PLDAC

Sparse Embedding for Information Retrieval

Rapport de Projet

Étudiants :
Idles MAMOU
Amine DJEGHRI

Numéros Étudiants :
3803676
3801757

Mai 2020

Table des matières

1 Recherche d'Information

1.1 Définition

La recherche d'information est le domaine qui consiste à retrouver des informations à partir d'une collection de documents. L'objectif est de relier le besoin en information d'un utilisateur, modélisé par une requête, avec un ensemble de documents en estimant leur pertinence par rapport à la requête et en les présentant selon un ordre lié à ce degré de pertinence (*Salton et McGill 1983* [29]).

1.2 Types de système de recherche d'information

Il existe principalement deux types de SRI : La recherche ad-hoc et la recherche par filtrage

1.2.1 La recherche ad-hoc

Ce type de système est basé sur la recherche des documents faite directement par l'utilisateur et permet d'obtenir des résultats pertinents à partir de la requête spécifiée. Il permet d'aider l'utilisateur à trouver des informations à un problème actuel et spécifique, tentant de représenter les informations dont nous avons besoin actuellement et à court terme, plutôt qu'à long terme. Les exemples les plus populaire sur la recherche Ad-hoc sont les moteurs de recherche, qui suite à une requete, nous retournent des pages pertinentes et d'autres non-pertinentes.

1.2.2 Filtrage d'information

Contrairement à la recherche ad-hoc, le filtrage est une modélisation à long terme des préférences de l'utilisateur et indique simplement les documents qui pourraient intéresser l'utilisateur. Ce système permet à l'utilisateur de construire les informations dont il a besoin à long terme en les comparant avec les nouvelles informations. Nous pouvons citer comme exemple les systèmes de recommandations qui visent à présenter les éléments qui sont susceptibles d'intéresser l'utilisateur.

1.3 Le processus de recherche d'information

Le processus de RI qui permet, à partir d'une requête, d'ordonnancer les documents est appelé "processus en U". Il se décompose en trois principales étapes, illustrées dans la Figure 1.3.1 et détaillées ci-dessous.

-Le choix d'une représentation pour les documents/requêtes :
En recherche d'information, les termes sont typiquement la plus petite unité de représentation pour l'indexation et la recherche. Différents schémas de représentation induisent différentes notions de similarité entre les termes. Certaines représentations considèrent chaque terme comme une entité distincte (ex [41],[36]), alors que d'autres cherchent à exhiber des similarités sémantiques (ex [40] [13], [51]). Certaines approches se basent sur des représentations en grande dimension (ex [11]) tandis que d'autres cherchent à réduire l'espace de représentation (ex [40],[44],[27]). Elles diffèrent aussi dans la manière d'agréger les représentations de termes pour représenter des unités plus larges comme les phrases ou les documents. Dés lors, bien choisir une représentation devient crucial pour une tache de RI.

-L'appariement :
L'appariement correspond à l'étape qui met en relation la collection de documents, préalablement traités, avec la requête, afin d'identifier les documents pertinents. Cette étape permet au système de RI de retourner une liste de documents à l'utilisateur dont l'idéal serait de faire correspondre deux types de pertinence (*Cosijn and Ingwersen 2000* [33]) :

- *La pertinence système* qui est une mesure algorithmique basée sur le calcul de l'adéquation entre la représentation de la requête et celle de la collection de documents.

- *La pertinence utilisateur* qui est la pertinence subjective que l'utilisateur aurait donné à chacun des documents.

Pour cette étape, le SRI dispose de modèles d'ordonnancement qui calculent la pertinence, appelée également score de similarité ou Relevance Status Value(RSV), pour la paire document-requête.

(ce n'est pas ad hoc non ?)-La re-formulation du besoin en information :
Cette étape permet de redéfinir le besoin de l'utilisateur au fur et à mesure de la session de recherche. Elle peut être effectuée :

- Manuellement, dans le cas où l'utilisateur soumet lui-même une nouvelle requête.
- De façon automatique, lorsque le SRI s'appuie sur les termes importants dans les documents les plus pertinents ou visités par l'utilisateur qui sont réutilisés selon :
  1. une boucle de rétro-pertinence –*relevance feedback*– qui considère les jugements de pertinence de l'utilisateur afin de construire une nouvelle requête traduisant plus finement le besoin en information de l'utilisateur (*Rocchio 1971 [39]*).
  2. une boucle de rétro-pertinence en aveugle ou –*pseudo-relevance feedback*– où le modèle utilise les n premiers documents retournés afin de construire la nouvelle requête (*Cao et al 2008* [15]).

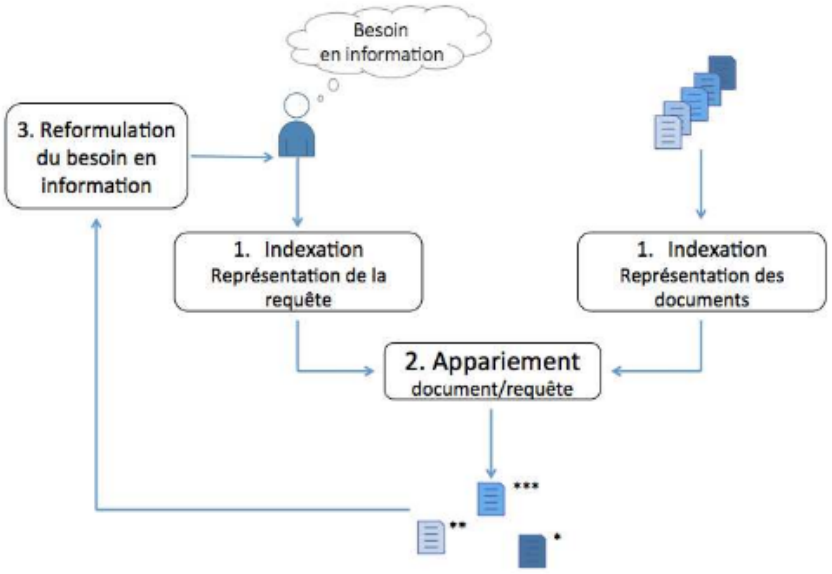


FIGURE 1.3.1 – Processus en U de RI

1.4 Modèles mathématiques de la RI

1.4.1 Modèles ensemblistes

- **Probabilistic model**

- **Le modèle booléen**
  Le modèle booléen est un modèle dans lequel nous pouvons poser des requêtes sous la forme d'une expression booléenne de termes qui peuvent etre combinés avec les opérateurs AND, OR et NOT. Le modèle considère chaque document comme un simple ensemble de mots.

1.4.2 Modèles algébriques

- **Vector space model**
- **Spreading activation model**
- **Latent semantic Indexing**

1.4.3 L'approche probabiliste

- **Probabilistic relevance model**
  Les modèles basés sur la Pseudo-Relevance Feedback ont démontré de solides performances en terme de ranking en contrepartie d'un tour de recherche en plus [48]. Le premier ensemble de documents retournés R_1 est utilisé pour reformuler la requête pour le second tour de recherche. Les documents retournés sont alors présentés à l'utilisateur.
  Dans les modèles RM, la fonction de scoring est représentée par la fonction de divergence de Kullback-Leibler [23] :

  $$score(q,d) = -\sum_{t \in T} p(t|\theta_q) log \frac{p(t|\theta_q)}{p(t|\theta_d)}$$

  Sans PRF :

  $$p(t|\theta_q) = \frac{tf(t,q)}{|q|}$$

  Sous le modèle **RM3** [5], après re-formulation du besoin :

  $$p(t|\theta_q) = \alpha \frac{tf(t,q)}{|q|} + (1-\alpha) \sum_{d \in R_1} p(t|\theta_d) p(d) \prod_{t_{new} \in q} p(t_{new}|\theta_d)$$

  En apportant de nouveaux termes à la requête en utilisant les résultats du premier tour de recherche, les approches PRF permettent un gain de robustesse face aux problèmes liés au peu de vocabulaire dans les requêtes.
- **Okapi BM25**
  Modèle de référence en RI. Proposé par *Robertson et Walker* [41] en 1994 et inspiré du Binary independant model [54], cette approche fait l'hypothèse de l'indépendance des termes dans les documents/requêtes et présente les documents selon un ordre décroissant de leur probabilité de pertinance $P(Relevance|query, document)$. En reformulant ces probabilités en fonction des tf (nombre d'occurrences d'un terme dans un document) et idf (fréquence inverse des documents), on aboutit à la fonction de ranking BM25 suivante :

  $$RSV(q,d) = \sum_{t_q \in q} idf(t_q) . \frac{tf(t_q,d).(k_1+1)}{tf(t_q,d)+k_1.(1-b+b.\frac{|d|}{avgdl}}.$$

  ou, $avgdl$ est la longueur moyenne des documents de la collection, k_1 et b sont des paramètres constants (habituellement 1.2 et 0.75 respectivement).
- **Modèles de langue**
  Dans les modèles de langue (*Ponte et Croft 1998* [36]), les documents sont ordonnés selon la probabilité de la requête sachant le document $p(q|d)$. Cette probabilité est estimée par maximum de vraisemeblance en faisant l'hypothèse de l'indépendance des termes. En outre, elle définit la probabilité de générer une requête q en échantillonnant aléatoirement des termes à partir du document d. Le score de similarité $RSV(q,d)$ est calculé comme suit :

$$RSV(q,d) = p(q|d) = \prod_t p(TF(t) = tf(t)|\theta_d)^{tf(t)} = \sum_t tf(t) \log \frac{tf(t)}{\sum_t tf(t)}.$$

En pratique, pour palier au cas ou un mot de la requête n'apparaîtrait pas dans le document, on effectue un lissage de cette probabilité. Différents lissages ont été proposé dont :

— *Jelinker-Mercer :* $p(t|d) = (1 - \lambda_t).p(t|\theta_d) + \lambda_t.p(t|\theta_C)$

— *Drichelet :* $p(t|d) = \frac{tf(t,d)+\mu.p(t|\theta_C)}{|d|+\mu}$

ou, C représente la collection de documents, λ un paramètre constant (habituellement 0.8 pour les requêtes courtes et 0.2 pour les requêtes longues).

