Metrics

Let's go over the metrics to evaluate the performance of the entity linking system.

We'll cover the following

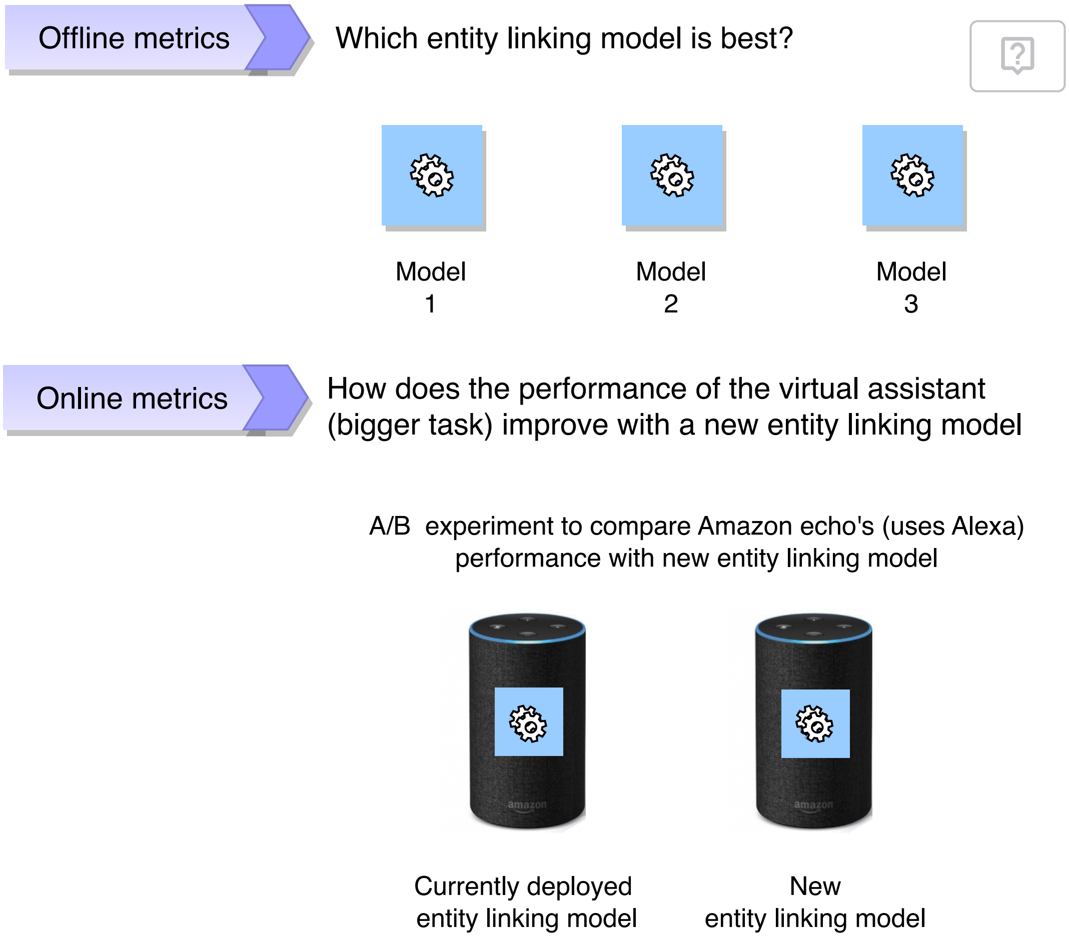
In the previous lesson, we talked about various applications of *named entity linking* system and how it can be used as a component in bigger tasks/systems such as a virtual assistant system. Therefore, we require metrics that:

1. Compare different entity linking models based on their performance.

   This will be catered to by offline metrics.

2. Measure the performance of the bigger task when a particular model for entity linking is used.

   This will be catered to by online metrics.


