

Projet RITAL - Reproduction de Papier

On the Benefit of Incorporating External Features in a Neural
Architecture for Answer Sentence Selection

Cai Gautier, Keddiss Adam, Théologien Antoine (Les Pupuces)

May 16, 2025

- **Sujet principal** : Sélection de réponse (Answer Sentence Selection) dans les systèmes Question/Réponse (QA).
- **Observation clé** : Les architectures neuronales négligent souvent les features externes.
- **Hypothèse** : Les features externes peuvent compléter les réseaux neuronaux.
- **Contribution** : Ajouter des features simples à un réseau convolutionnel (CNN) améliore les performances.

- Expérimentation sur deux benchmarks reconnus : **WikiQA** et **TRECQA**.
- Le Deep Learning domine les approches QA (CNN, LSTM, attention).
- Les anciennes features utiles sont souvent ignorées.
- L'article explore si elles sont toujours pertinentes.
- Comparaison de CNN seul et CNN + features à d'autres modèles de l'état de l'art.

- **Architecture CNN** : bi-CNN inspirée de Severyn et Moschitti.
- **Embeddings** : 50D (Wikipedia), 300D (Google News).
- **21 features externes**, en 3 catégories :
 - Lexical/semantic matching : BM25, Word2Vec, ESA, TAGME.
 - Readability : syllabes, mots complexes, Dale-Chall.
 - Focus : MatchedNGram sur les head words.
- **Fusion** dans la couche fully-connected.

- **Amélioration significative** sur TREC QA et WikiQA.
- **+10.8% en S@1** sur WikiQA dans certaines configs.
- **Compétitif avec PairwiseRank, ABCNN-3.**
- Montre que les CNN seuls ne capturent pas tout.
- Soutient l'usage conjoint des features classiques + DL.

- **Résultats non systématiquement meilleurs.**
- **Dépendance au dataset/config (dropout, embeddings).**
- **Features simples**, peu d'analyse approfondie de leur contribution.
- **Portée limitée** à une architecture CNN.

- Expérimentation via différents paramètres : dropout, mise en place d'un module d'attention, embedding
- Dans tous les cas : meilleurs résultats sur **TRECQA**, sans attention et avec dropout
- **TRECQA** : MAP allant de -3% à +2.6%, MRR de -3.8 à +3.5
- **WIKIQA** : MAP allant de -2% à +7.2%, MRR de -2.5 à +6.9

- **Récupération du code** : Dépôt GitHub fourni par les auteurs.
- **Premier problème** : Deux implémentations distinctes :
 - Une version en Python 2.
 - Une autre en Python 3, mais incomplète.
- **TRECQA** : Dataset déjà formalisé et prêt à l'emploi.
- **WIKIQA** : Format non pris en charge : travail de préparation nécessaire.

- **Second problème** : Partie du code manquante.
 - Génération des identifiants des réponses absente.
- **Notre solution** : Génération via un algorithme simple :
 - Utilisation d'un dictionnaire Réponse : Identifiant.

System	Attn?	Drop?	TREC QA		WikiQA	
			MAP	MRR	MAP	MRR
Runs (AQUAINT/Wikipedia)						
CNN	No	No	75.25	80.90	26.37	26.84
Combined Model	No	No	77.44	82.19	23.46	23.86
Combined Model	No	Yes	77.44	82.19	23.46	23.86
CNN	Yes	No	75.37	80.80	25.09	25.49
Combined Model	Yes	No	76.59	81.45	25.23	25.66
Combined Model	Yes	Yes	76.59	81.45	25.23	25.66
Runs (Google News)						
CNN	No	No	75.18	80.57	25.87	26.24
Combined Model	No	No	76.41	82.90	24.82	25.22
Combined Model	No	Yes	76.41	82.90	24.82	25.22
CNN	Yes	No	74.15	80.60	25.43	25.95
Combined Model	Yes	No	75.16	79.68	25.34	25.87
Combined Model	Yes	Yes	75.16	79.68	25.34	25.87

- Résultats à relativiser : très petit échantillon
- **Méthode** : Question avec réponses : ordonnancement des réponses
- **Résultat** :

- L'ajout des features apportent bien des améliorations sur le TREC QA.
- Mauvais résultats sur WikiQA : provient sans doute d'une mauvaise reconstruction des données
- Couche d'attention moins efficace comme dans le papier original
- Dropout qui ne change rien : on a pourtant testé avec différentes valeurs...

Comment représenter la sémantique ?