- **Modèles de translation TLM**
  Les deux modèles cités ci-dessus estiment la pertinence d'un document en comptant seulement les occurrences des termes de la requête dans le document. Ainsi, les relations de ces occurrences avec les autres termes du document ne sont pas prises en compte.
  Berger and Lafferty [8] ont proposé une méthode alternative pour estimer $p(t_q|d)$ dans l'approche des modèles de langue. Il s'agit d'un processus de translation dont la formule est la suivante :

$$p(t_q|d) = \sum_{t_d \in d} p(t_q|t_d).p(t_d|d)$$

  La probabilité de translation $p(t_q|t_d)$ permet ainsi au modèle d'extraire de la pertinence des termes présents dans le document et absents de la requête en mettant en évidence une certaine relation sémantique entre les termes.
  Afin d'estimer cette probabilité de translation, *Karimzadeghan et Zhai* [56] ont proposé de se baser sur l'information mutuelle qui mesure la similarité entre la probabilité jointe $p(X, Y)$ et le produit des deux marginales $p(X)p(Y)$ pour deux variables aléatoires X et Y. Dans ce modèle **TLM-MI**, l'information mutuelle entre deux termes u et v est obtenue de la manière suivante :

$$I(u,v) = \sum_{X_u=0,1} \sum_{X_v=0,1} p(X_u, X_v) log \frac{p(X_u,X_v)}{p(X_u)p(X_v)}$$

  où, X_u, X_v sont des variables binaires indiquant la présence ou l'absence du terme dans le document. Les valeurs de l'information mutuelle sont ensuite normalisées pour obtenir la probabilité de translation via la formule suivante :

$$p_{mi}(u|v) = \frac{I(u,v)}{\sum_{u'} I(u',v)}$$

Ces différentes modèles, bien que traditionnels, produisent de très bons résultats en évaluation. Ils servent d'importantes baselines pour la comparaison avec les nouvelles approches type machine learning et deep learning.

1.5 Le Machine Learning pour la RI

Traditionnellement, les modèles de ranking $RSV(q, d)$ ne nécessitent pas une étape d'entraînement. Pour le modèle Okapi-BM25, par exemple, la fonction RSV est représentée par une distribution de probabilité conditionnelle $P(Relevance|q, d)$ [*Robertson et Walker 1994* [41]]. Pour les modèles de langues, $RSV(q, d)$ est représentée par la vraisemblance de la requete sachant le document $p(q|d)$ [*Ponte et Croft 1998* [36]]. Ces probabilités sont obtenues par un simple comptage des occurrences des termes de la requêtes et du document, aucune étape d'entraînement n'est donc nécessaire (sauf pour le paramétrages de quelques constantes nécessaires [49]).
Très vite, avec l'essor des techniques de machine learning, la construction automatique de modèles de ranking pour la RI, et plus précisément pour la recherche web, devint à la mode. Ceci est motivé par plusieurs facteurs. Pour la recherche web, il y a plusieurs signaux qui peuvent représenter de la pertinence, par exemple, le score PageRank d'une page internet. Pour les moteurs de recherche, une quantité énorme de logs comme les données de clicks sont accumulés. Incorporer de telles informations au modèle de ranking et le construire automatiquement à l'aide des techniques de machine learning devient un choix naturel [Hang

Li 2010 [24]]

Pour un modèle $L2R$, une paire requête-document est représentée par un vecteur de réels $x \in \mathbf{R}^n$ et un modèle $f : x \rightarrow \mathbf{R}$ est entraîné pour calculer un score de pertinence. Les features associés au modèle $L2R$ appartiennent à une des trois catégories suivantes [*Tie-Yan Liu 2009* [25]] :

- Des features qui dépendent uniquement du document (ex PageRank, longueur du document).
- Des features qui dépendent aussi bien de la requête que du document (ex TF-IDF, score BM25).
- Des features qui dépendent uniquement de la requête (ex Nombre de mots dans la requête).

Typiquement, le dataset pour un modèle $L2R$ contient un ensemble de requêtes et un un ensemble de documents par requête. Contrairement aux approches habituelles de classification/régression supervisée, la requête et les documents correspondants forment un groupe. Les groupes sont $i.i.d$ alors que les entités à l'intérieur d'un même groupe ne le sont pas [24].
L'information de pertinence, c'est à dire le label, peut être représenté de différentes manières selon l'approche de ranking choisie. Ces approches $L2R$ sont séparées en trois catégories [*Tie-Yan Liu 2009* [25]] :

- The *Pointwise approach* selon laquelle l'information de pertinence est représentée par une valeur numérique associée à chaque paire requête-document. Les documents sont indépendants et on se ramène donc à un cas typique de régression ou de classification supervisée où on cherche à minimiser l'erreur de prédiction. Parmi les algorithmes qui rentrent dans le cadre de cette approche on cite le *Subset Ranking* [47], le *McRank* [34] ou encore le *SVM for Ordinal Classification* [42].
- The *Pairwise approach* selon laquelle l'information de pertinence est représentée sous forme de préférence entre deux documents étant donnée une même requête. Il y a une indépendance entre les pairs de documents et on se ramène donc à un problème de classification binaire pour prédire le document le plus pertinent. Ici, on cherche à minimiser le nombre de couples mal ordonnés plutôt que le score de pertinence. Parmi les algorithmes qui rentrent dans le cadre de cette approche on cite le *RankingSVM* [38], le *RankNet* [4], le *LambdaRank* [10] ou encore le *LambdaMart* [37].
- The *Listwise approach* suivant cette approche, une instance du modèle représente un groupe de documents liés à une requête. Les mesures d'évaluation sont directement incorporées dans la fonction de coût et les modèles sont façonnés de manière à optimiser une mesure (ex MAP ou $NDCG$). Pour une certaine requêteq_i, le modèle de ranking attribut un score à chacun des documents associés d_{ij} représentés par un vecteur de features x_{ij}. Les documents sont ensuite ordonnés et une permutation est obtenue.Parmi les algorithmes qui rentrent dans le cadre de cette approche on cite le *ListNet* [55], le *AdaRank* [50] ou encore le *SVM MAP* [53].

1.6 Evaluation d'un SRI

Afin d'évaluer l'éfficacité d'un système à retourner des documents pertinents, des collections de tests et des mesures d'évaluation orientées rang et non-rang ont été mis en place.

1.6.1 Collections de tests

Les collections de tests se composent d'une collection de documents, une suite de tests de requetes, et d'un ensemble de jugements de pertinence qui se définie généralement par une évaluation binaire (pertinentes ou non pertinentes) pour chaque paire requête-document.

Cranfield

TREC

CLEF

1.6.2 Mesures d'évaluation orientées non-rang

la précision
Les deux mesures de base de la précision sont la précision et le rappel. L'idéal, c'est de réussir à avoir autant de précision que de rappel, et à parfaire ces deux derniers. Mais en réalité, ils se compromettent, et souvent, nous devons faire des sacrifices sur l'un pour pourvoir améliorer l'autre.

F1 Score

1.6.3 Mesures d'évaluation orientées rang

MAP

MRR

DCG et NDCG

11 - point interpolated average precision

2 Représentation des termes

La manière de représenter les termes devient fondamentale en Recherche d'information dès lors que les modèles sont guidés par ces représentations. Dans cette section, nous allons voir en revue les différentes façon de représenter les termes d'une requête ou d'un document.

2.1 Représentations en *one-hot vector*

Sous ce type de représentations, chaque terme appartenant à un vocabulaire de taille fixe V est représenté par un vecteur binaire $v \in \{0,1\}^{|V|}$, ou seulement une des valeurs, correspondant à un terme, du vecteur est à 1 et toutes les autres sont à 0. Ce vecteur représente donc la présence ou absence d'un terme dans une requête/document et les relations sémantiques existantes entre les termes sont donc ignorées.
Typiquement, les modèles de RI traditionnels présentés plus haut se basent tous sur ce type de représentions.

2.2 Représentations distribuées

Les modèles basés sur un représentation binaire (Présence/Absence) de terme produisent de bons résultats en terme d'appareillement lexical. Cependant, ces modèles dépendent beaucoup trop du vocabulaire utilisé dans l'expression des requêtes et des documents. Ils se retrouvent donc en difficulté face aux changements de vocabulaire ou à l'utilisation des synonymes [*Mitra et Craswell 2017* [31]].
Sous l'angle des représentations distribuées, un terme est représenté par un vecteur $v \in R^{|k|}$. v peut être un vecteur à valeurs dispersées (sparse) [35] ou un vecteur dense [40],[44], un vecteur de features obtenues manuellement (cas des modèles $L2R$) ou une représentation issue d'une phase d'apprentissage dont chaque dimension ne peut être interprétée de manière isolée.
Le choix d'une représentation distribuée est motivé par l'hypothèse [18] que les termes qui sont utilisés dans un contexte similaire tendent à être proches sous un angle sémantique. Cependant, comme la notion de similarité n'est pas bien définie en terme de relations entre les termes, le choix des features elles mêmes définit la sémantique voulue.
Nous distinguons donc deux sous-catégories de représentations distribuées. Quand le vecteur de représentation est de grande dimension et à valeurs dispersées, on parle de *représentation explicite* ou *sparse*. D'autre part, lorsque les vecteurs sont denses, de petite dimension et issus d'un apprentissage à partir des données, on parle de représentation projetée ou *embeddings* .

2.2.1 Représentations distribuées explicites (Sparse representations)

Les représentations distribuées explicites partagent les mêmes propriétés de haute dimensionnalité et de dispersion. Elles diffèrent, cependant, dans les choix des features. On peut, par exemple, choisir de représenter un terme selon les documents qui le contiennent (Figure 2.2.1.a) ou selon les termes apparaissant dans son voisinage en faisant le choix ou non de la prise en compte de la distance entre les termes dans ce voisinage. Ces représentations peuvent aussi être améliorées en ajoutant des facteurs de pondération (un indice TF-IDF par exemple).

