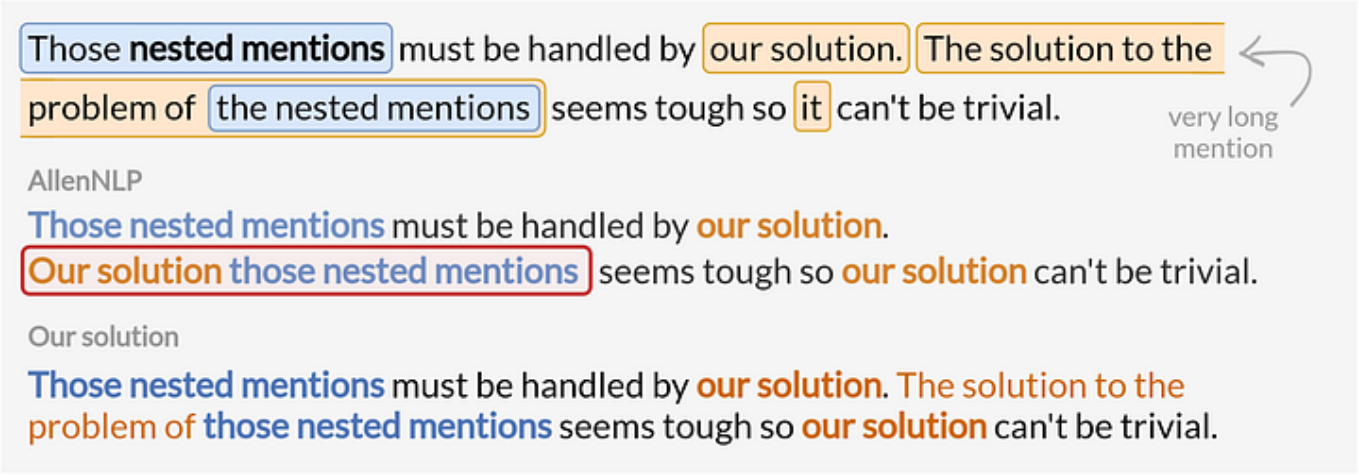
All of the above problems are represented below with examples:



Text that lacks a meaningful mention (e.g. noun-phrase) in the cluster.



Text with cataphora — a pronoun precedes a noun phrase.



Text with nested coreferent mentions.

We propose the following solutions to those problems:

- do not consider a cluster at all if it doesn't contain any noun phrases
- consider the first noun phrase (not the first mention) in the cluster as its head
- resolve only the inner span in nested coreferent mentions

In the next section, these approaches are explained in detail — both AllenNLP improvements and strategies for combining models. Also, whether you'd like to go further in-depth or just skip the details, all code from the next chapter is available on our NeuroSYS GitHub repository.

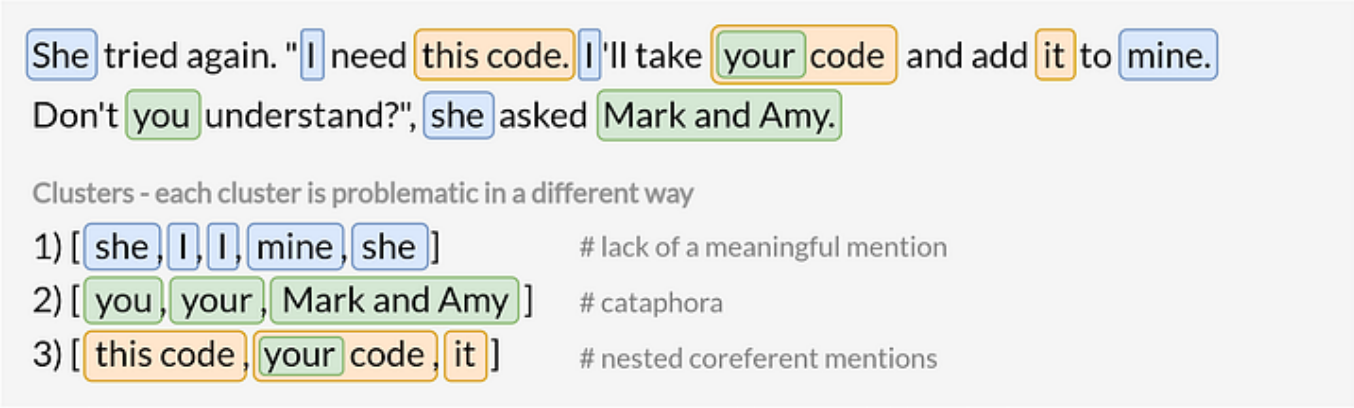
Improvements in-depth

Thanks to Huggingface being based on spaCy it's effortless to use and provides multiple additional functionalities. However, it's much more complicated to modify because spaCy provides many mechanisms that prohibit you from accessing or changing the underlying implementation.