Offline metrics
Which entity linking model is best?
Model 1
Model 2
Model 3
Online metrics
How does the performance of the virtual assistant (bigger task) improve with a new entity linking model
A/B experiment to compare Amazon echo's (uses Alexa) performance with new entity linking model
Currently deployed entity linking model
New entity linking model

Offline vs online metrics

> 📝 Offline metrics will be aimed at improving/measuring the performance of the entity linking component. Online metrics will be aimed at improving/measuring the performance of the larger system by using a certain entity linking model as its component.

Offline metrics#

The named-entity linking component is made of two layers, as discussed previously:

1. Named entity recognition
2. Disambiguation

We will first look at offline metrics for each of the layers individually and will then discuss a good offline metric to measure the overall entity linking system.

Named entity recognition#

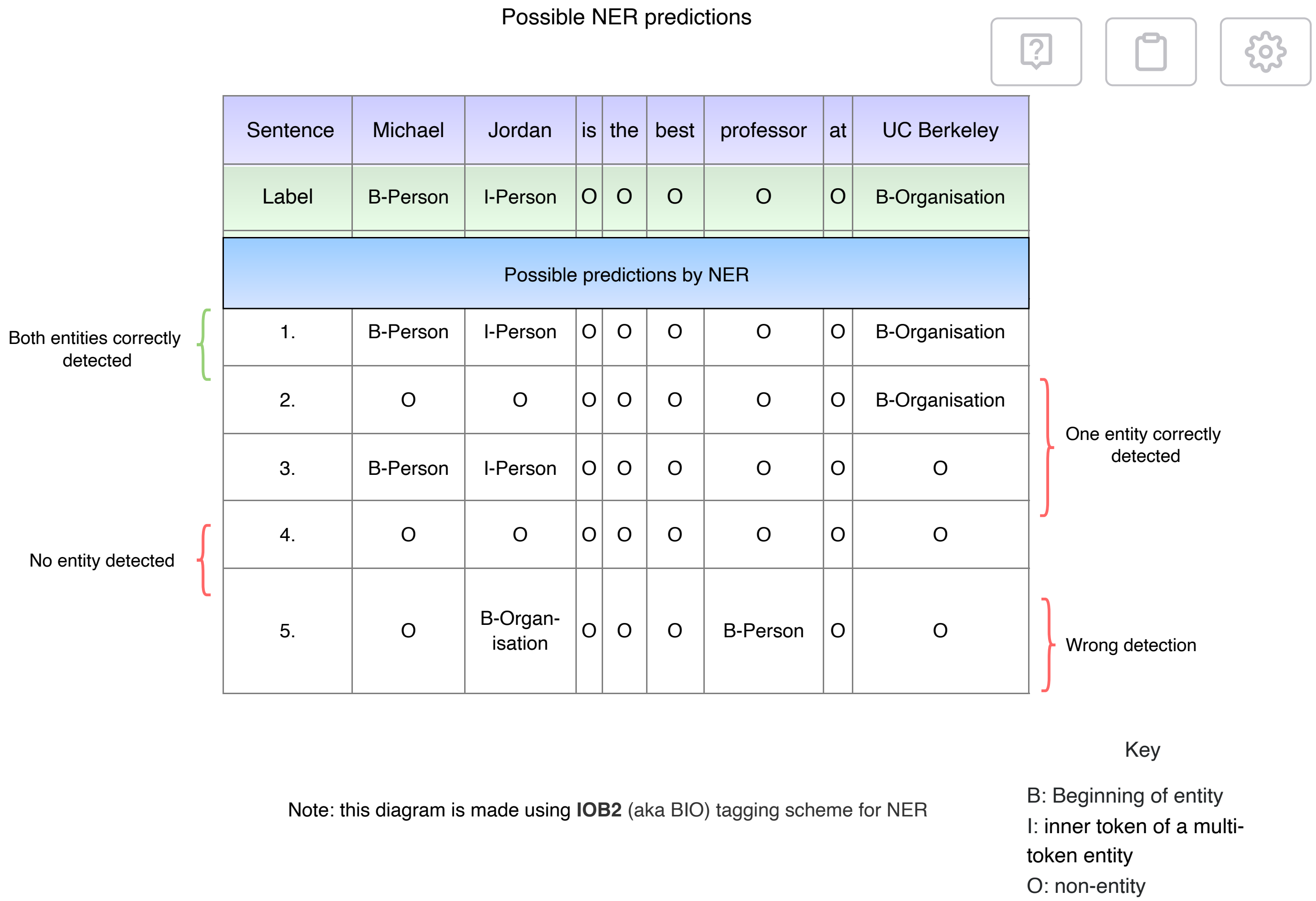
For the first layer/component, i.e., the recognition layer, you want to extract all the entity mentions from a given sentence. We will continue with the previous sentence example, i.e., “Michael Jordan is the best professor at UC Berkeley”.