- LSA ?
 - 1 Réduction de bruit.
 - 2 Pas de contexte.
 - 3 Embedding de document.
- Word2vec ?
 - 1 Contextuelle.
 - 2 Nécessite un grand corpus.
 - 3 Embedding de mot.

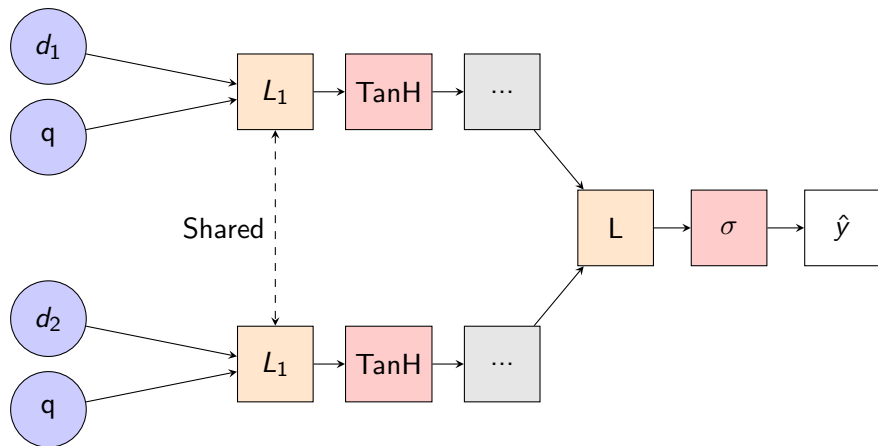
Finalement, on optera pour un LSA.

Le LSA réduit les dimensions, donc pour garder cette avantage, on vectorise en token de taille 5 à 7:

- Gestion des mots nouveaux.
- Sémantique des "racines".

On obtient **1.4M dimensions** au total, puis on applique le LSA pour conserver **10K dimensions**.

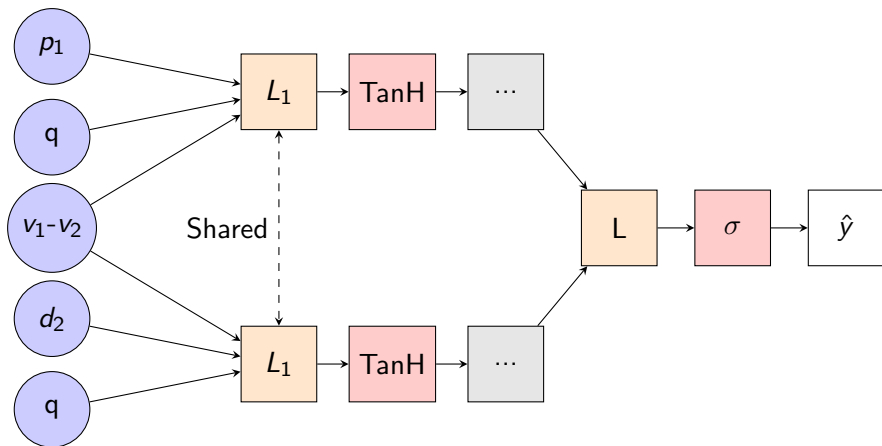
Un document devient donc la moyenne de ses phrases (AVG pool).



Maintenant on "sait" trouver le document, mais la phrase ?

- Point-wise: Pas d'interactions.
- Pair-wise: Interactions et polyvalent.
- List-wise: Encore plus d'interactions mais moins polyvalent.

On opte finalement pour le **pair-wise** faute de temps/matériel.



Qu'est ce que v_i ?

On s'inspire de l'article ...

- Length
- ExactMatch
- Alignement max.
- ...
- Similarité Cosinus.

Finalement, $v_i \in \mathbb{R}^{17}$, l'idée est que $v_i - v_j$ puisse encoder le plus proche de la query.

One vs One (document)

- Accuracy: 67.02%
- F1: 0.56

One vs One (phrase)

- MRR : 14.89
- MAP : 11.37

Quelle feature et quel impact ?

En mettant simplement les deux features à 0, donc la soustraction n'apportera aucune information, on peut déduire quelle feature a de l'importance dans la prédiction ...

Sans surprise:

- ExactMatch : $\text{MRR} \pm 1.23$
- Alignement max: $\text{MRR} \pm 0.56$

Les améliorations:

- Vers un modèle *end-to-end*.
- Embedding des mots.
- List-Wise, donc un réseau par document ?

- Veille scientifique.
- Ré implémenter des articles.
- Essayer des hypothèses.

Merci pour votre attention !
(Et le semestre)