What's more, in addition to AllenNLP acquiring a higher number of valid clusters compared to Huggingface, we've also found the former to be much easier to modify.

In order to modify AllenNLP's behavior, we focus on the *coref_resolved(text)* method. It iterates through all clusters and replaces all spans in every cluster with the first found mention. Our improvements concern only this function and the nested methods inside it.

Below is an example of a short text that contains all three above-mentioned problems we're trying to solve, on which we'll focus for now.



Redundant clusters

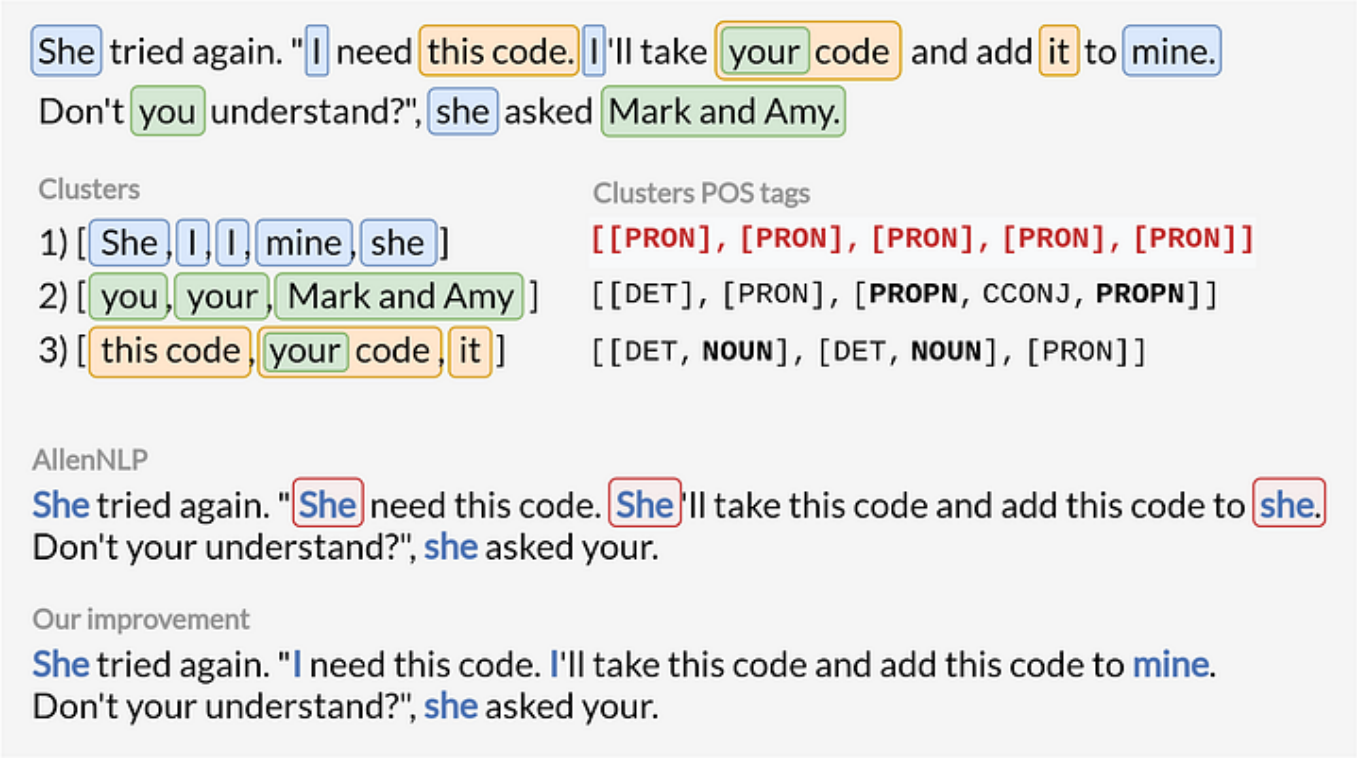
As we try to keep our solution straightforward, we've decided to define a meaningful mention simply as any noun phrase.

For each cluster, we acquire indices of spans containing noun phrases (we use them also in our following improvement). The simplest way to verify if a token is a noun or not is using spaCy! It turns out that underneath AllenNLP also uses a spaCy language model, but only to tokenize an input text.

The nested method that interests us here the most is *replace_corefs(spacy_doc, clusters)*. Not only does it make use of the spaCy Doc object but also contains all the logic necessary to implement our improvements. It looks something like this:

From the spaCy documentation, we know that nouns are represented by two part-of-speech (POS) tags: *NOUN* and *PROPN*. We need to check whether a POS tag of every span in a cluster (actually every token in each span) is one of these two.