It has two entity mentions:

1. Michael Jordan
2. UC Berkeley

NER should be able to detect both entities correctly. However, it may detect:

- Both correctly
- One correctly
- None correctly (wrongly detect non-entity as an entity)
- Correct entity but with the wrong type
- No entity, i.e., altogether miss the entities in the sentence



Possible NER predictions

Sentence	Michael	Jordan	is	the	best	professor	at	UC Berkeley
Label	B-Person	I-Person	O	O	O	O	O	B-Organisation
Possible predictions by NER								
1.	B-Person	I-Person	O	O	O	O	O	B-Organisation
2.	O	O	O	O	O	O	O	B-Organisation
3.	B-Person	I-Person	O	O	O	O	O	O
4.	O	O	O	O	O	O	O	O
5.	O	B-Organ-isation	O	O	O	B-Person	O	O

Both entities correctly detected
One entity correctly detected
No entity detected
Wrong detection

Note: this diagram is made using **IOB2** (aka BIO) tagging scheme for NER

Key

B: Beginning of entity
I: inner token of a multi-token entity
O: non-entity

Possible predictions by NER for given sentence

> 📝 You will call a recognition/detection of a named entity correct, only if it is an exact match of the entity in the labeled data. If NER only recognizes “Michael” as an entity and misses the “Jordan” part, it would be considered wrong. Moreover, if NER recognizes “Michael Jordan” as an entity but with the wrong type (say Organization), again, it would be considered wrong.

Given the above context on the correctness of the system, both precision and recall are important for measuring the performance of NER. They will be defined as:

$$\text{Precision} = \frac{no.\ of\ correctly\ recognized\ named\ entities}{no\ of\ total\ recognized\ named\ entities}$$

Recall = $\frac{no.\ of\ correctly\ recognized\ named\ entities}{no\ of\ named\ entities\ in\ corpus}$

You may have models that have excellent precision but poor recall. For instance, when such a model recognizes entities, they will be mostly correct. However, it may not recognize all the entities present in the sentence. Similarly, you could have models that have excellent recall but poor precision.

You want your model to have both high precision and high recall. To look at both, collectively, you will use the F1-score as the metric.

