Open in Colab

# Log odds-ratio

The log odds ratio with an informative (and uninformative) Dirichlet prior (described in Monroe et al. 2009, Fighting Words) is a common method for finding distinctive terms in two datasets (see Jurafsky et al. 2014 for an example article that uses it to make an empirical argument). This method for finding distinguishing words combines a number of desirable properties:

- it specifies an intuitive metric (the log-odds) for the ratio of two probabilities
- it can incorporate prior information in the form of pseudocounts, which can either act as a smoothing factor (in the uninformative case) or incorporate real information about the expected frequency of words overall.
- it accounts for variability of a frequency estimate by essentially converting the log-odds to a z-score.

In this homework you will implement this ratio for a dataset of your choice to characterize the words that differentiate each one.

```
# Import libraries NLTK and PDF plumber
# download the packages first here
!pip install nltk
!pip install PyPDF2
import nltk

nltk.download('punkt_tab')
import requests # Download from github
from PyPDF2 import PdfReader # PDF reader
from io import BytesIO
import math
import operator
from collections import Counter
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.2.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk) (1.5.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1)
Requirement already satisfied: PyPDF2 in /usr/local/lib/python3.12/dist-packages (3.0.1)
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

## Part 1

Your first job is to find two datasets with some interesting opposition -- e.g., news articles from CNN vs. FoxNews, books written by Charles Dickens vs. James Joyce, screenplays of dramas vs. comedies. Be creative -- this should be driven by what interests you and should reflect your own originality. **This dataset cannot come from Kaggle**. Feel feel to use web scraping (see here for a great tutorial) or manually copying/pasting text. Aim for more than 10,000 tokens for each dataset.

Save those datasets in two files: "class1_dataset.txt" and "class2_dataset.txt"

**Describe each of those datasets and their source in 100-200 words.**

```
def pdf_to_txt(url, output_file):
  # Modify the URL to get the raw content from GitHub
  response = requests.get(url)
  reader = PdfReader(BytesIO(response.content))
  text = "\n".join(page.extract_text() or "" for page in reader.pages)
  with open(output_file, "w", encoding="utf-8") as f:
      f.write(text.lower())
```

```
# Download and save the dataset in txt files
pdf_to_txt("https://raw.githubusercontent.com/willie84/anlp-assignments-2025/main/The_Constitution_of_Kenya_2010.pc
pdf_to_txt("https://raw.githubusercontent.com/willie84/anlp-assignments-2025/main/SAConstitution-web-eng.pdf", "cla
```

Type your response here:

---

The two datasets are the constitutions of Kenya and South Africa. The Kenyan constitution has been obtained as a PDF document from the Kenyan Parliament website here: https://www.parliament.go.ke/sites/default/files/2017-05/The_Constitution_of_Kenya_2010.pdf .

---

The South African constitution has also been obtained as a PDF ✦ n the South African Department of Justice website here: https://www.justice.gov.za/constitution/SAConstitution-web-eng.pdf . I have downloaded the PDFs and hosted them on my GitHub to be able to download them and use the PDF library to read the PDF content and convert it to text.

## Part 2

Tokenize those texts by filling out the `read_and_tokenize` function below (your choice of tokenizer). The input is a filename and the output should be a list of tokens.

```
def read_and_tokenize(filename: str) -> list[str]:
  """Read the file and output a list of strings (tokens)."""

  from nltk.tokenize import sent_tokenize, word_tokenize

  # Read file
  with open(filename, "r") as f:
    text = f.read()

  # use NLTK word tokenize for tokenizing
  tokens = word_tokenize(text)
  return tokens
```

```
# change these file paths to wherever the datasets you created above live.
class1_tokens = read_and_tokenize("class1_dataset.txt")
class2_tokens = read_and_tokenize("class2_dataset.txt")
print(class1_tokens[:10])
print(class2_tokens[:10])
print(len(class1_tokens))
print(len(class2_tokens))
```