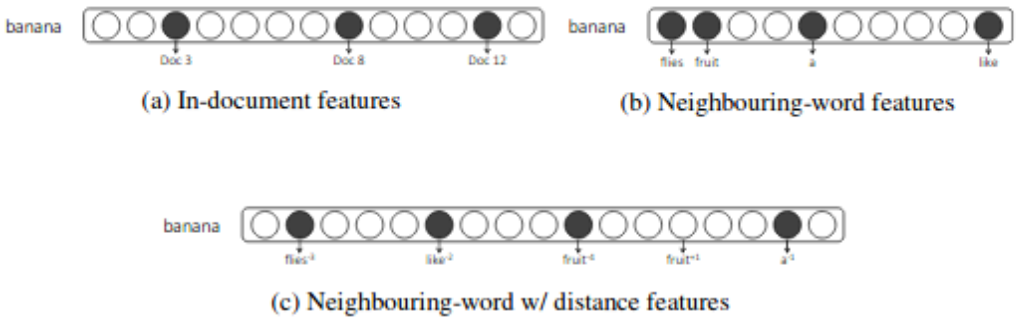


FIGURE 2.2.1 – Exemples de représentations distribuées explicites pour le terme *"banana"*

Ces représentations font apparaître d'importantes notions de similarité entre les termes. En traitement automatique du langage naturel (NLP), dans la quête des analogies entre les termes, on arrive à de bons résultats en effectuant de simples opérations vectorielles sur ce type de représentations [51]
Sur la Figure 2.2.2 on peut voir que le vecteur obtenu en sortie de l'opération vectorielle *Seahawks* (équipe de basket de Seattle) - *Seattle* + *Denver* est tres proche du vecteur *Brancos* (équipe de Denver).

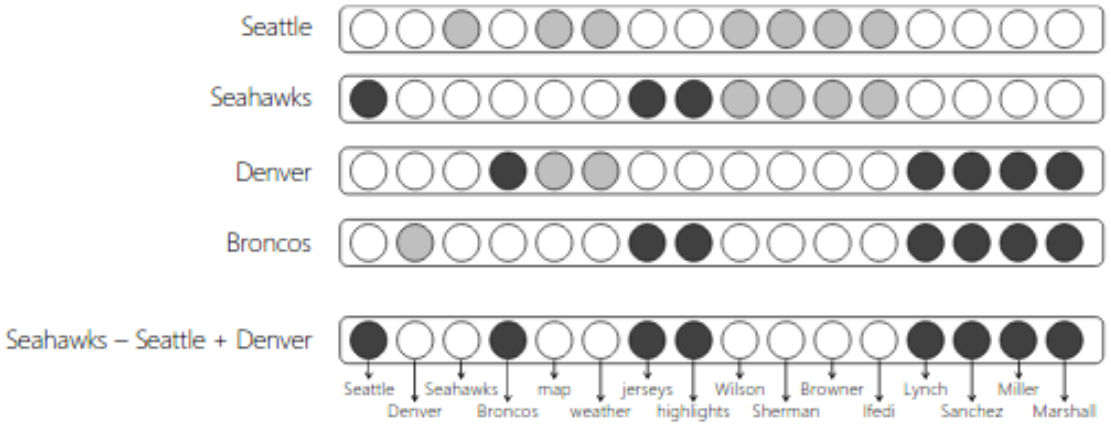


FIGURE 2.2.2 – Démonstration visuelle des analogies entre termes basées sur une représentation explicite en fonction du voisinage des termes

2.2.2 Embeddings

Les représentations distribuées explicites peuvent capturer d'intéressantes notions de similarité entre les termes. Cependant, la majorité des valeurs des features sont nulles et le grand nombre de dimensions, sou-

vent de l'ordre du nombre de documents ou de la taille du vocabulaire, rend les choses difficiles en pratique. Une alternative serait d'apprendre une représentation de plus petite dimension tout en gardant l'aspect sémantique des représentations explicites [27].

Un *Embedding* est une projection d'un terme dans un nouvel espace de dimension inférieure ou les relations sémantiques entre les termes sont respectées. Concrètement, pour assimiler les propriétés des relations termes à termes, les *embeddings* sont construits à partir des représentations distribuées explicites.
Dans ce qui suit, nous présentons quelques-unes des approches populaires pour la construction d'embeddings. On dénote deux principales catégories d'approches :

- **Approches par factorisation**
  Ces approches ont comme point d'entrée la matrice des occurrences terme-document et opèrent une factorisation sur cette dernière pour construire les embeddings.

  — **Latent Semantic Analysis (LSA)**

  Le LSA utilise une matrice X qui décrit l'occurrence des termes dans les documents. C'est une matrice creuse dont les lignes correspondent aux termes et dont les colonnes correspondent aux documents. En effectuant un décomposition en valeurs singulières SVD [12] sur la matrice X on obtient une approximation de la matrice X de rang inférieur X_k [*Deerwester et al. 1990* [40]]. Une ligne (terme) de l'ancienne matrice est donc transformée en un vecteur de k composantes, qui chacune donne l'importance du terme dans chacune des k différentes classes latentes. On peut ainsi voir dans quelle mesure des termes (resp documents) i et j sont liés dans l'espace des classes latentes en évaluant la similarité cosinus entre les deux vecteurs ligne (resp colonne).
  En combinant les dimensions de la matrice des occurrences, on s'arrange pour que deux termes similaires (en matière de sémantique) soient similaires dans l'espace latent.

  — **Probabilistic Latent Semantic Analysis (PLSA)**

  Comparée au modèle LSA qui découle de l'algèbre linéaire pour réduire la matrice des occurrences, le modèle PLSA [Hofmann 1999 [19]] emploi une approche probabiliste pour estimer les différents concepts de la décomposition :

  - Les vecteurs propres U et V de la SVD sont estimés par la distribution de probabilités de tirer un document $P(d|classe)$ et la probabilité de tirer un terme $P(t|classe)$.
  - La matrice des valeurs propres est estimé par la distribution de probabilité des classes (thématiques) $P(classe)$.

  Suivant le principe de vraisemblance, les paramètres du modèle $P(d|classe)$, $P(t|classe)$,$P(classe)$ sont obtenus en maximisant la vraisemblance suivante :

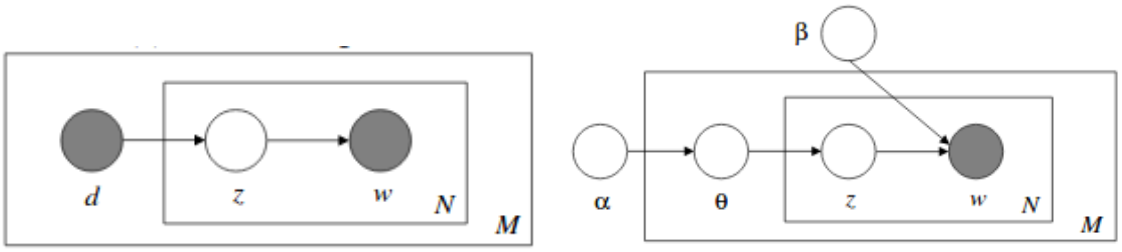
  $$L = \sum_d \sum_t n(d,t) log(\sum_{classe} P(classe)P(t|classe)P(d|classe))$$

  où, $n(d,t)$ est la fréquence du terme t dans le document d.
  En pratique, l'algorithme EM[1] est utilisé pour estimer les différents paramètres.

  — **Latent Drichelet Allocation (LDA)**

  Le Latent Dirichlet Allocation [*Blei et al 2003* [14]] est un modèle probabiliste génératif de corpus. Comme pour le modèle PLSA, le modèle LDA fait l'hypothèse que chaque document est modélisé par un mélange de thèmes qui génère ensuite chaque mot du document [6]. Cependant, contrairement au modèle PLSA, tous les paramètres disposent d'une loi générative, même les documents [voir Figure 2.2.3]. On attribue donc un à priori sur un thème à chaque mot de

chaque document, selon une distribution de Dirichlet sur un ensemble de thèmes [*Francesiaz et al 2015*[43]].



(a) Modèle PLSA (b) Modèle LDA

FIGURE 2.2.3 – Comparaison des modèles PLSA/LDA

- **Approches neuronales pour la construction des embediings**
  Contrairement aux modèles comme LSA qui se basent sur une factorisation de la matrice termes-documents, les modèles neuronaux sont entraînés pour apprendre une représentation des termes en fonction du contexte. Les données en entrée et en sortie du réseaux de neurones sont des représentations *bag of words* et le modèle apprend une représentation dense à faible dimension dans la couche intermédiaire dans le processus de minimisation de l'erreur de prédiction [*Mitra et al 2017* [31]].

  — **Word2Vec**
  Proposés par *Mikolov et al* en 2013 [45], les modèles de ce type tentent de prédire, selon l'architecture, la représentation du contexte (voisinage) en fonction de la représentation du terme (modèle *skip-gram*), ou bien la représentation du terme en fonction du voisinage (modèle *CBOW*).

  L'architecture **skip-gram** est un simple réseaux de neurones à une couche cachée. La représentation bag of words d'un terme est donnée en entrée du modèle, ce dernier apprend une représentation dense à faible dimension dans la couche intermédiaire puis retourne une représentation bag of words du voisinage du terme en question [voir Figure 2.2.4].
  La fonction de coût correspondante à ce modèle est la suivante :

$$Loss_{skip-gram} = -\frac{1}{|S|}\sum_{i=1}^{|S|}\sum_{-c\leq j\leq +c}\log p(t_{i+j}|t_i)$$

  où,

$$p(t_{i+j}|t_i) = \frac{exp((W_{out}v_{t_{i+j}})^T(W_{in}v_{t_i}))}{\sum_{k=1}^{|T|}exp((W_{out}v_{t_k})^T(W_{in}v_{t_i}))}$$

  S est l'ensemble des fenêtres sur le texte d'entraînement, c est le nombre de voisins à prédire de chaque coté du terme t_i.

  On note que le modèle apprends deux matrices de poids W_{in} et W_{out}. En général, seule W_{in} est utilisée pour construire les $embeddings$.