Below is shown how this small piece of code improves coreference replacement. For now, we focus mainly on the first cluster, as that's the one that shows the current problem.



Solving cataphora problem

Many coreference resolution models, such as Huggingface, have serious problems with detecting cataphora as it's a rather rare occurrence. On one hand, we might not be interested in resolving such unusual cases that cause even more problems. On the other hand, depending on the text, we may lose more or less information, if we neglect them.

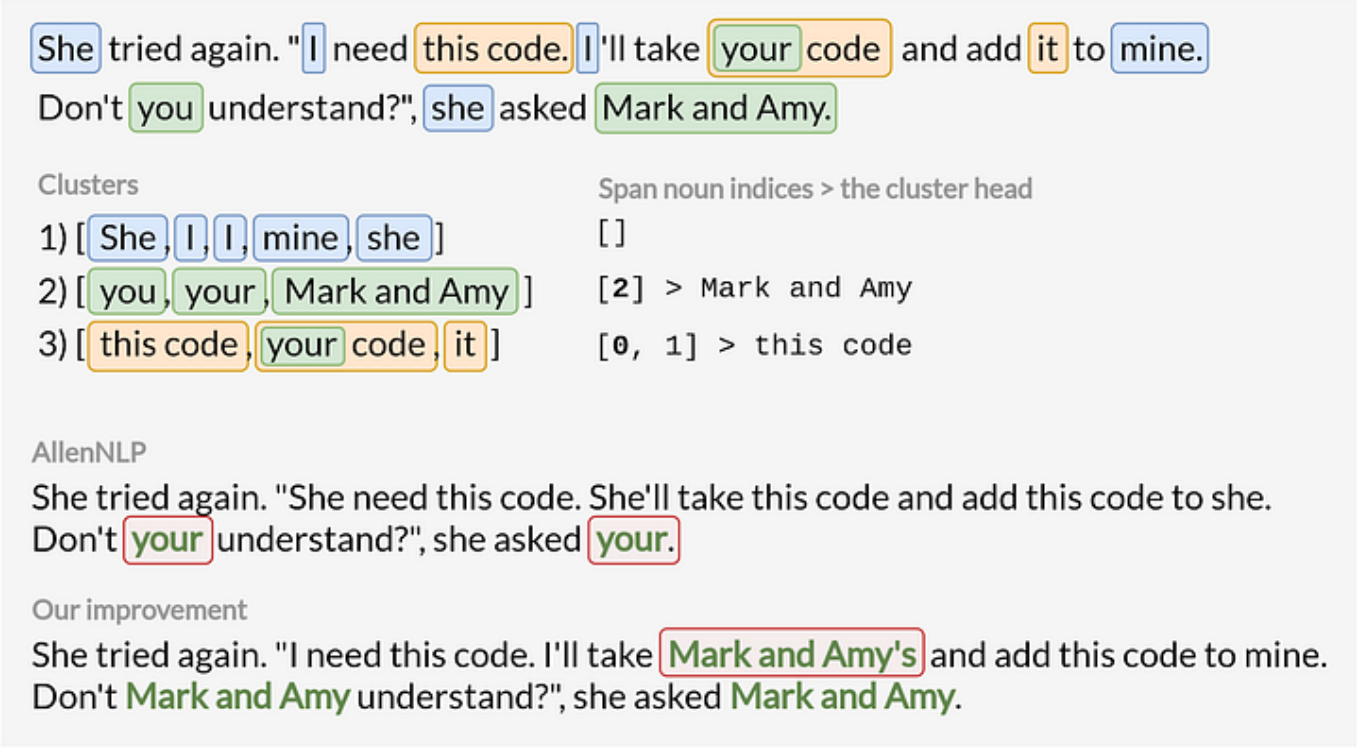
AllenNLP detects cataphora but since it treats the first mention in a cluster as its head, it leads to further errors. That's because the cataphor (e.g. a pronoun) precedes the postcedent (e.g. a noun phrase) so a meaningless span becomes a cluster's head.

To avoid this we've proposed to consider the first noun phrase in the cluster (not just any mention) as its head, replacing all preceding and following spans with it. This solution is trivial and though we can see the advantages of other, more sophisticated ideas, our one seems to be effective enough for most cases.

Let's take a closer look at a few key lines in the AllenNLP's *replace_corefs* method.

To make our solution work, we need to redefine the *mention_span* variable (the head of the cluster) so it represents the first found noun phrase. To achieve this, we use our *noun_indices* list — its first element is the one we want.