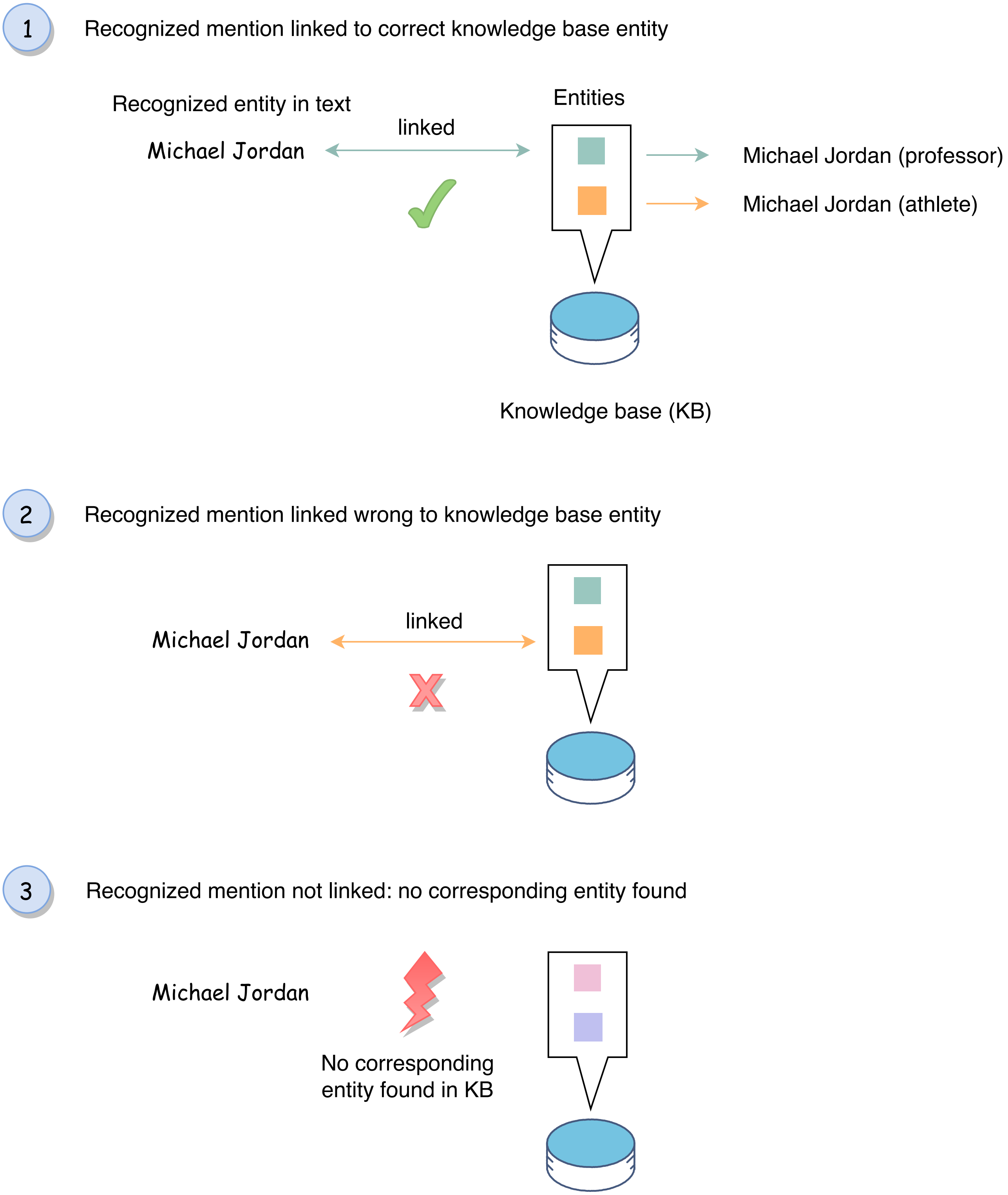
F1-score = $2 * \frac{precision * recall}{precision + recall}$

Disambiguation#

The disambiguation layer/component receives the recognized entity mentions in the text and links them to entities in the knowledge base. It might:

- Link the mention to the correct entity
- Link the mention to the wrong entity
- Not link the mention to any entity (in case it does not find any corresponding entity in the knowledge base)

Possible disambiguation outputs


1
Recognized mention linked to correct knowledge base entity
Recognized entity in text
Michael Jordan
linked
✓
Entities
Michael Jordan (professor)
Michael Jordan (athlete)
Knowledge base (KB)
2
Recognized mention linked wrong to knowledge base entity
Michael Jordan
linked
X
3
Recognized mention not linked: no corresponding entity found
Michael Jordan
No corresponding
entity found in KB

Possible disambiguation outputs for recognized entity mentions in text

This component will perform linking for all entities recogniz[illegible] w[illegible] either be linked to an object in the knowledge base or not linked at all (i.e., the model predicts that it doesn’t have a corresponding object in the knowledge base). So, the concept of recall doesn’t really apply here as each entity is going to be linked(to either an object or Null). Therefore, it makes sense to only use precision as the metric for disambiguation.