```
['laws', 'of', 'kenya', 'the', 'constitution', 'of', 'kenya', ',', '2010', 'published']
['the', 'constitution', 'of', 'the', 'republic', 'of', 'south', 'africa', ',', '1996']
61830
59399
```

## Part 3

Now let's find the words that characterize each of those sources (with respect to the other). Implement the log-odds ratio with an uninformative Dirichlet prior. This value, $\hat{\zeta}_w^{(i-j)}$ for word $w$ reflecting the difference in usage between corpus $i$ and corpus $j$, is given by the following equation:

$$\hat{\zeta}_w^{(i-j)} = \frac{\hat{d}_w^{(i-j)}}{\sqrt{\sigma^2\left(\hat{d}_w^{(i-j)}\right)}}$$

Where:

$$\hat{d}_w^{(i-j)} = \log\left(\frac{y_w^i + \alpha_w}{n^i + \alpha_0 - y_w^i - \alpha_w)}\right) - \log\left(\frac{y_w^j + \alpha_w}{n^j + \alpha_0 - y_w^j - \alpha_w)}\right)$$

$$\sigma^2\left(\hat{d}_w^{(i-j)}\right) \approx \frac{1}{y_w^i + \alpha_w} + \frac{1}{y_w^j + \alpha_w}$$

And:

- $y_w^i$ = count of word $w$ in corpus $i$ (likewise for $j$)
- $\alpha_w$ = 0.01
- $V$ = size of vocabulary (number of distinct word types)
- $\alpha_0 = V * \alpha_w$
- $n^i$ = number of words in corpus $i$ (likewise for $j$)

In this example, the two corpora are your class1 dataset (e.g., $i$ = your class1) and your class2 dataset (e.g., $j$ = class2). Using this metric, print out the 25 words most strongly aligned with class1, and 25 words most strongly aligned with class2. Again, consult Monroe et al. 2009, Fighting Words for more detail.

```
def logodds_with_uninformative_prior(tokens_i: list[str], tokens_j: list[str], display=25):
    """Print out the log odds results given two lists of tokens."""
    # Count tokens # Use Counter library
    num_of_tokens_per_word_1 = Counter(tokens_i)
    num_of_tokens_per_word_2 = Counter(tokens_j)

    # Get all types ie unique words in the corpora
    all_types = set(num_of_tokens_per_word_1) | set(num_of_tokens_per_word_2) # Make the union

    # Prior
    Prior_constant = 0.01
    Prior_constant_by_len_of_types = len(all_types) * Prior_constant

    dict_of_types_with_log_odds =  {}
    for typez in all_types:
        type_count_in_corpora_1 = num_of_tokens_per_word_1.get(typez, 0)
        type_count_in_corpora_2 = num_of_tokens_per_word_2.get(typez, 0)

        # As per the class notes
        log_odds = (
            math.log((type_count_in_corpora_1 + Prior_constant) / (len(tokens_i) + Prior_constant_by_len_of_types -
            - math.log((type_count_in_corpora_2 + Prior_constant) / (len(tokens_j) + Prior_constant_by_len_of_types
        )

        # From variance formula

        variance = 1.0 / (type_count_in_corpora_1 + Prior_constant) + 1.0 / (type_count_in_corpora_2 + Prior_consta

        # Somehow get the Z value
        dict_of_types_with_log_odds[typez] = log_odds / math.sqrt(variance)

    # Sort the log odds

    sorted_log_odds = sorted(dict_of_types_with_log_odds.items(), key=lambda item: item[1], reverse=True) # descenc

    # Display results
    print("Words that are aligned with document 1 which is Kenyan constitution:")
    for typezz, score in sorted_log_odds[:display]:
        print(f"{score:.3f}\t{typezz}")

    print("Words that are aligned with document 2 which is South Africa  constitution:")
    for typezz, score in reversed(sorted_log_odds[-display:]):
        print(f"{score:.3f}\t{typezz}")
```