Let’s see how it improves the output. This time we focus on the second cluster. Now there is no information loss! However, since AllenNLP returns results depending on the order of the clusters found there is also a small mistake in our version here. Don’t worry though, we are going to fix it in the next section.



Nested mentions

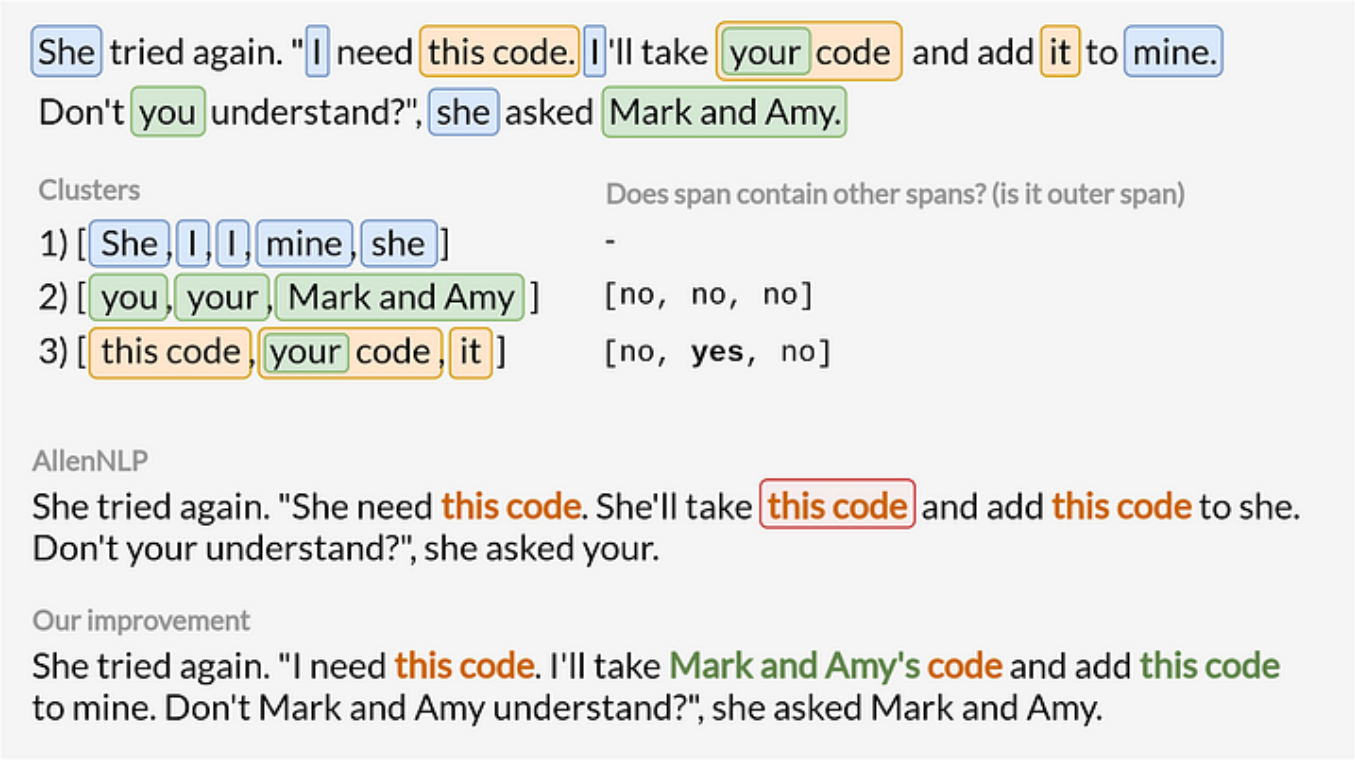
The last improvement concerns spans that are composed of more than one mention. As we've shown in our previous article there are several strategies that can be employed to solve this problem, none of which are flawless:

1. replacing a whole, nested mention with only the outer span → we lose information
2. replacing both inner and outer span → doesn't work in many cases, leads to different results depending on the found clusters order
3. replacing only the inner span → works for the majority of texts however, some replacements result in gibberish sentences
4. omitting nested mentions; not replacing spans at all → we do not gain information that the coreference resolution model was supposed to provide us in the first place but we have 100% certainty that the text is grammatically correct

We believe that the 3rd strategy — replacing only the inner span — is a good compromise between information gain and the number of possible errors in the final text. It also comes with one additional feature — now regardless of the clusters ordering the output will always be the same.

To implement it, we just need to detect outer spans and omit them. To do so, we simply have to check if particular span indices contain any other mentions.

In the end, we acquire a text that has all of our improvements. We're aware that it could've been resolved even better but we find this trade-off between the solution's simplicity and validity of the resolved text just right.