Precision = $\frac{no.\ of\ mentions\ correctly\ linked}{no.\ of\ total\ mentions}$

Named-entity linking component#

You have seen the metrics for two main components of the entity linking system. Now, let’s devise a metric to measure the goodness provided by the entity linking component as a whole. You will use F1-score as the end-to-end metric. The definition of a true positive, true negative, false positive, and false negative, for the calculation of end-to-end F1-score (and precision and recall by extension), is as follows:

🔷 *True positive:* an entity has been correctly recognized and linked.

🔷 *True negative:* a non-entity has been correctly recognized as a non-entity or an entity that has no corresponding entity in the knowledge base is not linked.

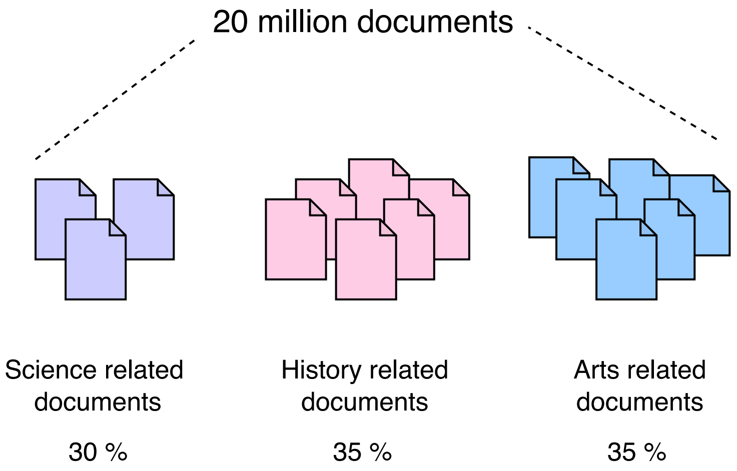
🔷 *False positive:* a non-entity has been wrongly recognized as an entity or an entity has been wrongly linked.

🔷 *False negative:* an entity is wrongly recognized as a non-entity, or an entity that has a corresponding entity in the knowledge base is not linked.

Micro vs. macro metrics#

We can compute *micro-averaged* or *macro-averaged* versions of metrics based on what is important for us in a particular situation.

Let's take an example of a corpus of documents to see scenarios where micro and macro metrics will have significantly different results. Let's assume that the corpus has one million documents that contain twenty million entities collectively. The documents can be categorized as science (30% of the corpus), arts (35% of the corpus), and history (35% of the corpus) related.



