Normalized Discounted Cumulative Gain

towardsdatascience.com/normalized-discounted-cumulative-gain-37e6f75090e9

8 augustus 2020

A Metric for Evaluating Recommendation Engines


Screening checkpoint ahead
Alberta Health Services
MAIN ENTRANCE
No visitors
Except designated essential visitors

Photo by Graham Ruttan on Unsplash

Do you remember the awkward moment when someone you had a good conversation with forgets your name? In this day and age we have a new standard, an expectation. And when the expectation is not met the feeling is not far off being asked “where do I know you from again?” by some the lady/guy you spent the whole evening with at the pub last week, awkward! — well I don’t actually go to the pub but you get my gist. We are in the era of personalization and personalized content is popping up everywhere — Netflix, Youtube, Amazon, etc. The user demands personalized content, and businesses seek to meet the demands of the users.

In the recent years, many businesses have been employing Machine Learning to develop effective recommender systems to assist in personalizing the users experience. As with all things in life, this feat comes with its challenges. Evaluating the impact of a recommender engine is a major challenge in the development stages, or enhancement stages of a recommender engine. Although we may be sure of the positive impact caused by a recommender system, there’s a much required need to quantify this impact in order to effectively communicate to stakeholders or for when we want to enhance our system in the future.

After a long-winded introduction, I hereby present to you... *Normalized Discounted Cumulative Gain* (NDCG).

> A measure of ranking quality that is often used to measure effectiveness of web search engine algorithms or related applications.

If we are to understand the NDCG metric accordingly we must first understand CG (Cumulative Gain) and DCG (Discounted Cumulative Gain), as well as understanding the two assumptions that we make when we use DCG and its related measures:

1. Highly relevant documents are more useful when appearing earlier in the search engine results list.
2. Highly relevant documents are more useful than marginally relevant documents, which are more useful than non-relevant documents

> (Source: Wikipedia)

Cumulative Gain (CG)

If every recommendation has a graded relevance score associated with it, CG is the sum of graded relevance values of all results in a search result list — see Figure 1 for how we can express this mathematically.

$$CG_p = \sum_{i=1}^{p} rel_i$$

Figure 1: Cumulative Gain mathematical expression

The Cumulative Gain at a particular rank position `p`, where the `rel_i` is the graded relevance of the result at position `i`. To demonstrate this in Python we must first let the variable `setA` be the graded relevance scores of a response to a search query, thereby each graded relevance score is associated with a document.

```
setA = [3, 1, 2, 3, 2, 0]
print(sum(setA))11
```

The problem with CG is that it does not take into consideration the rank of the result set when determining the **usefulness** of a result set. In other words, if we was to reorder the graded relevance scores returned in `setA` we will not get a better insight into the usefulness of the result set since the CG will be unchanged. See the code cell below for an example.

```
setB = sorted(setA, reverse=True)
print(f"setA: {setA}\tCG setA: {cg_a}\nsetB: {setB}\tCG setB: {sum(setB)}")setA: [3, 1, 2, 3, 2, 0]     CG setA: 11
setB: [3, 3, 2, 2, 1, 0]        CG setB: 11
```

`setB` is clearly returning a much more useful set than `setA`, but the CG measure says that they are returning equally as good results.

Discounted Cumulative Gain

To overcome this we introduce DCG. DCG penalizes highly relevant documents that appear lower in the search by reducing the graded relevance value logarithimically proportional to the position of the result — see Figure 2.

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

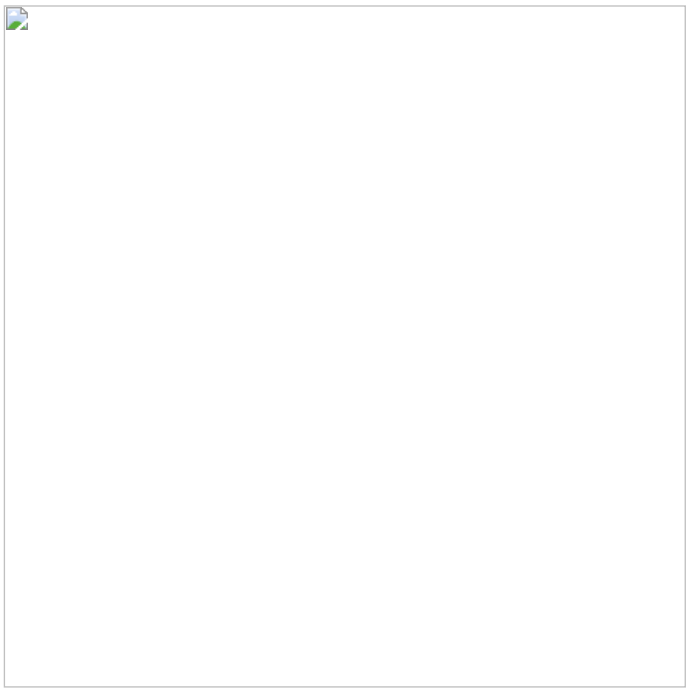

Figure 2: Discounted Cumulative Gain mathematical expression

Below we have created a function called `discountedCumulativeGain` to calculate DCG for `setA` and `setB`. If this is an effective measurement, `setB` should have a higher DCG than `setA` since its results are more useful.

```
import numpy as np

def discountedCumulativeGain(result):
    dcg = []
    for idx, val in enumerate(result):
        numerator = 2**val - 1
        # add 2 because python 0-index
        denominator =  np.log2(idx + 2)
        score = numerator/denominator
        dcg.append(score)
    return sum(dcg)print(f"DCG setA: {discountedCumulativeGain(setA)}\nDCG setB: {discountedCumulativeGain(setB)}")DCG setA:
13.306224081788834
DCG setB: 14.595390756454924
```

The DCG of `setB` is higher than `setA` which aligns with our intuition that `setB` returned more useful results than `setA`.