As shown in the image above, we can obtain better results while enhancing only how the found clusters are resolved.

Ensemble strategies

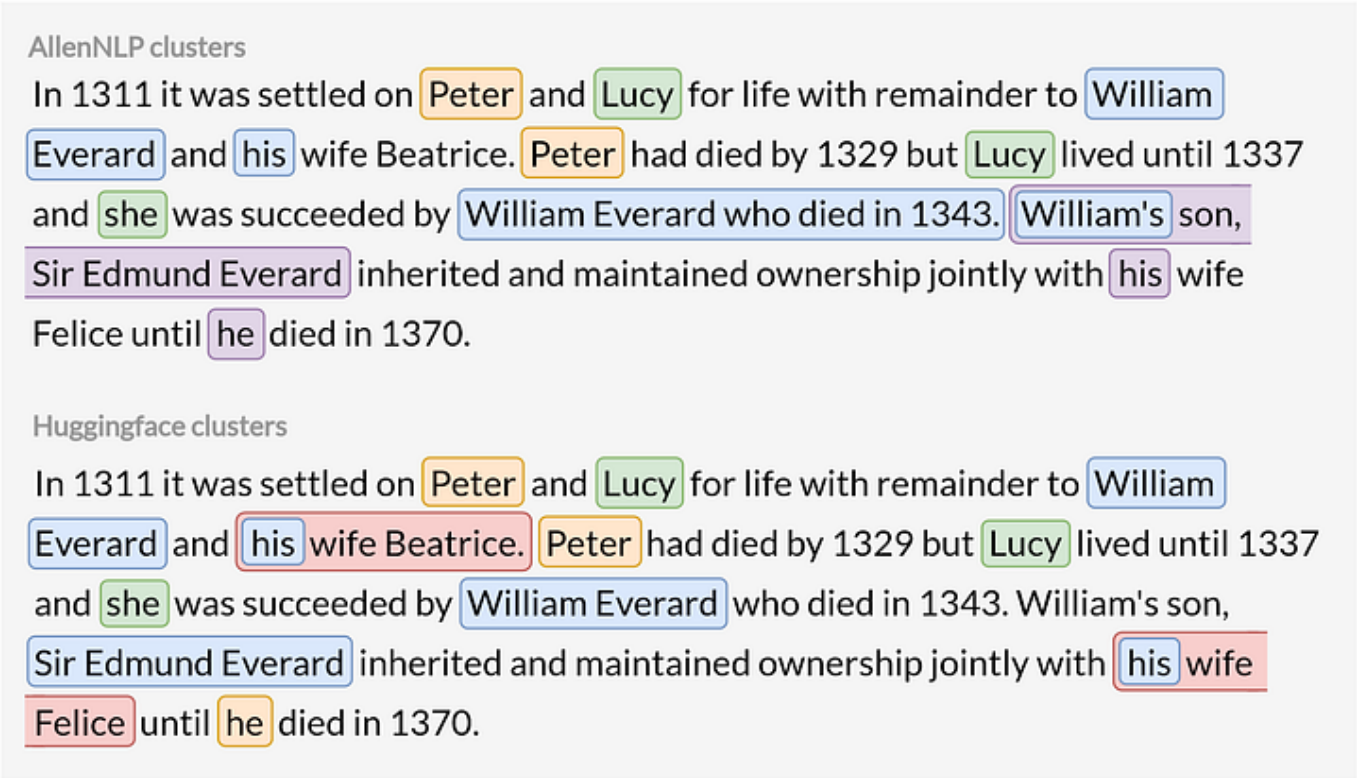
After all the improvements, we can now move to intersection strategies — ideas on how to combine both AllenNLP and Huggingface clusters. As we've mentioned before, we believe that AllenNLP produces notably better clusters, yet it isn't perfect. To acquire the highest possible confidence level about the final clusters, without any fine-tuning, we propose several ways of merging both models outputs:

- strict — leave only those clusters which are exactly the same in both Huggingface and AllenNLP model outputs (intersection of clusters)
- partial — leave only those spans which are exactly the same in both Huggingface and AllenNLP (intersection of spans)