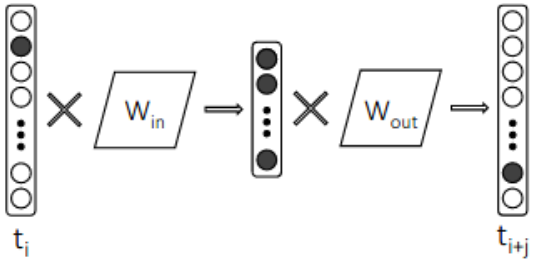


FIGURE 2.2.4 – Architecture Skip-gram

L'architecture **CBOW** (*Continious bag of words*) est similaire à l'architecture *skip-gram* à la différence que ce modèle tente de prédire le terme t_i en fonction des représentations des termes de son voisinage $\{t_{i-c}, ..., t_{i-1}, t_{i+1}, ..., t_{i+c}\}$ [voir Figure 2.2.5].
La fonction de coût correspondante à ce modèle est la suivante :

$$Loss_{CBOW} = -\frac{1}{|S|} \log p(t_i | \sum_{-c \leq j \leq +c} t_{i+j})$$

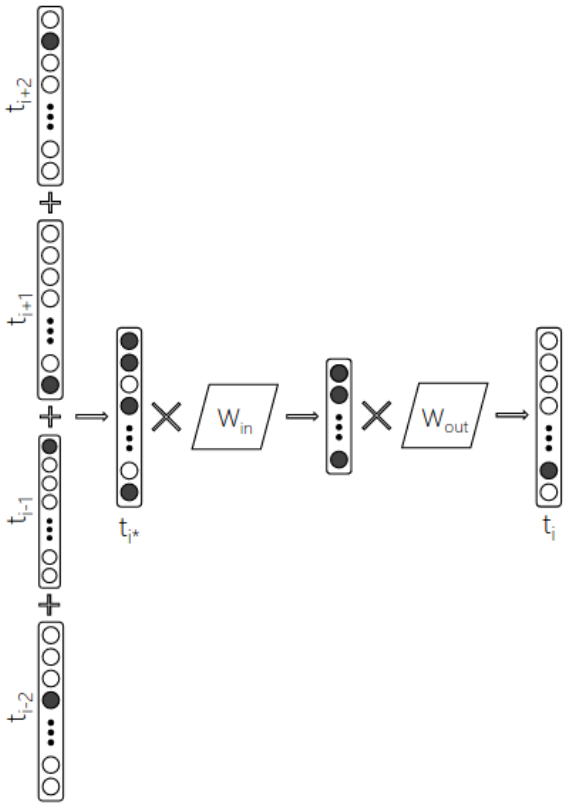


FIGURE 2.2.5 – Architecture CBOW

— **Global Vectors (GloVe)**
Contrairement au modèle skip-gram, qu'on entraîne sur un ensemble contenant plusieurs fois la même paire terme-voisin proportionnellement à leur fréquence, le modèle GloVe [*Pennington*

et al 2014 [21]] agrège les échantillons dans une matrice X, tel que x_{ij} est le nombre d'occurrences du terme t_j dans le voisinage du terme t_i dans l'ensemble d'apprentissage.
Le modèle construit les embeddings dans le processus de minimisation de la fonction de coût suivante :

$$Loss_{GloVe} = -\sum_{i=1}^{|V|}\sum_{j=1}^{|V|} f(x_{ij})(log(x_{ij} - v_{w_i}^T v_{w_j}))^2$$

Comme le modèle word2vec, GloVe génère deux matrices de poids W_{in} et W_{out}. Cependant, contrairement au word2vec, le modèle fait la somme des poids des deux matrices pour générer les embeddings de termes.

— **Paragraph2Vec**
Suivant la popularité du modèle Word2Vec, plusieurs architectures neuronales ont été proposé. Le modèle Paragraph2Vec proposé par *Le et Mikolov* [46] est entraîné pour prédire un terme en fonction de l'ID du document(ou passage) dans lequel il apparaît. En option, le voisinage du terme peut aussi être donné en entrée au modèle [voir Figure 2.2.6] En tentant de minimiser l'erreur de prédiction, le modèle apprend une représentation dense des termes mais aussi des documents (ou passages).

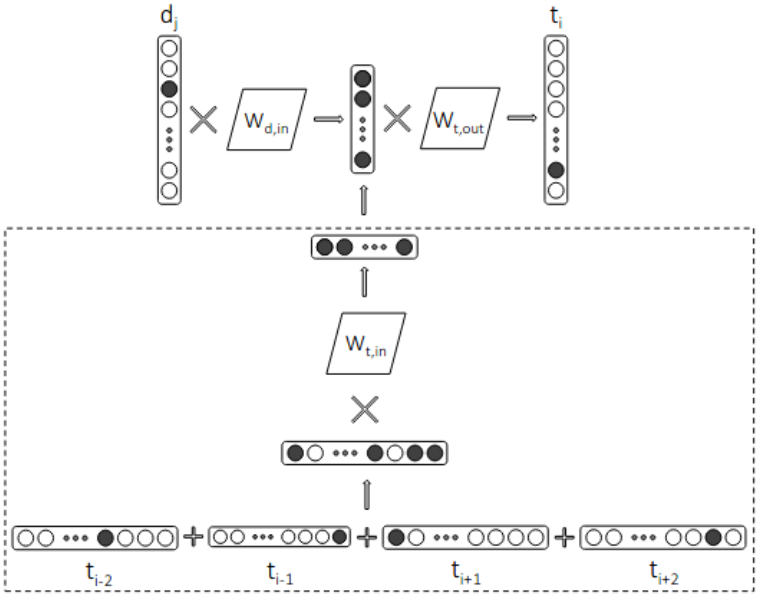


FIGURE 2.2.6 – Architecture du modèle Paragraph2Vec

— **FastText**
Proposé par *Mikolov et al* en 2016 [2], ce modèle est avant tout un modèle de classification de texte. Pour palier au problèmes de dimensions que posent les représentations bag of words en entrée des classifieurs linéaires, le modèle FastText construit une représentation latente de faible dimension qui peut potentiellement être réutilisable [voir figure 2.2.7] et qui est ensuite donnée en entrée à un classifieur linéaire (ex regresseur logistique ou SVM) pour la prédiction d'une classe donnée.
L'architecture du modèle FastText est similaire au modèle CBOW de Mikolov et al (2013) [45]. Cependant, contrairement à ce dernier qui cherche à prédire un terme au milieu d'une fenêtre, FastText cherche à prédire un label (classe).

Aussi, pour introduire une certaine notion d'ordre entre les mots, c'est une représentation bag of n-grams qui est donnée en entrée du modèle plutôt qu'une représentation bag of words.

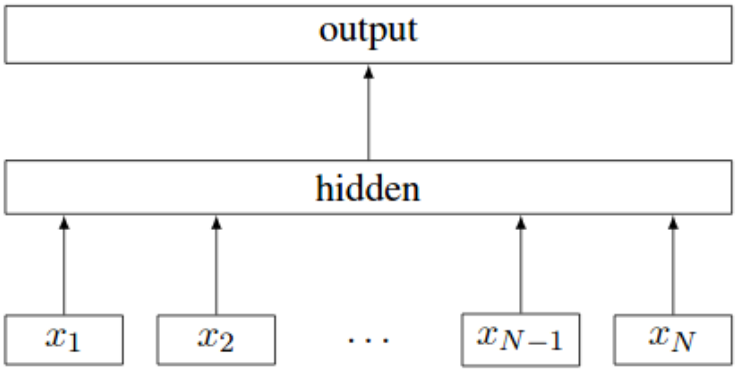


FIGURE 2.2.7 – Architecture du modèle FastText

Pour l'apprentissage de ce modèle, pour un ensemble de N documents, cela revient à minimiser la vraisemblance négative suivante :

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} y_i log f(B.A.x_i)$$

où, x_i est la représentation bag of n-grams du i ème document, y_i le label correspondant, A et B les matrices de poids. Une fonction softmax f est utilisée pour faire apparaître la distribution de probabilité des differentes classes.

— **ElMo**
Proposé par *Peters et al* en 2018 [30],ce modèle utilise des représentations qui dérivent d'un modèle LSTM bidirectionnel couplé à un modèle de langue (LM) et qui sont entrainés sur de larges corpus de textes.

Le modèle ELMo diffère des autres modèles d'embeddings cités plus haut comme [45],[21] dans le sens ou chaque token se voit attribué une représentation qui est en fonction de toutes les couches internes du modèle couplé biLM. Cette idée vient principalement du fait que différentes couches d'un modèle de langue peuvent faire ressortir différents types d'information sur un token [30] (ex les étiquettes morpho-syntaxiques POS sont mieux prédites par les basses couches d'un biLM alors que les désambiguisations lexicales sont mieux encodées par les dernières couches). Concaténer toutes les couches permet donc de combiner une variété de représentations donnant ainsi de meilleures performances.

.

3 Les réseaux de neurones pour la RI

Afin d'ordonner la liste de documents à présenter à l'utilisateur en réponse à une certaine requête, les modèles learning to rank traditionnels emploient des techniques de machine learning sur des caractéristiques (ou features) "fabriquées à la main" [24]. A l'inverse, les modèles neuronaux apprennent des représentations du langage à partir du texte brut pouvant ainsi faire face au problème de différence de vocabulaire entre les documents et les requête, et qui représente le principal obstacle pour les modèles basés sur le comptage des termes [*Mitra et al 2016* [9]] .

Cependant, apprendre une représentation de texte adéquate au problème de ranking requière une énorme quantité de données d'apprentissage avant d'être déployée.
Ainsi, la représentation du texte peut être apprise de manière supervisée ou de manière non supervisée [*Mitra et al 2017* [31]].

L'approche supervisée utilise des données labelisées, comme le score de pertinence pour une paire requête-document ou des recueils de clicks utilisateurs [32], pour apprendre une représentation façonnée et optimisée de bout en bout pour la tache de RI en question. Dans le cas où de pareilles informations ne sont pas disponibles, l'approche non supervisée permet d'apprendre une représentation du texte en se basant uniquement sur les documents/requêtes. Ainsi, le choix d'un modèle de représentation de texte conduit à différentes notions de similarité. Ce choix doit donc être pris avec parcimonie, guidé par la tache de RI à effectuer.

3.1 Modèles de RI basés sur des représentations pré-entraînées

En recherche d'information, la collecte de scores de pertinence de documents par rapport à une requête est une tache extrêmement coûteuse et fastidieuse. Ainsi, il est extrêmement difficile de se procurer des données labélisées [*Mitra et al 2017* [31]].
D'autre part, les avancées dans le domaine de la représentation du texte et la réussite des modèles d'apprentissages d'embeddings, comme le *Word2Vec* [*Mikolov et al* en 2013 [45]], à capturer d'importants liens sémantiques entre les termes a conduit à l'émergence de nouveaux modèles de ranking intégrant l'approche non supervisée basée sur des représentations d'embeddings pré-entraînées. Nous présentons dans cette section le DESM [*Mitra et al 2016* [9]] et le NTLM [*Zuccon et al 2015* [16]] .

3.1.1 DESM : a Dual Embedding Space Model

Cette approche reprend le modèle d'apprentissage du Word2vec via l'architecture CBOW. Cependant, contrairement à la manière dont le modèle est habituellement exploité, on utilise la matrice de poids W_{in} pour construire les embeddings relatifs aux requêtes, et la matrice de poids en sortie W_{out} pour construire les embeddings des documents.
La représentation des requêtes et des documents par des embeddings séparés (représentation IN-OUT) permet la mise en évidence de plus de sémantique distribuée entre les termes qui apparaissent souvent dans les mêmes documents en comparaison avec les représentations partagées IN-IN ou OUT-OUT [9]. La figure 3.1.1 montre que pour ces représentations partagées, les similarités cosinus sont importantes entre les termes ayant une similarité fonctionnelle, tandis que, pour les représentations séparées, les similarités sont importantes entre les termes qui apparaissent souvent ensemble.

