Pour bien comprendre pourquoi les poids de 8 / 4 / 1 n’ont pas un impact suffisant, il va falloir analyser plus en détail comment SOLR déduit le SCORE a partir duquel il fait sont « Tri par pertinence »

Ci-dessous deux PROFILS qui sont renvoyé par la recherche supercalifragilisticexpialidocious

Dans l’ordre de retour de SOLR

**DOCUMENT 1**

nomEtablissementFormation : POIDS – 1.0

titreFormationAncienne : POIDS – 1.0

{

"**sid**": "2583777",

"**objectifProfessionnel**": "Les recruteurs voient cet objectif juste sous le titre de votre profil. \nCe n’est pas une simple annonce de recherche, c’est ce qui est important pour vous. Appuyez-vous sur une expérience, un diplôme ou un talent particulier pour présenter ce que vous visez.",

"**titreCV**": "Testeur de makis",

"**nomEtablissementFormation**": [

"Unviversité",

"Test supercalifragilisticexpialidocious"

],

"**titreFormationAncienne**": [

"Autre formation supercalifragilisticexpialidocious"

],

"**score**": 2.259898

},

**DOCUMENT 2**

objectifProfessionnel : POIDS – 4.0

{

"**sid**": "2583790",

"**objectifProfessionnel**": "supercalifragilisticexpialidocious Les recruteurs voient cet objectif juste sous le titre de votre profil. \nCe n’est pas une simple annonce de recherche, c’est ce qui est important pour vous. Appuyez-vous sur une expérience, un diplôme ou un talent particulier pour présenter",

"**titreCV**": "Testeur de vacances",

"**nomEtablissementFormation**": [

"Ecole spécialisée"

],

"**score**": 0.8788492

},

On voit ici que le profil dont les poids sont de 1 ont un meilleur score que le profil avec un poids de 4.

Pour comprendre il va falloir faire des mathématiques.

Tout d’abord voilà la requête SOLR simplifié :

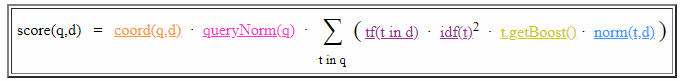
(titreFormationAncienne:(supercalifragilisticexpialidocious)^1.0) OR

(nomEtablissementFormation:(supercalifragilisticexpialidocious)^1.0) OR

(objectifProfessionnel:(supercalifragilisticexpialidocious)^4.0)

titreFormation et nomEtablissement ont un poid de 1 et objectifProfessionel a un poid de 4

Et voila la formule utilisé par SOLR pour calculer le score :



Explication de termes un à un :

***tf(t in d)*** est le Term Frequency, c’est-à-dire que si le terme apparait plus d’une fois dans un champ il a plus de valeur que si il apparait que une fois.

Ici dans notre exemple, le terme supercalifragilisticexpialidocious n’apparait qu’une seule fois dans chaque champs – son tf est de 1.0 pour chaque champs

***idf(t)*** est le Inverse Document Frequency, c’est-à-dire que les terme les moins fréquent dans l’index ont une plus haute contribution au score total.

Ici le terme supercalifragilisticexpialidocious est peux fréquent dans l’index, ce qui fait que son idf est de 12.783917

***t.getBoost()*** est simplement le Boost qu’on a choisi, 1.0 ou 4.0.

***norm(t,d)*** est determiné en fonction d’un certain nombre de paramètre qui ont lieu au moment de l’indexation (mais je crois que on ne fait rien de special, et chaque champs doit avoir la même priorité). Mais SURTOUT - [**lengthNorm**(field)](https://lucene.apache.org/core/3_0_3/api/core/org/apache/lucene/search/Similarity.html#lengthNorm(java.lang.String,%20int)) est calculé au moment de l’indexation.

Cette valeur dépend du nombre de mot dans le champs. Les champs les plus COURT ont un impact plus important que les champs les plus longs.

Ici ***nomEtablissementFormation :*** *Test supercalifragilisticexpialidocious*

a un norm de **0.5**

Et *"****titreFormationAncienne****": Autre formation supercalifragilisticexpialidocious* a un norm de **0.625**

*"****objectifProfessionnel****": "supercalifragilisticexpialidocious Les recruteurs voient cet objectif juste sous le titre de votre profil. \nCe n’est pas une simple annonce de recherche, c’est ce qui est important pour vous. Appuyez-vous sur une expérience, un diplôme ou un talent particulier pour présenter"*, a un norm de **0.21875**

Champs plus long, donc norm plus petit.

Ce qui fait que si on fait le produit de ces 4 valeurs pour chaque champ :

titreFormationAncienne

termFreq \* idf \* boost \* fieldNorm

1.0 \* 12.783917 \* 1.0 \* 0.625 = 7.989948125

nomEtablissementFormation

1. \* 12.783917 \* 1.0 \* 0.5 = 6.3919585

objectifProfessionnel

1.0 \* 12.783917 \* 4.0 \* 0.21875 = 11.185927375

Dans le calcul il y a aussi queryWeight qui ici a une valeur de (0.018437404 \* idf)

Donc les valeurs finales :

titreFormationAncienne = 7.989948125 \* (12.783917 \* 0.018437404) = 1.8832488

nomEtablissementFormation = 6.3919585 \* (12.783917 \* 0.018437404) = 1.5065991

objectifProfessionnel = 11.185927375 \* (12.783917 \* 0.018437404) = 2.6365483

Maintenant il y a un paramètre qu’on a oublié : titreFormationAncienne et nomEtablissementFormation apparaissent 2 fois dans le même document.

C’est le paramètre  [coord(q,d)](https://lucene.apache.org/core/3_0_3/api/core/org/apache/lucene/search/Similarity.html" \l "formula_coord)  ici on a trois occurrence du terme, et donc

[coord(q,d)](https://lucene.apache.org/core/3_0_3/api/core/org/apache/lucene/search/Similarity.html#formula_coord) = 2/3 pour le document 1.

[coord(q,d)](https://lucene.apache.org/core/3_0_3/api/core/org/apache/lucene/search/Similarity.html#formula_coord) = 1/3 pour le document 2.

Donc on a le résultat suivant :

Document 1 = 0.6666667 \* (1.8832488 + 1.5065991) = **2.2598987**

Document 2 = 0.33333334 \* (2.6365483) = **0.87884945**

C’est pour ça qu’un terme qui apparait une fois dans une longue phrase avec un poids de 4 est moins important que 2 termes qui apparaissent dans deux champs courts avec un poids de 1