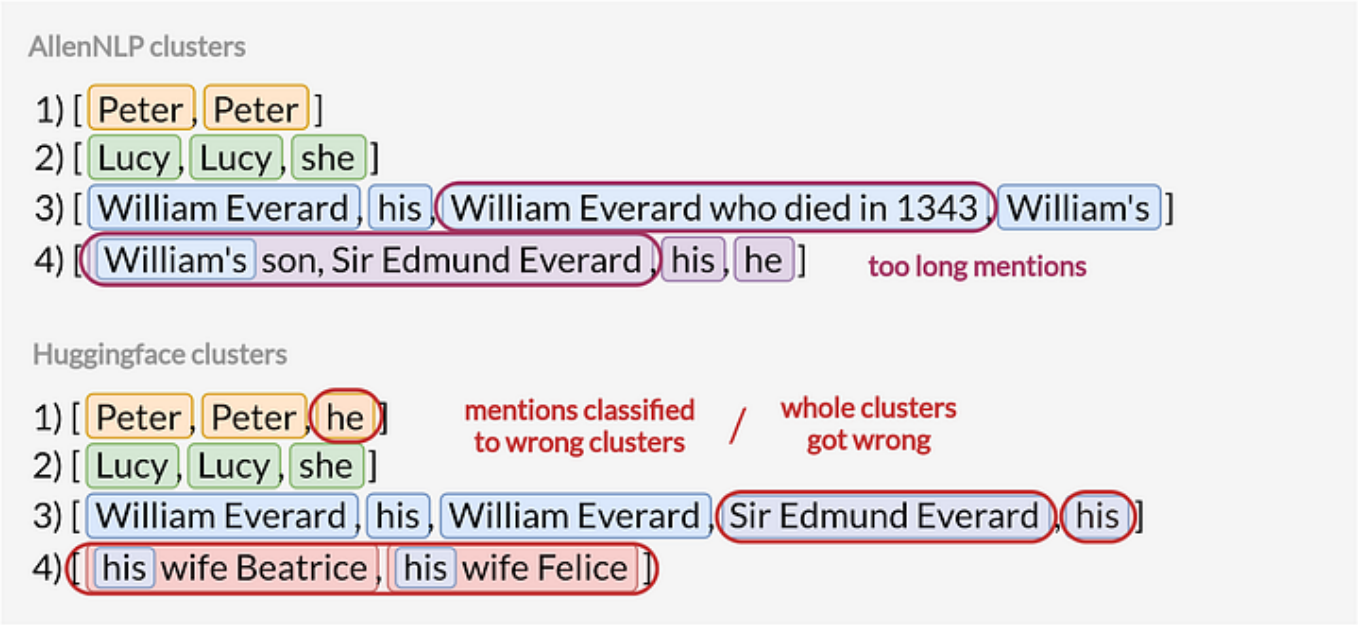
- fuzzy — leave all spans that are even partially the same (that overlap) in Huggingface and AllenNLP, but prioritize the shorter one

As AllenNLP is usually better we treat it as our base, so in dubious cases, we construct the output solely from its findings. As the code isn't as short as in the case of discussed improvements we provide a detailed Jupyter Notebook, and here just peek at the strategies' outputs.

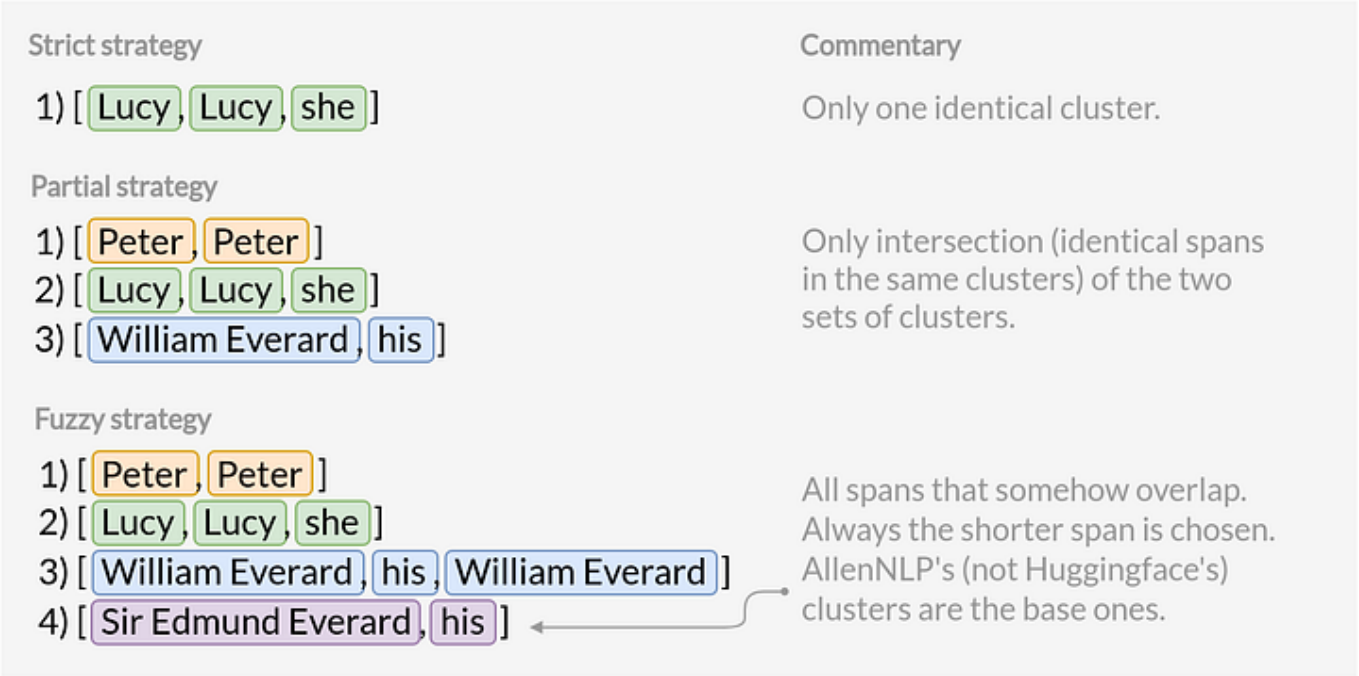
The following example is taken from the GAP dataset, which we've previously explained in detail.