```
logodds_with_uninformative_prior(class1_tokens,class1_tokens)
```

```
Words that are aligned with document 1 which is Kenyan constitution:
0.000   contents
0.000   ability
0.000   nominate
0.000   thirtieth
0.000   occur
0.000   248
0.000   they
0.000   tabulated
0.000   193
0.000   method
0.000   singly
0.000   none
0.000   murang
0.000   166-appointment
0.000   examination
0.000   two-republic
0.000   substance
0.000   mobility
0.000   211-borrowing
0.000   incurred
0.000   discontinue
0.000   165-high
0.000   nominates
0.000   acquisition
0.000   city165
Words that are aligned with document 2 which is South Africa  constitution:
0.000   fee
0.000   margin
0.000   context
0.000   programme
0.000   chairing
0.000   question
0.000   deputy
0.000   won
0.000   before
0.000   reflects
0.000   reflect
0.000   concurrent
0.000   2-independent
0.000   biodiversity
0.000   execute
0.000   entrusted
0.000   includes-
0.000   attainable
0.000   intellectual
0.000   2008
0.000   .192
0.000   representative
0.000   danger
```

```
0.000   146
0.000   exercises
```

To check your work, you can run log-odds on the party platforms from the lab section. With `nltk.word_tokenize` *before* lower-casing, these should be your top 5 words (and scores, roughly). Depending on your tokenization strategy, your scores might be slightly different.

**Democrat**:

```
president:  4.75
biden:  4.27
to: 4.11
he: 4.09
has:    4.08
```

**Republican**

```
republicans:    -13.45
our:    -11.23
will:   -10.88
american:   -10.01
restore:    -7.97
```

```
!wget --no-check-certificate https://raw.githubusercontent.com/dbamman/anlp25/main/data/2024_democrat_party_platfo
!wget --no-check-certificate https://raw.githubusercontent.com/dbamman/anlp25/main/data/2024_republican_party_plat
```

```
--2025-09-09 22:09:29--  https://raw.githubusercontent.com/dbamman/anlp25/main/data/2024_democrat_party_platform.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 283046 (276K) [text/plain]
Saving to: ‘2024_democrat_party_platform.txt.3’

2024_democrat_party 100%[===================>] 276.41K  --.-KB/s    in 0.02s

2025-09-09 22:09:30 (15.2 MB/s) - ‘2024_democrat_party_platform.txt.3’ saved [283046/283046]

--2025-09-09 22:09:30--  https://raw.githubusercontent.com/dbamman/anlp25/main/data/2024_republican_party_platform.t
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.111.133, 185.199.110.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 35319 (34K) [text/plain]
Saving to: ‘2024_republican_party_platform.txt.3’

2024_republican_par 100%[===================>]  34.49K  --.-KB/s    in 0.004s

2025-09-09 22:09:30 (7.81 MB/s) - ‘2024_republican_party_platform.txt.3’ saved [35319/35319]
```

```
import nltk
logodds_with_uninformative_prior(
    [w.lower() for w in read_and_tokenize("2024_democrat_party_platform.txt")],
    [w.lower() for w in read_and_tokenize("2024_republican_party_platform.txt")]
)
```

```
Words that are aligned with document 1 which is Kenyan constitution:
4.751   president
4.267   biden
4.110   to
4.090   he
4.076   has
3.761   more
3.389   democrats
3.139   for
3.133   also
3.064   administration
3.056   's
2.931   his
2.900   a
2.818   is
2.775   $
2.563   in
2.514   than
2.511   care
2.488   communities
2.297   as
2.275   continue
2.263   working
2.260   work
```

```
2.246   americans
2.235   year
Words that are aligned with document 2 which is South Africa  constitution:
-13.446 republicans
-11.235 our
-10.882 will
-10.014 american
-7.966  restore
-7.110  great
-6.325  illegal
-6.072  republican
-5.910  policies
-5.797  :
-5.787  stop
-5.563  again
-5.555  we
-5.551  inflation
-5.506  4
-5.436  must
-5.323  1
-5.304  party
-5.281  3
-5.181  common
-5.154  bring
-5.142  peace
-5.117  5
-5.113  education
-4.944  commitment
```

Start coding or generate with AI.