La fonction de ranking associée au modèle DESM utilise une similarité cosinus pour comparer chaque IN-embedding de la requête au document représenté par le centroïde des OUT-embeddings des termes présents dans ce document :

$$DESM_{IN-OUT}(Q, D) = \frac{1}{|Q|} \sum_{q_i \in Q} \frac{{q_i}_{IN} . D_{OUT}}{||{q_i}_{IN}||.||D_{OUT}||}$$

où, Q represente la requete et D represente le centroid des embeddings des termes du document.

seahawks			eminem		
IN-IN	OUT-OUT	IN-OUT	IN-IN	OUT-OUT	IN-OUT
seahawks	seahawks	seahawks	eminem	eminem	eminem
49ers	broncos	highlights	rihanna	rihanna	rap
broncos	49ers	jerseys	ludacris	dre	featuring
packers	nfl	tshirts	kanye	kanye	tracklist
nfl	packers	seattle	beyonce	beyonce	diss
steelers	steelers	hats	2pac	tupac	performs

FIGURE 3.1.1 – Les plus proches voisins des mots "$eminem$" et "$seahawks$" en terme de similarité cosinus en fonction des différentes représentations d'embeddings

Ce modèle assure de solides performances dans le ré-ordonnancement des tops documents retournés par un premier modèle basé sur le comptage de termes. Cependant, ses performances se dégradent dés lors que le nombre de documents dans le corpus augmente.
Pour palier à ce problème, une extension de ce modèle consisterait en une mixture du DESM et d'un modèle de matching lexical (eg BM25) dont la formule du calcul du score serait la suivante :

$$MixModel(Q, D) = \alpha DESM_{IN-OUT}(Q, D) + (1 - \alpha)BM25(Q, D)$$

3.1.2 NTLM : Neural Translation Language Model

Ce modèle reprend la fonction de scoring des modèles de translation [8] présentés avec les modèles traditionnels de RI. Cependant, à l'inverse des modèles comme le $TLM - MI$ qui se basent sur une mesure de l'information mutuelle entre les termes pour estimer la probabilité de translation $p(t_q|t_d)$ [*Karimzadeghan et Zhai* [56]], le $NTLM$ calcule une similarité cosinus entre les embeddings des deux termes t_q et t_d [*Zuccon et al 2015* [16]]. La probabilité de translation est obtenue suivant la formule suivante :

$$p(t_q|t_d) = \frac{cos(t_q, t_d)}{\sum_{t \in V} cos(t, t_d)}$$

Pour construire les représentations latentes des termes, Le modèle $NTLM$ se base sur le modèle $Word2Vec$. Il présente une certaine robustesse vis à vis du choix du modèle (*CBOW ou SkipGram*) pour la construction des embeddings.
Aussi, les expérimentations démontrent qu'il n'est pas nécessaire que les représentations des termes soit construites à partir de la même collection utilisée pour le ranking.

3.2 Modèles bout en bout de RI

Les modèles de RI basés sur des représentations pré-entraînées sont idéalement conçus pour retrouver les liens sémantiques entre les termes de la requête et des documents [44],[40]. Cependant, ces modèles ne sont pas spécifiques aux taches de RI, telles que la recherche et la présentation des documents pertinents sachant une requête, car les représentations latentes des termes sont apprises, de manière non supervisée, indépendamment de la tache de ranking [9], [16].
Ainsi, la nécessité de guider la représentation des termes en fonction de la tache de ranking a permis l'essor de nouveaux modèles de deep learning conçus de bout en bout pour la RI. On dénote deux familles de modèles suivant l'approche suivie :

- **Les Modèles basés sur la représentation :** utilisent les réseaux de neurones pour construire une représentation latente pour les requêtes et les documents séparément puis effectuent une correspondance dans cet espace latent [32],[52].

- **Les Modèles basés sur l'interaction :** construisent d'abord des interactions locales entre des pairs de termes de la requête et du document puis utilisent les réseaux de neurones pour apprendre des patterns hiérarchiques pour la correspondance document-requête [20],[26],[22].

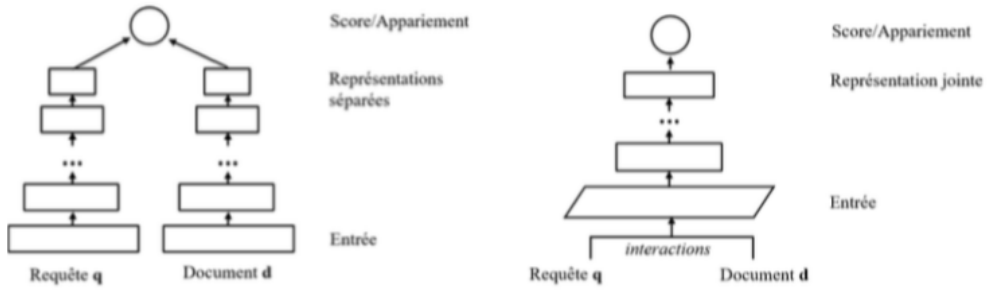


FIGURE 3.2.1 – Approche basée représentation (à gauche) et approche basée interaction (à droite)

3.2.1 Les Modèles basés sur la représentation :

- **DSSM : a Deep Structured Semantic Model**
  Contrairement aux modèles reposants sur des embeddings pré entraînés [9],[16], le modèle DSSM [*Huang et al 2013* [32]] s'appuie sur les données des clics utilisateurs, consistant en une liste de requêtes et les documents "cliqués" correspondants, pour l'apprentissage. Aussi, pour palier au problème de dimentionnalité lié à la taille du vocabulaire, un hashage des termes est opéré projetant les représentations des termes de la requête et des documents dans un espace de dimension inférieure basé sur le comptage des n-grammes.

  Le modèle DSSM repose sur un perceptron multi couches (MLP) pour ordonner un ensemble de documents sachant une certaine requête. D'abord, une projection non linéaire est effectuée pour transformer les vecteurs des termes, préalablement hashés, en des représentations latentes de dimension inférieures. Ensuite, la pertinence de chaque document sachant une requête est calculée à l'aide d'une similarité cosinus entre leurs vecteurs latents.
  Lors de l'apprentissage, les paramètres du modèle sont optimisé pour maximiser la probabilité à posteriori du document clické sachant la requête selon la fonction softmax suivante :

$$P(d|q) = \frac{exp(\lambda cosine(y_q, y_d)}{\sum_{d' \in D} exp(\lambda cosine(y_q, y_{d'})}$$

  où, y_q, y_d les vecteurs latents en sortie du perceptron multi-couches et λ un parametre constant.
  Maximiser cette probabilité à posteriori revient à minimiser la cross-entropie suivante :

$$Loss = -log \prod_{(q,d)} P(d|q)$$

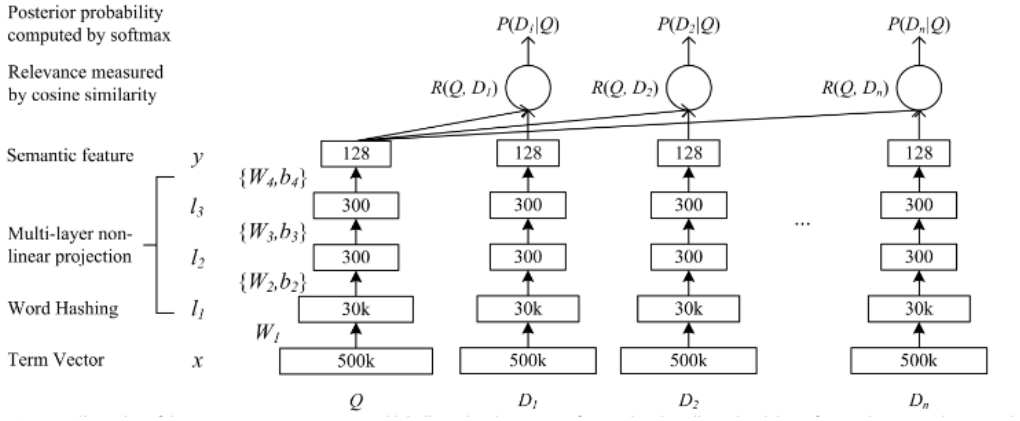


FIGURE 3.2.2 – Illustration du modèle DSSM

- **CLSM :a Convolutional latent semantic model**
  Ce modèle reprend le principe du modèle DSSM. Cependant, à l'inverse de ce dernier, qui manipule une représentation en sac de mots des documents sans prise en compte du contexte, le CLSM [*Shen et al 2014*[52]] est conçu pour capturer le lien sémantique au voisinage d'un terme.Un réseau de neurones convolutionnel est utilisé à la place du perceptron multi couches.

  L'architecture du CLSM est illustrée dans la Figure 3.2.3 ci-dessous. Le modèle contient (1) une couche de n-mots obtenus par application d'une fenêtre de taille n sur la séquence de terme en entrée, (2) une couche de n-grammes qui comme pour le DSSM applique un hashage des termes pour avoir une représentation de dimension inférieure de n-grammes, (3) une couche de convolution qui extrait les caractéristiques contextuelles de chaque mot et son voisinage. (4) une couche d'agrégation qui prend le maximum de chaque dimension parmi tout les vecteurs de n-grammes pour obtenir une unique représentation du document et (5) une couche sémantique qui fournit en sortie du réseau une représentation latente du document.