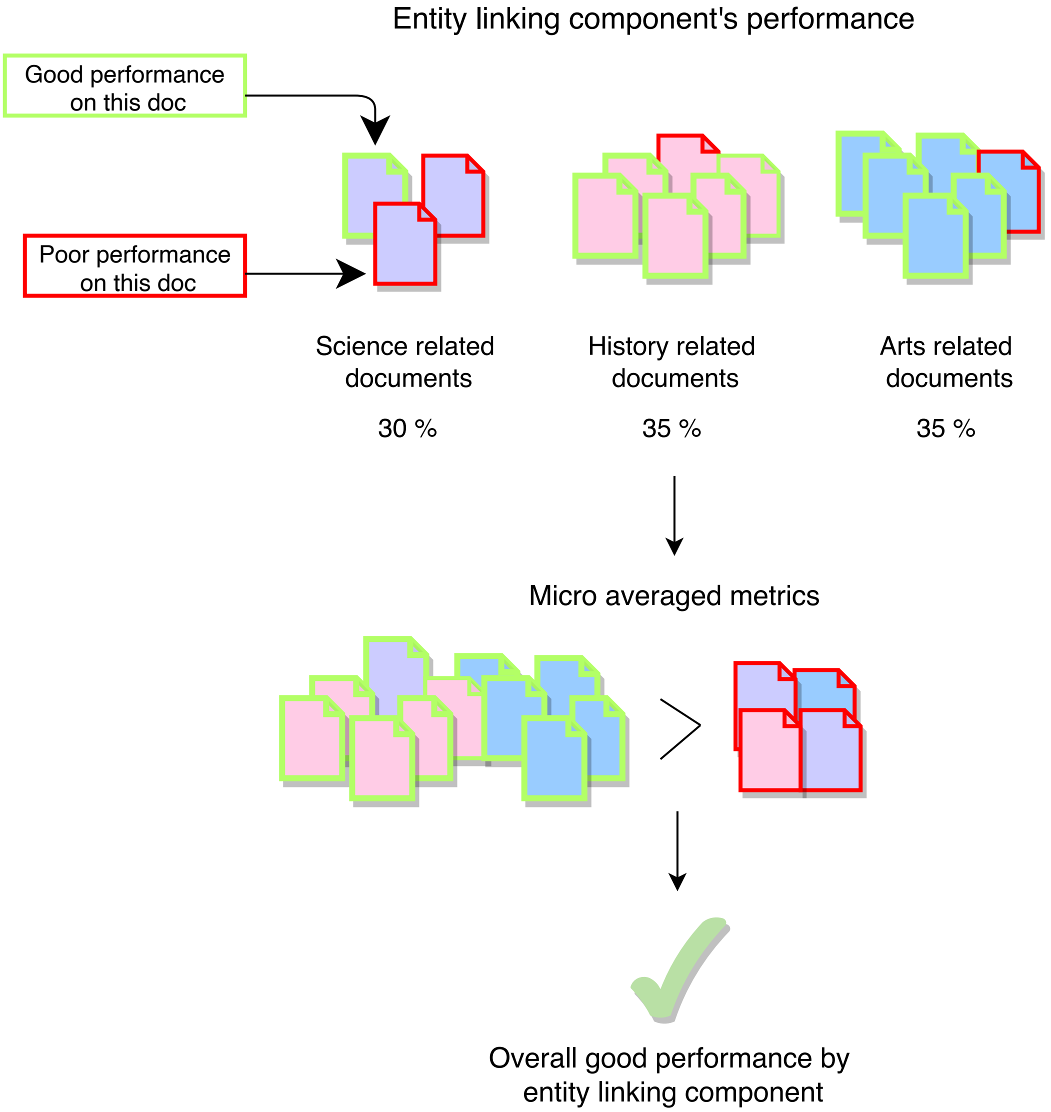
Corpus

> 📝 A macro-average computes the metric independently for each document and takes the average (giving equal weightage to all documents). In contrast, a micro-average aggregates the contributions of all documents to compute the average metric.

- Micro-averaged metrics

  Assume that you are only interested in how well you recognize these twenty million entities, without paying any attention to the performance across individual documents. Here, you will opt for micro-averaged F1-score and micro-averaged precision and recall, by extension.

> 📝 We are focused on the overall performance of the li component. We don’t care if the performance is better for a certain set of documents and not good for another set, so we opt for micro-averaged metrics.


Entity linking component's performance
Good performance on this doc
Poor performance on this doc
Science related documents
30 %
History related documents
35 %
Arts related documents
35 %
Micro averaged metrics
Overall good performance by entity linking component

Micro-averaged metrics

Micro-averaged metrics are computed, as shown below.

Micro-averaged precision = $\frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FP_i}$ _{where\; n= no. \;of\; documents,\; TP_i= true \;positives\; in\; document \;'i' \;and\; FP_i= false\; positive\; in\; document \;'i'}

Micro-averaged recall = \frac{\sum_{i=1}^n TP_i}{\sum_{i=1}^n TP_i + \sum_{i=1}^n FN_i}_{where\; FN_i= false\; positive\; in\; document \;'i'}

Micro-averaged F1-score = $2 * \frac{Micro{-}averaged\ precision\ *\ Micro{-}averaged\ recall}{Micro{-}averaged\ precision\ +\ Micro{-}averaged\ recall}$