To efficiently compare proposed intersection strategies, it's easier to think of models' outputs as sets of clusters, like the ones illustrated below. It's worth noting that AllenNLP and Huggingface not only found slightly different mentions but also whole clusters.



Clusters found by both libraries — AllenNLP and Huggingface.



Clusters obtained by each intersection strategy — an ensemble of AllenNLP and Huggingface.

We lean mostly towards the fuzzy strategy. It provides a higher certainty than the output of a single model while also providing the largest information gain. All strategies have their pros and cons so it's best to experiment and see what works best for your dataset.

Let's look at the final result — example with resolved coreferences, including our previous improvements:

Fuzzy strategy - resolved coreferences (with the improvements)

In 1311 it was settled on Peter and Lucy for life with remainder to William Everard and William Everard's wife Beatrice. Peter had died by 1329 but Lucy lived until 1337 and Lucy was succeeded by William Everard who died in 1343. William's son, Sir Edmund Everard inherited and maintained ownership jointly with Sir Edmund Everard's wife Felice until he died in 1370.

Even though the final text doesn't sound very natural, we must remember that the aim of coreference resolution is often disambiguation for language models. As a result, they can have a better understanding of the input and are able to produce more appropriate embeddings. In this instance, we lack only a single piece of information — who died in 1370 — while having obtained many correct substitutions and got rid of overly long mentions.

Summary

Throughout our articles, we've tackled a major issue in NLP — coreference resolution. We've explained what CR is, introduced the most common libraries as well as the problems they come with, and now shown **how to improve upon the available solutions**.

We've tried to clarify everything we could have with multiple images and various examples. Both basic usage, as well as our modifications, are available on our NeuroSYS GitHub.

Hopefully, by now you are familiar with coreference resolution and can easily adapt our proposed solutions to your project!

NLP Data Science Deep Learning Machine Learning Artificial Intelligence

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 Word2ve

· 15 min rea

View list

Written by Marta Maślankowska

30 Followers · Writer for Towards Data Science

Following

More from Marta Maślankowska and Towards Data Science

Marta Maślankowska in Towards Data Science

Most popular coreference resolution frameworks

What's the best coreference resolution model? What to look for when choosing one?

9 min read · Jan 22, 2021

27 2

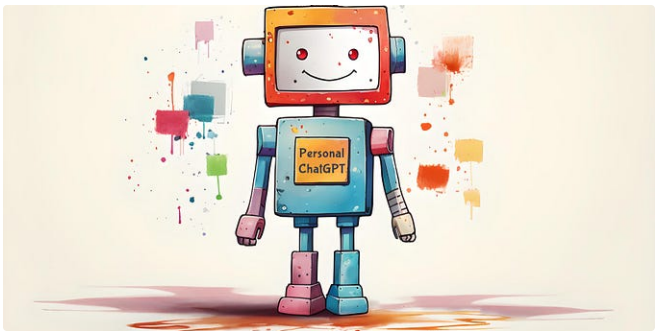
Antonis Makropoulos in Towards Data Science

How to Build a Multi-GPU System for Deep Learning in 2023

This story provides a guide on how to build a multi-GPU system for deep learning and...