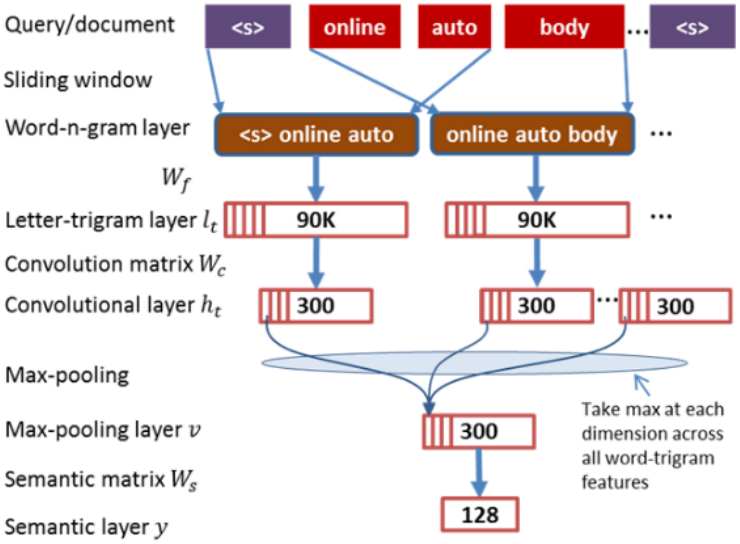


FIGURE 3.2.3 – Illustration du modèle CLSM

Pour l'apprentissage de ce modèle, la même fonction de coût (cross entropie) que le DSSM est utilisée. Cette dernière est optimisée de la même manière.

- **Architecture I (ARC I)**
  L'architecture I, proposé par *Hu et Lu en 2014 [7]* et illustrée dans la figure 3.2.4 ci-dessous, se base sur une approche convolutive.

  D'abord, chaque texte est présenté en entrée à un réseau de neurones CNN qui, au travers des différentes séries de convolution et de pooling, apprends une représentation du texte en entrée.
  Les représentations des deux textes en sortie du CNN sont ensuite données en entrée à un perceptron multi couches (MLP) pour le calcul du score de correspondance.

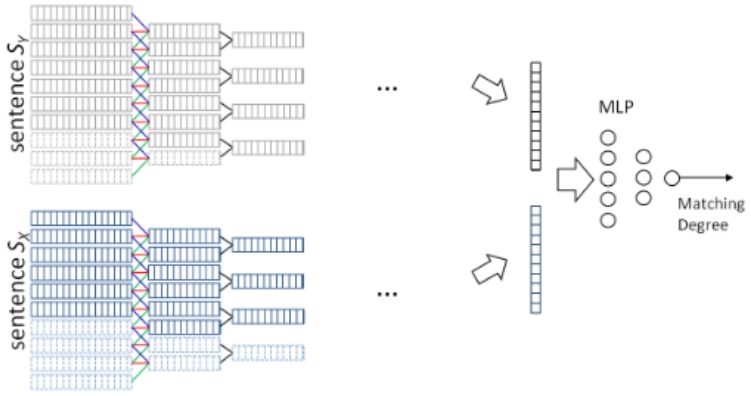


FIGURE 3.2.4 – Illustration du modèle ARC I

3.2.2 Les Modèles basés sur l'interaction :

- **DRMM : a Deep Relevance Matching Model**
  L'architecture du DRMM [*Guo et al 2017* [20]], comme illustrée dans la figure 3.2.5 ci-dessous, construit tout d'abord les interactions locales entre chaque terme de la requête et du document. Ces interactions sont calculées en faisant une similarité cosinus entre les embeddings des deux termes. Pour chaque terme de la requête, on transforme son vecteur d'interaction (de taille variable selon le nombre de termes du document) en un histogramme de taille fixe.Ensuite, en se basant sur cet histogramme, un perceptron multi couche est employé pour apprendre les différents patterns de correspondance entre le terme d'une requête et ceux du document et calculer un score de correspondance pour chaque terme de la requête. Enfin, les scores en sortie des différentes MLP, correspondant à chaque terme de la requête, sont donnés en entrée à un modèle d'agrégation qui fournit en sortie le score de pertinence du document sachant la requête.

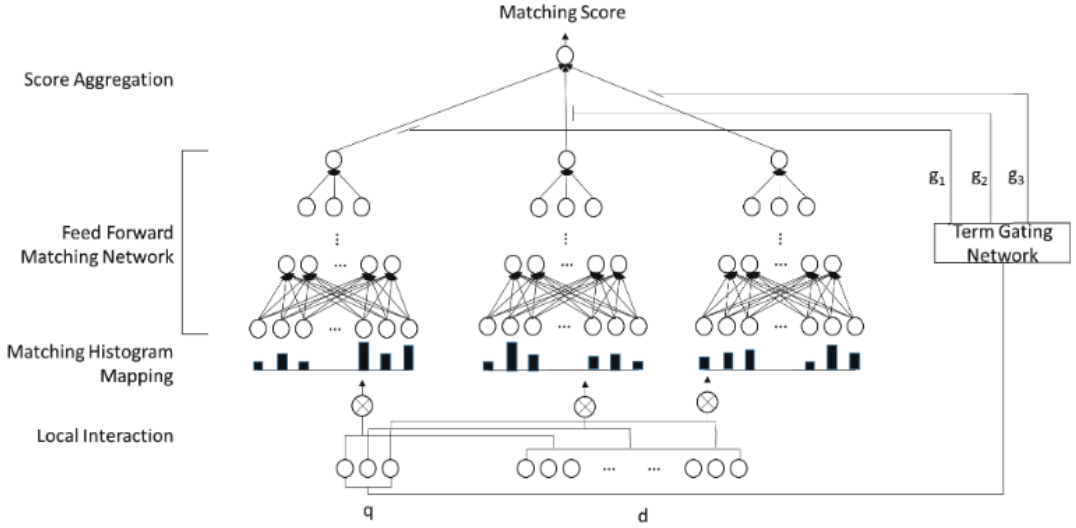


FIGURE 3.2.5 – Illustration du modèle DRMM

Lors de l'apprentissage de ce modèle, une fonction de cout *hinge loss* est optimisée. Étant donnée un triplet, (q,d^+,d^-), où le document d^+ est classé plus haut que le document d^- sachant une requête q, la fonction de coût est définie comme suit :

$$Loss(q, d^+, d^-, \Theta) = max(0, 1 - score(q, d^+) + score(q, d^-))$$

où, Θ englobe les paramètres du MLP et du modèle d'agrégation.

- **MatchPyramid**
  Proposé par *Pang et Lan* en 2016 [22], ce modèle est inspiré des modèles convolutifs utilisés en vision et modélise le problème de matching de textes comme un problème de reconnaissance d'image.

  L'architecture MatchPyramid, comme illustrée dans la figure 3.2.6 ci-dessous, construit tout d'abord les interactions locales entre chaque terme de la requête et du document sous forme d'une matrice d'interaction.
  Comme pour le modèle DRMM, les interactions sont calculées par similarité cosinus entre les embeddings des deux termes. Cette matrice est ensuite vue comme une image qu'on donne en entrée à un réseau de neurones CNN qui, couche après couche, capture des patterns de correspondance sémantique entre les textes.

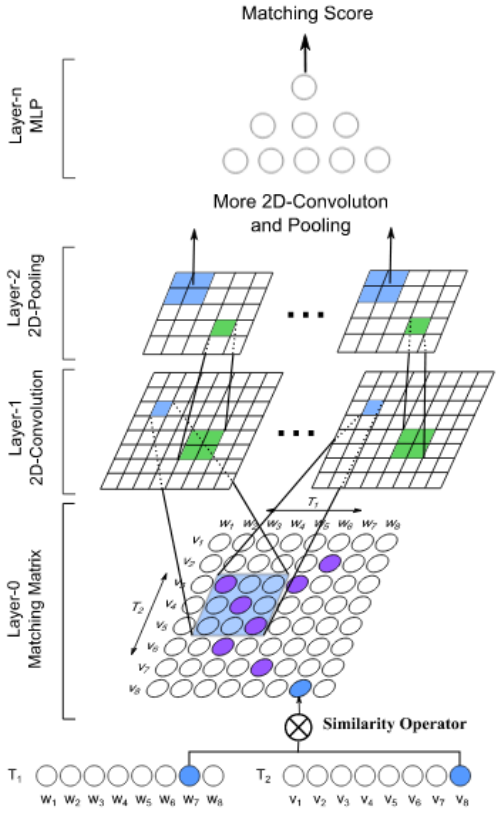


FIGURE 3.2.6 – Illustration du modèle MatchPyramid

Afin de produire le score de correspondance final , un perceptron multi couches (MLP) est utilisé et ses parametres sont optimisés pour minimiser une cross entropie qui, dans le cas d'une classification binaire (matching/NONmatching) se définit comme suit :

$$Loss = -\sum_i [y^{(i)} log p_1^{(i)} + (1 - y^{(i)}) log p_0^{(i)}]$$

$$p_k = \frac{\exp(S_k)}{\exp(S_0)+\exp(S_1)}, k = 0, 1$$

où, $y^{(i)}$ est la classe du i ème exemple d'entraînement,S_0, S_1 sont les scores de correspondance des deux classes.

- **Architecture II (ARC II)**
  Pour palier au risque de pertes de détails lié à la rencontre tardives dans l'architecture ARC I des textes a faire correspondre, *Hu et Lu* ont proposé l'architecture ARC II [7], qui est construite directement dans l'espace d'interaction entre les textes.

  Comme illustré dans la figure 3.2.7, la première couche du réseau prend en entrée une fenêtre opérant sur chaque texte et modélise les possibles combinaisons entre ces fenêtres au travers de convolutions unidimensionnelles (1D). Un max pooling 2D est ensuite opéré suivie d'une série de convolutions et de pooling bidimensionnels (2D).
  La derniere couche de l'architecture, comme pour le modèle ARC I, utilise un perceptron multi couche pour le calcul d'un score de correspondance.

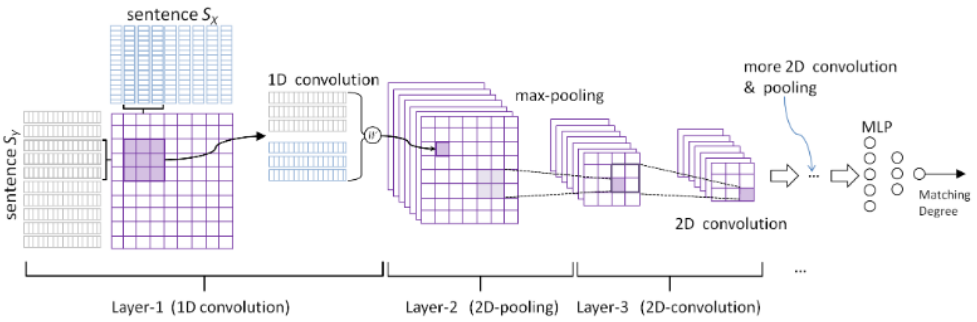


FIGURE 3.2.7 – Illustration du modèle ARC II

4 Vers des modèles de RI neuronaux sans ré-ordonnancement

Les modèles neuronaux conçus de bout en bout pour la tache de ranking ont démontré de solides performances dans différentes taches de RI [31]. Cependant, la plupart de ces modèles partagent un même inconvénient : leur performance se dégrade avec l'accroissement du nombre de documents dans la collection. Étant donné que ces modèles de ranking se basent sur des représentations denses, le calcul d'un score de pertinence pour chaque document d'un large corpus devient très vite infaisable [17]. Ainsi, en pratique, ces modèles sont employés uniquement pour le ré-ordonnancement d'un sous-ensemble de documents potentiellement pertinents pour une certaine requête, ce sous-ensemble de documents étant pré-sélectionné par un premier modèle efficient (ex BM25 [41], ML [36]).