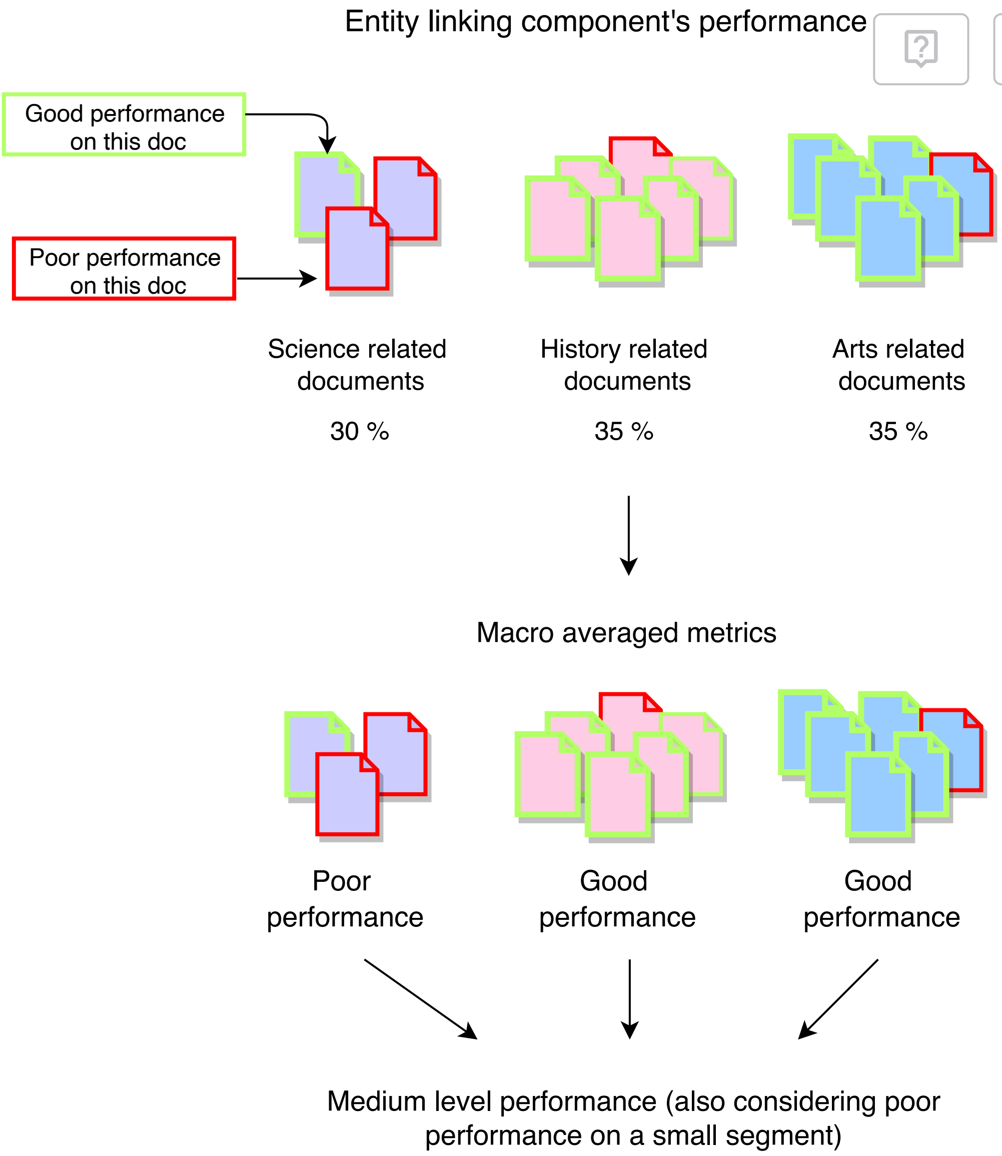
- Macro-averaged metrics

  When you are interested in the individual performance of entity linking across the different types of documents (e.g., science, history, and arts), you will shift to macro-averaged f1 score and macro averaged precision and recall, by extension.

  Let's look at a scenario where you would need to look at entity linking's performance across different types of documents. Assume that 30% of science-related documents contain only 10% of the entities. Now entity linking performs poorly for this small segment, but it performs well for the rest of the 70% documents. Macro averaged metrics allow you to

  average, thereby giving an equal chance of representation to the small segment of science-related documents.

  On the contrary, micro-averaged metrics would not pay attention to these small number of documents where performance is low and give a biased result.