An issue arises with DCG when we want to compare the search engines performance from one query to the next because search results list can vary in length depending on the query that has been provided. Hence, by normalizing the cumulative gain at each position for a chosen value of `p` across queries we arrive at NDCG. We perform this by sorting all the relevant documents in the corpus by their relative relevance producing the max possible DCG through position `p` (a.k.a Ideal Discounted Cumulative Gain) - see Figure 3.

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$

Where

$$IDCG_p = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

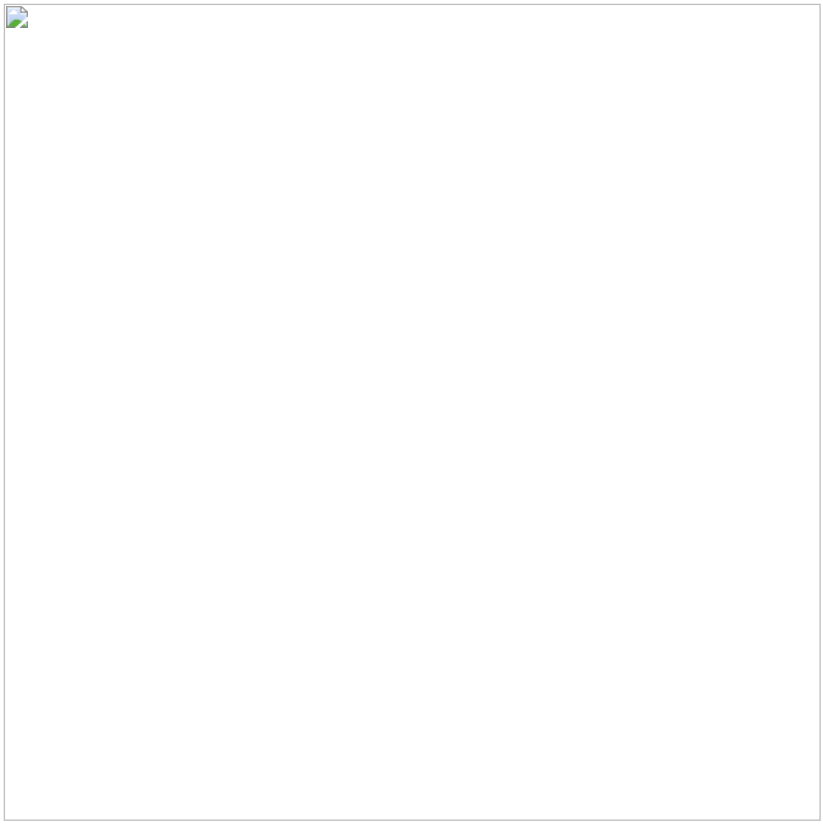

Figure 3: Normalized Discounted Cumulative Gain Mathematical expression; REL_p represents the list of relevant documents (ordered by their relevance) in the corpus up to position p.

To perform this metric in python we created the function `normalizedDiscountedCumulativeGain` to assist with this functionality.

```
def normalizedDiscountedCumulativeGain(result, sorted_result):
    dcg = discountedCumulativeGain(result)
    idcg = discountedCumulativeGain(sorted_result)
    ndcg = dcg / idcg
    return ndcgprint(f"DCG setA: {normalizedDiscountedCumulativeGain(setA, setB)}\nDCG setB:
{normalizedDiscountedCumulativeGain(setB, setB)}")DCG setA: 0.9116730277265138
DCG setB: 1.0
```

The ratios will always be in the range of [0, 1] with 1 being a perfect score — meaning that the DCG is the same as the IDCG. Therefore, the NDCG values can be averaged for all queries to obtain a measure of the average performance of a recommender systems ranking algorithm.

Limitations of NDCG

> (source: Wikipedia)

1. The NDCG does not penalize for bad documents in the results
2. Does not penalize missing documents in the results
3. May not be suitable to measure performance of queiries that may often have several equally good results

Wrap Up

The main difficulty that we face when using NDCG is that often times we don't know the ideal ordering of results when only partial relevance feedback is available. However, the NDCG has proven to be an effect metric to evaluate ranking quality for various problems, for example the Personalized Web Search Challenge, AirBnB New User Booking Challenge, and Personalize Expedia Hotel Searches — ICDM 2013 to name a few.

Thank you for reading to the end of this post. If you'd like to get in contact with me, I am most accessible on LinkedIn.

Kurtis Pykes - AI Writer - Towards Data Science | LinkedIn

View Kurtis Pykes' profile on LinkedIn, the world's largest professional community. Kurtis has 1 job listed on their…

www.linkedin.com