10 min read · Sep 17

549 11



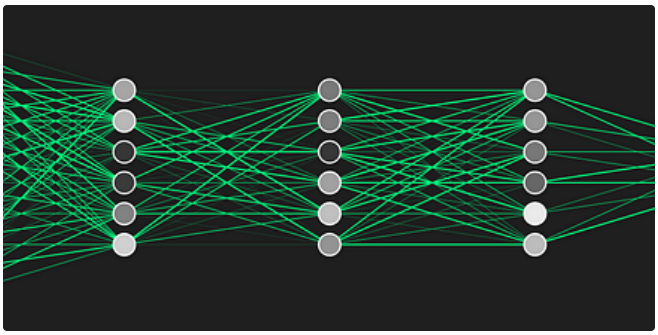
Robert A. Gonsalves in Towards Data Science

Your Own Personal ChatGPT

How you can fine-tune OpenAI's GPT-3.5 Turbo model to perform new tasks using you...

· 15 min read · Sep 8

595 7



Callum Bruce in Towards Data Science

How to Program a Neural Network

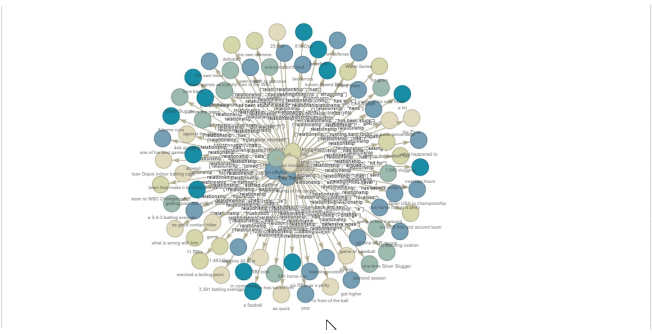
A step-by-step guide to implementing a neural network from scratch

· 14 min read · Sep 24

470 1

See all from Marta Maślankowska | See all from Towards Data Science

Recommended from Medium



Wenqi Glantz in Better Programming

7 Query Strategies for Navigating Knowledge Graphs With...

Exploring NebulaGraph RAG Pipeline with the Philadelphia Phillies

✨ · 17 min read · 4 days ago

501 4

Dixn Jakindah

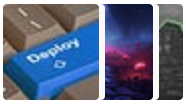
Top P, Temperature and Other Parameters

Large Language Models(LLMs) are essential tools in natural language processing (NLP)...

3 min read · May 18

7

Lists



Predictive Modeling w/ Python

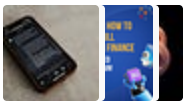
20 stories · 452 saves

Natural Language Processing

669 stories · 283 saves

Practical Guides to Machine Learning

10 stories · 519 saves



ChatGPT prompts

24 stories · 459 saves

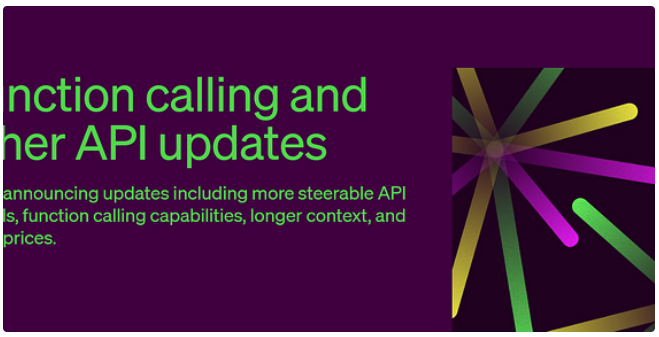
Ritesh

Speaker diarization using Whisper ASR and Pyannote

What is speaker diarization?

8 min read · Jul 22

4 1



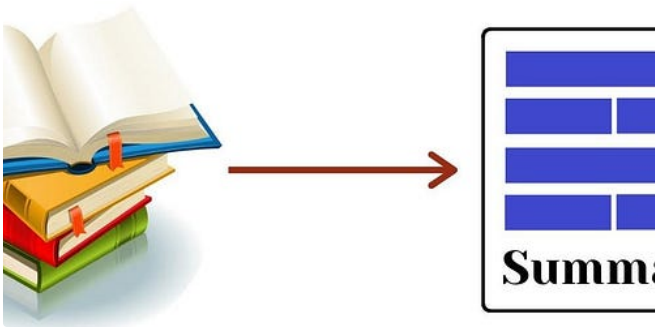
Benjamin De Kraker

Fun with Functions: (Almost) Everything About GPT API...

OpenAI has announced support for "Functions," a new feature within the GPT-3.5...

11 min read · Jun 14

5 1



Fabiano Falcão

Metrics for evaluating summarization of texts performe...

Text summarization performed by Transformers is one of the most fascinating...

Farzad Mahmoodinobar in Towards Data Science

Grammatical Error Correction with Machine Learning—Overview an...

Using Grammatical Error Correction: Tag, Not Rewrite (GECTor)

7 min read · Apr 23

4

· 8 min read · Nov 17, 2022

42 1

See more recommendations