Multi-label kNN with TF-IDF vectorization

for „Data Ninja” competition

Recipe

High level algorithm:

1) Data preprocessing

2) Feature selection

3) Creating vocabulary

4) TF-IDF creation

5) Closest neighbors selection

6) Label evaluation

Data preprocessing

'Nawiąże współpracę z odbiorcami płynów eksploatacyjnych luzem i pakowanych takich jak:\n\ nGLIDEX koncentrat zielony luz -70°C \n\nCena 8 zł. za1L brutto detal\n\nCeny hurtowe upust do -35%\n\n\n\ nDostarczamy kurierem najczęściej w bańkach \n\n35 L. 200L. 1'

'nawiaze wspolprace z odbiorcami plynow eksploatacyjnych luzem i pakowanych takich jak glidex koncentrat zielony luz 70 c cena 8 zl za1l brutto detal ceny hurtowe upust do 35 dostarczamy kurierem najczesciej w bankach 35 l 200l 1'

Note:
Correct labels often contain special characters
It was okay to leave „&” or „+”

Feature selection

title:

„opiekunki do osob starszych na terenie city firma zatrudni”

description:

„firma gwarant tomczyk sp j zatrudni osoby do pracy stalej lub dorywczej dodatkowej w wybrane dni tygodnia w charakterze opiekunow osob starszych chorych na terenie city wyksztalcenie min podstawowe oferujemy praca na podstawie u”

labels:

„praca praca_gdynia gdynia opieka gdyni”

Drawbacks:

- **omitting localization data**
- **use of categories might have helped**

Vocabulary

Built from unique words in correct labels only

Size: ~150k

This way we **omit stop words** and focus on informations important for label selection only.

TF-IDF (Term Frequency - Inverse Document Frequency)

Matrix M with term counts, its size is m x n:
m is the number of ads, and n is the size of vocabulary

$$\mathrm{tf}_{i,j} = \frac{n_{i,j}}{\max_k n_{i,k}}$$

$$\mathrm{idf}_j = \log\left(\frac{m}{|\{d : t_j \in d\}|}\right)$$

$$\text{tf-idf}_{i,j} = \mathrm{tf}_{i,j} \cdot \mathrm{idf}_j$$

Weighted kNN

Find 46 closest training neighbors for each of the test examples,
and save found distances...

Consider metrics:

- cosine similarity
- asymetric cosine similarity
- S-Plus
- TS-SS

Note:

It’s difficult to do with large, sparse arrays, but efficent libraries can make it a few hours.

Selection by voting

Calculate cumulative sum of weighted votes for each of the neighbors' labels

The weight of a label depends on:

1) the distance from the neighbor
2) which label in order it was

<u>Note</u>:

E.g. first label is usually better than the second, thus this order was weighted too.

Final evaluation

Lower the score of a label if it (or its part) is not present in the title!

Raise the score of a label if it (or its part) is present in the title!

Note:

It requires a lot of parameters tuning, since
there should be a proper weight for each part of the word being absent or present.

Thank you!