L'usage d'un modèle d'ordonnancement à deux (ou plusieurs) étapes donne l'avantage d'allier l'efficience des modèles traditionnels à l'efficacité des modèles neuronaux. Cependant, la première étape peut agir comme une barrière limitant l'ensemble de documents pertinents à donner en entrée de la seconde étape rendant impossible la découverte par le modèle neuronal de nouveaux documents pertinents ou en introduisant et en propageant les erreurs [17].
Pour palier à ces différents problèmes, une nouvelle famille de modèles neuronaux a vu le jour avec pour but l'ordonnancement à grande échelle.

4.1 SNRM : a Standalone Neural Ranking Model

Introduit par *Zamani et al* en 2018 [17], ce modèle propose plutôt que d'apprendre des représentations denses, qui sont à l'origine des problèmes d'efficience, d'apprendre des représentations parcimonieuses (sparse) des textes. Ce choix de représentation est motivé par le fait qu'à l'inverse des représentations denses sous l'angle desquelles on calcule un score pour tous les documents du corpus, une représentation sparse permet de diminuer largement le nombre de documents à faire correspondre (voir figure 4.1.1), ce qui permet un large gain d'efficience. Dès lors, le modèle SNRM a pour objectif l'optimisation de deux critères : (*i*) un objectif de pertinence qui maximise l'efficacité et (*ii*) un objectif de parcimonie ayant pour effet de maximiser l'efficience de la recherche [3].

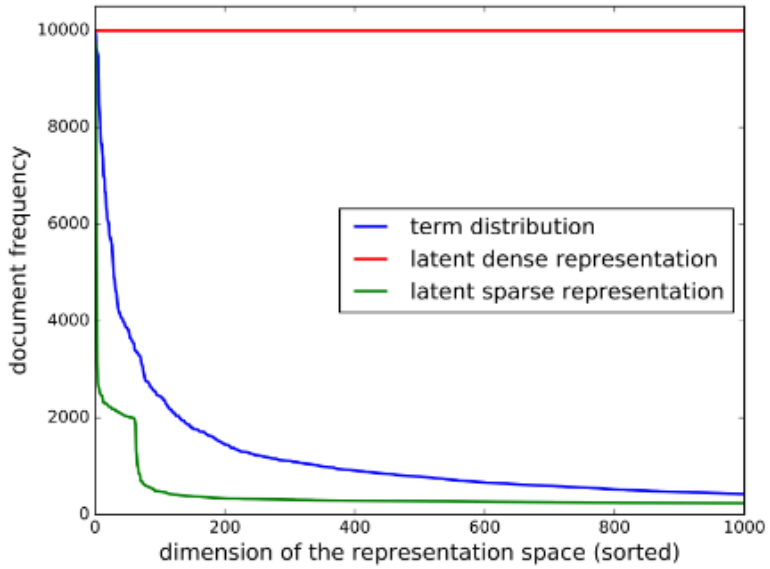


FIGURE 4.1.1 – Fréquence des documents à faire correspondre en fonction de la dimension de représentation (Sparse vs Dense)

L'architecture du modèle SNRM, comme illustrée dans la figure 4.1.2 ci-dessous, est basée sur l'apprentissage de représentations parcimonieuses en grande dimension de n-grams. D'abord, les embeddings de chaque terme du n-gram sont collectés (des embeddings issus du modèle word2vec par exemple). Ces n vecteurs latents sont ensuite concaténés puis donnés en entrée à un perceptron multi couches en forme de sablier. Cette structure, dans un premier temps, force la compression de l'information donnée en entrée du modèle permettant ainsi d'apprendre sa sémantique. Le nombre de neurones augmente ensuite dans les dernières couches permettant d'avoir en sortie une représentation parcimonieuse en grande dimension d'un n-gram. Enfin, la représentation finale du texte est obtenue en calculant la moyenne des représentations des différents n-grams.
Une fois les représentations entraînées on construit un index inverse de chaque terme latent vers les documents concernés.

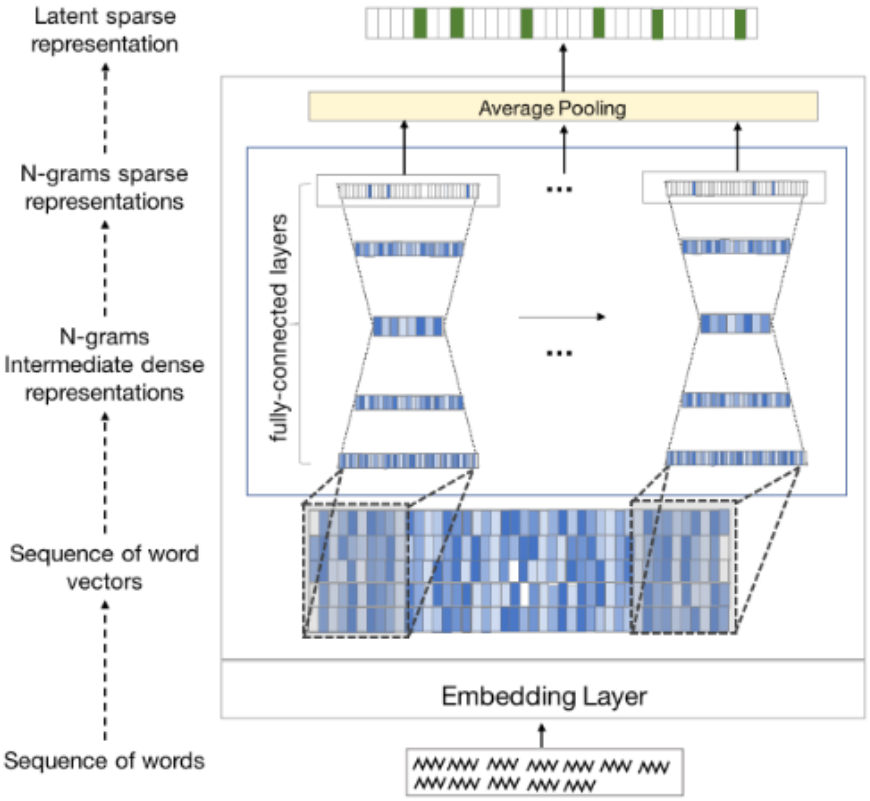


FIGURE 4.1.2 – Illustration du modèle SNRM

Le modèle SNRM, est entraîné par faible supervision [28] suivant deux objectifs :

— **Un objectif de pertinence :** On entraîne le modèle selon un approche pairwise avec $y_i \in \{-1, 1\}$ selon la pertinence du document d_{i1} par rapport au document d_{i2}.
Ainsi, une fonction Hinge loss est employée :

$$L = max\left\{0, \epsilon - y_i[v_{q_i}.v_{d_{i1}} - v_{q_i}.v_{d_{i2}}]\right\}$$

— **Un objectif de parcimonie :** maximiser la parcimonie de la représentation revient à minimiser la norme L_1 :

$$L_1(v) = \sum_{k=1}^{|v|} |v_k|$$

Au final, nous obtenons la fonction de coût suivante :

$$Loss(q_i, d_{i1}, d_{i2}, y_i) = L(q_i, d_{i1}, d_{i2}, y_i) - \lambda L_1(q_i||v_{d_{i1}}||v_{d_{i2}})$$

où, $||$ signifie la concaténation. L'hyper-paramètre λ contrôle le degré de parcimonie de la représentation.

Au moment de l'inférence, étant donné une requête, on transforme cette dernière en une représentation parcimonieuse de grande dimension puis, étant donné le nombre restreint d'éléments non négatifs dans la représentation, on retrouve les documents dans le large corpus à l'aide de l'index inverse construit et un score de pertinence est calculé par produit cartésien entre la représentation de la requête et du document(voir figure 4.1.3).

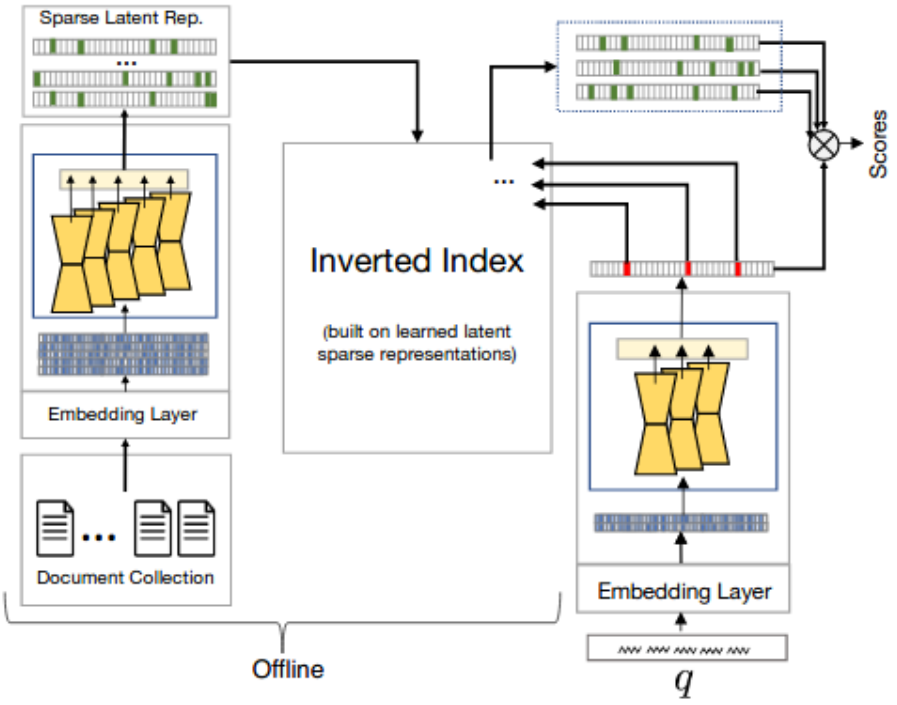


FIGURE 4.1.3 – Moment de l'inférence

Références

[1] et D. B. Rubin A. P. DEMPSTER N. M. Laird. « Maximum Likelihood from Incomplete Data Via the EM Algorithm ». In : *Journal of the Royal Statistical Society: Series B* 39.1 (1977), p. 1-22. DOI : 10.1111/j.2517-6161.1977.tb01600.x..

