

Model details

Goal

- 600,000 queries
- 1,500 categories
- A query is a sequence of words
- A query belongs to a category
- Goal :
 - Predict the category of a query

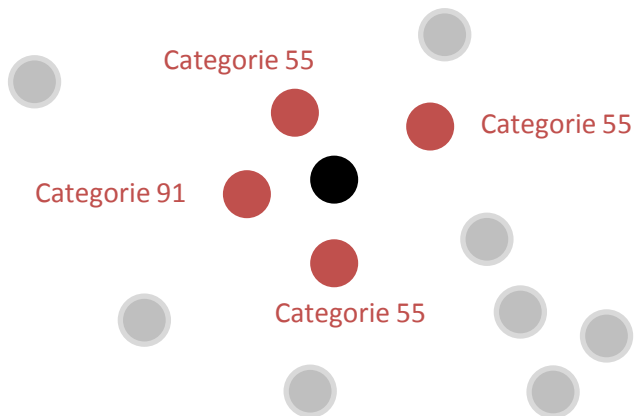
Not a trivial ML problem

- Sparsity of data
 - 70,000 words as features
 - A query has only few words
- Skewed distributions
- 1,500 different outcomes
- Conclusion
 - Logistic regression, Boosting trees might not do the job

KNN, overview

- In such a case, from XP
 - KNN might be a good fit
- Basic ideas behind KNN algorithm
 - For a query target :
 1. Extract the TOP N closest queries in the training set
 2. Compute « category » distribution among the TOP N
 3. Pick up the mode's distribution as the prediction

KNN, overview



- Query Target
- TOP N closest queries
 - **Catégorie 55** : $\frac{3}{4}$
 - Catégorie 91 : $\frac{1}{4}$
- Prediction of Query Target :
 - **Catégorie 55**

KNN, similarity

- What *close to* means?
 - Need a similarity measure
- Similarity between a and b :
 - $S(a, b) = \sum P_{w_i}$
 - W_i , the words shared by a and b
 - P_{w_i} , the Discriminant Power of word i

KNN, similarity

Query a : **the jordan game** tickets

Query b : **the game** of **jordan** book

Words W_i : **jordan** , **game** , **the**

500

100

25

Discriminant Power of the word 'game'



$$S(a,b) : 500 + 100 + 25 = 625$$

KNN, similarity

- We prefer to normalize the similarity
 - We will use the maximum similarity possible to normalize the score
 - $S_n(a, b) = \left(\frac{S(a, b)}{S(a, b)_{MAX}} \right)$

Query target : the jordan game tickets

25

500

100

175

A query should contains the words [*the,jordan,game,tickets*] to

Get the maximum similarity with the query target.

In this case, the similarity will be **800** (25+500+100+175)

$$S_n(target, 'game of jordan') = \frac{600}{800} = 3/4$$

Discriminant Power, definition

- Discriminant Power, P_{wi} of the word i is :

- $$P_{W_i} = \frac{p(\text{prediction}=\text{TRUE} | w_i)}{p(\text{prediction}=\text{TRUE})}$$

- $p(\text{prediction} = \text{TRUE} | w_i) = \int p(c|w_i)^2 dc$
- $p(\text{prediction} = \text{TRUE}) = \iint p(c|w)^2 p(w) dc dw$

Discriminant Power, justification

- Why? $p(\text{prediction} = \text{TRUE} | w_i) = \int p(c | w_i)^2 dc$

$p(c | \text{basket})$

A random query comes in :

- I know it contains the word basket
- The probability it is of 'Sport' category is therefore 0.5
- The probability I will correct predict 'Sport' when it is actually 'Sport' is $0.5 * 0.5$
- The probability I will correct predict any category is $\text{Sum}(p(c | \text{basket})^2)$

1. Sport : 0.5
2. Shopping : 0.3
3. News : 0.1
4. Book : 0.1

Discriminant Power, examples

- Othaheel : 1249
- Nuoma : 1275
- Oscars : 1050
- Fluorigard : 703
- Chunks : 406
- Positive : 206
- Maker : 94
- Low : 19
- The : <0.001
- A : <0.001
- For : <0.001

Remark

- $p(c|w_i)$ is **not that trivial to compute**
- Let's imagine only two categories A and B
 - **3** queries of category A containing the word 'covid'
 - **0** queries of category B containing the word 'covid'
- What is $p(c=A | \text{'covid'})$? 1?
- What is $p(c=B | \text{'covid'})$? 0?
- We know it is **not a good approximation**

Remark

- $p(c|w_i)$ is not that trivial to compute
- To solve this problem, we used Bayes Theorem
- To not spend too much time on it, please go and see my [solution to a very similar problem](#)

Picking Candidates Strategy, motivation

- How to find the top N closest queries to a query target?
 - Compute similarity with all the queries except the target?
 - 600,000 queries , the cost would be too high
 - Compute Similarity with the candidate queries (sharing the same words)?
 - ~3,000 queries on avg, the cost would be better but still very high
- Idea : looking for the best queries first

Picking Candidates Strategy, details

- Rank the structure of potential closest queries
- Look first in the best set of potential queries
 - Then look for the second best
 - And so on ...
 - Stop when you got enough*

** The only hyperparameter of the model*

Picking Candidates Strategy, example

Query Target : **jordan** game ticket

500

100

250

The query that is the closest to the query target should contains

- The words : *jordan, game, ticket*
- It would give a maximum similarity of **850**, (relative score of 1)

Set of potential queries ranking

1. 850 (1.00) : *jordan, ticket, game*
2. 750 (0.88) : *jordan, ticket*
3. 600 (0.70) : *jordan, game*
4. 500 (0.58) : *jordan*
5. 350 (0.41) : *ticket, game*
6. 250 (0.29) : *ticket*
7. 100 (0.12) : *game*

Picking Candidates Strategy, hyperparameter

Set of potential queries ranking

1. 850 (1.00) : *jordan, ticket, game*
2. 750 (0.88) : *jordan, ticket*
3. 600 (0.70) : *jordan, game*
4. 500 (0.58) : *jordan*
5. 350 (0.41) : *ticket, game*
6. 250 (0.29) : *ticket*
7. 100 (0.12) : *game*

Hyperparameter = 0.95
gives the best results

We stop looking for other queries when :

- (relative score) < hyperparameter value
- AND
- Number of queries reached so far > 0
 - If Number of queries reached so far = 0:
 - Continue to the next set

Picking Candidates Strategy, benefice

- With this strategy :
 - No need to collect candidates anymore
 - Retrieve directly the top N queries according to hyp
 - Therefore, only looking for ~15 queries VS ~3,000

Remark

- What if a query target does not contains a **known** word?
 - Then, we assume we know nothing about this query
 - So, we will pick the most frequent category as prediction