Entity linking component's performance
Good performance on this doc
Poor performance on this doc
Science related documents
30 %
History related documents
35 %
Arts related documents
35 %
Macro averaged metrics
Poor performance
Good performance
Good performance
Medium level performance (also considering poor performance on a small segment)

Macro-averaged metrics

Macro-averaged metrics are computed, as shown below.

Macro-averaged precision =\frac{\sum_{i=1}^n P_{di}}{n}_{where\; n= no. \;of\; documents,\; P_{di}= precision\; over\; document\; 'i'}

Macro-averaged recall = \frac{\sum_{i=1}^n R_{di}}{n}_{where\; n= no. \;of\; documents,\; R_{di}= recall\; over\; document\; 'i'}

Macro-averaged F1-score = $2 * \frac{Macro{-}averaged\ precision * Macro{-}averaged\ recall}{Macro{-}averaged\ precision + Macro{-}averaged\ recall}$

Online metrics#

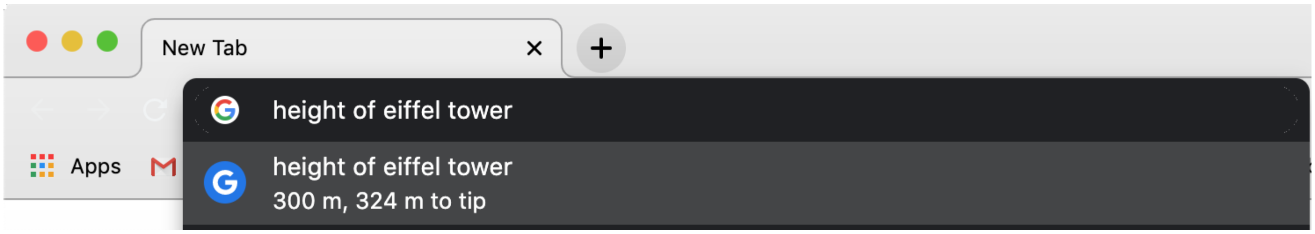
Once you have a model with good offline scores, you still nee [illegible] e [illegible] e bigger task/system's performance will improve if you plug in your new model. So, you would do A/B experiments for these bigger systems to measure their performance with your new entity linking component. The metrics in these A/B tests would be devised based on the overall system, which, in turn, would indicate how well your new model for entity linking is performing as it gets integrated.

To get a better understanding, let's think about two such larger tasks/systems:

1. Search engines
2. Virtual assistants

Search engines

Semantic search allows us to directly answer the user's query by returning the entity or its properties that the user wants to know. The user no longer needs to open up search results and look for the information that is required. As shown in the example below, the user typed their query in the search bar, where entity linking recognized and linked the entity mention *"eiffel tower"*. This enables the system to fetch the entity's property "height" from the knowledge base and directly answer the user's query.



height of eiffel tower

height of eiffel tower
300 m, 324 m to tip

Entity linking for search query allows us to provide direct answers

You have seen how a search engine may use your entity linking component. Now, let's come up with the evaluation metric for the search engine.

For search engines, user satisfaction lies in the query being properly answered, which can be measured by *session success rate*, i.e., % of sessions with user intent satisfied. So, your online A/B experiment will measure the effect of your new entity linking systems on the session success rate.

Virtual assistants

Virtual assistants (VAs) helps perform tasks for a person based on commands or questions. For instance, a user asks Alexa, “What is the height of the Eiffel tower?”. In order to answer this question, the VA needs to detect and link the entities in the user’s question (the entity linking component is required here). In this case, the entity would be “Eiffel tower”. Once this has been done, the VA can fetch the entity’s property “height” and answer the user’s question.

The evaluation metric for the VA would be user satisfaction (percentage of questions successfully answered). It could be measured in various ways by using implicit and explicit feedback. However, the key aspect here is that user satisfaction should improve by plugging in a better model for the entity linking component in the system.

← Back
Problem Statement

Next →
Architectural Components

☑ Mark as Completed

Report an Issue