[2] T.Mikolov A.JOULIN E.Grave. « Bag of Tricks for Efficient Text Classification ». In : *arXiv:1607.01759 [cs]* (2016).

[3] J. Guo et AL. « A Deep Look into Neural Ranking Models for Information Retrieval ». In : *arXiv:1903.06902 [cs]* (2019).

[4] C. Burges et AL. « Learning to rank using gradient descent ». In : *The 22nd international conference on Machine learning ICML '05, Bonn, Germany* (2005), p. 89-96. DOI : 10.1145/1102351.1102363.

[5] N. Abdul-Jaleel et AL. *UMass at TREC 2004: Novelty and HARD*. Rapp. tech. Jan. 2004.

[6] S. Galeshchuk et B. CHAVES. « Modélisation thématique à l'aide des plongements lexicaux issus de Word2Vec ». In : ().

[7] H. Li B.HU Z. Lu. « Convolutional Neural Network Architectures for Matching Natural Language Sentences ». In : *arXiv:1503.03244 [cs]* (2014).

[8] A. BERGER et J. LAFFERTY. « Information Retrieval as Statistical Translation ». In : *ACM SIGIR Forum* 51.2 (1999), p. 222-229.

[9] Nick Craswell BHASKAR MITRA Eric Nalisnick. « A Dual Embedding Space Model for Document Ranking ». In : (2016).

[10] R. Ragno C. BURGES et Q. LE. « Learning to rank with non smooth cost functions ». In : *Advances in Neural Information Processing Systems*. MIT Press. 2006.

[11] K. Lund et C. BURGESS. « Producing high-dimensional semantic spaces from lexical co-occurrence ». In : *Behavior Research Methods, Instruments, Computers* 28.2 (1996), p. 203-208. DOI : 10.3758/BF03204766.

[12] G. H. Golub et C. REINSCH. « Singular value decomposition and least squares solutions ». In : (1970).

[13] et D. C. Plaut D. L. T. ROHDE L. M. Gonnerman. « An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence ». In : (2006).

[14] Andrew Y. Ng et Michael I. DAVID M. BLEI. « « Latent dirichlet allocation » ». In : *Journal of machine Learning research* (2003), p. 993-1022.

[15] et S. Robertson G. CAO J. Gao. « Selecting good expansion terms for pseudo-relevance feedback ». In : *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '08, Singapore, Singapore* (2008), p. 243-250. DOI : 10.1145/1390334.1390377.

[16] B. Koopman G. ZUCCON et P. BRUZA. « Integrating and Evaluating Neural Word Embeddings in Information Retrieval ». In : *the 20th Australasian Document Computing Symposium on ZZZ - ADCS '15, Parramatta, NSW, Australia* (2015). DOI : 10.1145/2838931.2838936.

[17] W. B. Croft H. ZAMANI M. Dehghani. « From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing ». In : *the 27th ACM International Conference on Information and Knowledge Management - CIKM '18, Torino, Italy* (2018), p. 497-506. DOI : 10.1145/3269206.3271800.

[18] Z. S. HARRIS. « Distributional Structure ». In : *WORD* 10.2-3 (1954), p. 146-162. DOI : 10.1080/00437956.1954.11659520.

[19] T. HOFMANN. « Probabilistic Latent Semantic Indexing ». In : *ACM SIGIR Forum* 51.2 (1999).

[20] et W. B. Croft J. GUO Y. Fan. « A Deep Relevance Matching Model for Ad-hoc Retrieval ». In : *the 25th ACM International on Conference on Information and Knowledge Management - CIKM '16* (2017), p. 55-64. DOI : 10.1145/2983323.2983769.

[21] et C. Manning J. PENNINGTON R. Socher. « Glove: Global Vectors for Word Representation ». In : *the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar* (2014), p. 1532-1543. DOI : 10.3115/v1/D14-1162.

[22] J. Guo L. PANG Y. Lan et J. XU. « Text matching as image recognition ». In : (2016).

[23] J. LAFFERTY et C. ZHAI. « Document language models, query models, and risk minimization for information retrieval ». In : *The 24th annual international ACM SIGIR conference on Research and development in information retrieval* (2001), p. 111-119.

[24] Hang LI. « A Short Introduction to Learning to Rank ». In : (2010).

[25] T.-Y. LIU. *Learning to Rank for Information Retrieval*. Berlin, Heidelberg: Springer Berlin. Foundation et Trends in Information Retrieval, 2011.

[26] Z. LU et H. LI. « A Deep Architecture for Matching Short Texts ». In : (2013).

[27] et G. Kruszewski M. BARONI G. Dinu. « Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors ». In : *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* 1 (2014), p. 238-247. DOI : 10.3115/v1/P14-1023.

[28] A. Severyn M. DEHGHANI H. Zamani. « Neural Ranking Models with Weak Supervision ». In : *arXiv:1704.08803 [cs]* (2017).

[29] G. Salton et M. J. MCGILL. *Introduction to modern information retrieval*. computer science series. McGraw-Hill, 1983.

[30] C.Clark M.PETERS M.Neumann. « Deep contextualized word representations ». In : *arXiv:1802.05365 [cs]* (2018).

[31] B. Mitra et N. CRASWEL. *Neural Models for Information Retrieval*. 2017.

[32] J. Gao P. HUANG X. He. « Learning deep structured semantic models for web search using clickthrough data ». In : *the 22nd ACM international conference on Conference on information knowledge management - CIKM '13, San Francisco, California, USA* (2013). DOI : 10.1145/2505515.2505665.

[33] E. Cosijn et P. INGWERSEN. « Dimensions of relevance ». In : *Information Processing Management* 36.4 (2000), p. 533-550. DOI : 10.1016/S0306-4573(99)00072-2.

[34] et C. J. Burges P. LI Q. Wu. « McRank: Learning to Rank Using Multiple Classification and Gradient Boosting ». In : *Advances in Neural Information Processing Systems 20* (2008).

[35] P. D. Turney et P. PANTEL. « From Frequency to Meaning: Vector Space Models of Semantics ». In : *Journal of Artificial Intelligence Research* 37 (2010), p. 141-188. DOI : 10.1613/jair.2934.

[36] J. M. PONTE et W. B. CROFT. « A language modeling approach to information retrieval ». In : *SIGIR98: 21st Annual ACM/SIGIR International Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, 1998.

[37] et J. Gao Q. WU C. J. C. Burges. « Adapting boosting for information retrieval measures ». In : *Information Retrieval* 13.3 (2010), p. 254-270. DOI : 10.1007/s10791-009-9112-1.

[38] T. Graepel R. HERBRICH et K. OBERMAYER. *Large Margin rank boundaries for ordinal regression*. MIT Press, Cambridge. 2000.

[39] J. ROCCHIO. « XXIII. RELEVANCE FEEDBACK IN INFORMATION RETRIEVAL ». In : (1971).

[40] G. W. Furnas S. DEERWESTER S. T. Dumais et T. K. LANDAUER. « Indexing by latent semantic analysis ». In : *Journal of the American Society for Information Science* 41.6 (1990), p. 391-407. DOI : 10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9.

[41] S. E. Robertson et S. WALKER. « Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval ». In : *SIGIR '94*. London: Springer. 1994, p. 232-241.

[42] A. SHASHUA et A. LEVIN. « Ranking with large margin principle: Two approaches ». In : *Advances in Neural Information Processing Systems*. MIT Press. 2003.

[43] et B. Metahri T. FRANCESIAZ R. Graille. « Introduction aux modèles probabilistes utilisés en Fouille de Données ». In : (2015).

[44] et J. Dean T. MIKOLOV K. Chen. « Efficient Estimation of Word Representations in Vector Space ». In : *arXiv:1301.3781 [cs]* (2013).

[45] et J. Dean T. MIKOLOV K. Chen. « Efficient Estimation of Word Representations in Vector Space ». In : *arXiv:1301.3781 [cs]* (2013).

[46] Q. Le et T. MIKOLOV. « Distributed Representations of Sentences and Documents ». In : (2014).

[47] D. Cossock et T. ZHANG. « Subset Ranking Using Regression ». In : *Learning Theory, Berlin, Heidelberg* (2006), p. 605-619. DOI : 10.1007/11776420_44.

[48] V. Lavrenko et W. B. CROFT. « Relevance based language models ». In : *The 24th annual international ACM SIGIR conference on Research and development ininformation retrieval. ACM* (2001), p. 120-127.

[49] D. Metzler W.B. CROFT et T. STROHMAN. *Search Engines -Information Retrieval in Practice*. 2009.

[50] J. XU et H. LI. « AdaRank: A Boosting Algorithm for Information Retrieval ». In : *the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (2007).

[51] O. Levy et Y. GOLDBERG. « Linguistic Regularities in Sparse and Explicit Word Representations ». In : *Proceedings of the Eighteenth Conference on Computational Natural Language Learning, Ann Arbor, Michigan* (2014), p. 171-180. DOI : 10.3115/v1/W14-1618.

[52] J. Gao Y. SHEN X. He. « Learning semantic representations using convolutional neural networks for web search ». In : *the 23rd International Conference on World Wide Web - WWW '14 Companion, Seoul, Korea* (2014), p. 373-374. DOI : 10.1145/2567948.2577348.

[53] F. Radlinski Y. YUE T. Finley. « A support vector method for optimizing average precision ». In : *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '07, Amsterdam* (2007), p. 271-278. DOI : 10.1145/1277741.1277790.

[54] G. YU C. T.; Salton. « Precision Weighting – An Effective Automatic Indexing ». In : *Journal of the ACM* (1976).

[55] T.Y. Liu Z. CAO T. Qin. « Learning to rank: from pairwise approach to listwise approach ». In : *The 24th international conference on Machine learning - ICML '07, Corvalis, Oregon* (2007), p. 129-136. DOI : 10.1145/1273496.1273513.

[56] M. Karimzadehgan C. ZHAI. « Estimation of statisticaltranslation models based on mutual information for ad hocinformation retrieval ». In : *SIGIR'10* (2010), p. 323-330.