Apprentissage de citations légales

Entreprise: Lexum Informatique Juridique inc.

Code projet PARI: Q2018-1488 (2ème bloc)

Conseiller technologique du Mila: Gaétan Marceau Caron

Nombre d'heures: 50

Définition de la tâche

Description du problème

La tâche consiste à inférer la relation de citation légale entre deux documents de la base de données CanLII. Cette base représente 2M de documents légaux en anglais et en français. Pour ce projet, nous allons nous restreindre aux 1M de documents en anglais. Tout d'abord, nous construisons l'ensemble de test en isolant 100 000 documents (10% du corpus).

Nous sommes dans un cadre d'apprentissage supervisé où la base d'apprentissage est composée d'une liste de paires de documents (A,B) avec leurs étiquettes respectives qui déterminent s'il y a une relation de citation de A vers B. Des méta-informations associées aux documents ainsi qu'au graphe de citations global sont prises en compte. Les méta-informations associées aux documents sont:

- *juridiction_id* (citant): 14 valeurs possibles indiquant la province ou au niveau canadien,
- *histogram_count* (cité): le nombre de citations pour chaque année,
- *decision_date* (cité): la date à laquelle la décision a été rendue,
- *decision_collection_id* (cité): 250 valeurs possibles représentants les différentes cours et leurs domaines de compétence,
- *citation_date* (module Film): date à laquelle la citation a eu lieu.

Les méta-informations associées au graphe de citations sont obtenues à partir d'un voisinage autour des deux documents concernés.

Mesures de performance

Les mesures de performance visent à évaluer la capacité du modèle à prédire la relation de citation entre deux documents. Nous utilisons des mesures de performance issues des travaux en *recherche d'informations* (*information retrieval*), telles que la *precision*, *recall*, *Normalized Discounted Cumulative Gain* (NDCG) et *mean Averaged Precision* (mAP). Tout d'abord, nous devons définir la notion de requête. Une *requête* est une paire de documents (A, B) associée à un score retourné par le modèle, **qui est supérieur à un seuil fixé**. Par exemple, le score peut être la probabilité de citation. Pour un document citant A donné, on considère $\widehat{R}_A$ l'ensemble des requêtes ordonnées en ordre décroissant par rapport au score supérieur à un seuil. On note R_A l'ensemble des requêtes associées au document citant A où B est un document cité. Nous allons calculer des mesures de performances sur $\widehat{R}_A$ en considérant seulement les k premières requêtes, où k dépend du nombre de requêtes affichées à l'utilisateur. La restriction de $\widehat{R}_A$ aux k requêtes avec les scores les plus élevés sera nommé $\widehat{R}_A@k$.

Noter qu'en augmentant le seuil, le modèle peut retourner un nombre de requêtes inférieur à k $(|\widehat{R}_A@k| < k)$. Dans le cas où le nombre de requêtes pertinentes est supérieur à k $(|R_A| \geq k)$, la mesure de recall va diminuer. Par contre, la mesure de precision@k peut artificiellement augmenter (le dénominateur sera plus petit). En réduisant le seuil, le modèle va retourner le nombre maximal de requêtes k, et par conséquent, le recall va augmenter au détriment de la précision.

Nous considérons les métriques suivantes:

- $recall@k(\widehat{R}_A) = \frac{|\widehat{R}_A@k \cap R_A|}{|R_A|}$
- $precision@k(\widehat{R}_A) = \frac{|\widehat{R}_A@k \cap R_A|}{|\widehat{R}_A@k|}$

Le *Normalized Discounted Cumulative Gain* (NDCG) est une mesure de performance qui prend en compte l'ordre des requêtes. La formule sans normalisation est:

- $$DCG@k = \sum_{i=1}^{|\widehat{R}_A@k|} \frac{Ind(\widehat{R}_A@k[i] \in R_A)}{log_2(i+1)}$$

où $Ind(\widehat{R}_A@k[i] \in R_A)$ est une fonction qui retourne 1 si la i-ème requête $\widehat{R}_A@k[i]$ appartient à R_A et 0 sinon. La mesure est maximale si tous les documents cités sont en haut de la liste de documents retournés. Notez que $DCG@k$ est égale à *precision@1*. Tout comme le recall@k, cette mesure n'est pas sensible à k lorsque le modèle a retourné tous les documents cités. Maintenant, la mesure NDCG divise le DCG par une constante qui représente le meilleur score possible soit:

- $$IDCG@k = \sum_{i=1}^{min(|R_A|,k)} \frac{1}{log_2(i+1)}$$

Par conséquent, NDCG est une mesure normalisée dans l'intervalle [0,1]:

- $$NDCG@k = \frac{DCG@k}{IDCG@k}$$

Le *Average Precision* (mAP) se calcule en moyennant les *precision@k* qui correspondent à un changement de la valeur de recall, ce qui correspond à trouver un nouveau document cité. La moyenne est prise par rapport au nombre total de documents pertinents.

- $$AP = \frac{1}{|R_A|} \sum_{k \in K} \frac{|\widehat{R}_A@k \cap R_A|}{|\widehat{R}_A@k|}$$

où $K = \left\{k \in \left\{1, |\widehat{R}_A|\right\} \; ; \; recall@k(\widehat{R}_A) > recall@k-1(\widehat{R}_A)\right\}$

Pour calculer K, on considère que k=1 appartient à K et on vérifie la condition à partir de k=2.
Par contre, cette métrique est contestée en raison de plusieurs problèmes expliqués dans [1], notamment qu'elle n'est pas normalisée correctement. Il faudrait donc revoir sa pertinence dans l'évaluation des modèles.

Toutes ces mesures sont moyennées sur tous les documents citants A. Le seuil de décision est déterminé par rapport au F0.5 sur un ensemble de validation qui possède la même distribution que l'ensemble de test.

Protocole expérimental

Préparation des données

Le prétraitement du corpus consiste à enlever les mots de taille inférieure à 3 lettres, les accents, les caractères spéciaux. Les mots restants sont ensuite stemmés et seulement les mots avec 1000 occurrences et plus sont conservés. Après ce filtrage, le vocabulaire contient 22 689 mots.

La génération des embeddings de documents (ou de paragraphes) consiste à utiliser l'algorithme doc2vec afin de générer de façon statique tous les vecteurs associés aux documents. La taille des document embeddings est fixé à 1000. Dans l'article original, les auteurs utilisent des embeddings de dimension 400 pour les mots et les documents sachant que leur corpus est plus petit.

Pour générer la base d'apprentissage, on extrait d'abord les N documents les plus cités, que nous allons nommer *top-N*. À partir de ces N documents, qui correspondent aux B de notre liste de paires de documents ci-haut, nous récupérons tous les documents A les citant. La table 1 donne le nombre d'exemples avec citations (*exemples positifs)* pour N=16000 pour chacun des ensembles de train, validation et test.

Nombre de documents cités	Entraînement	Validation	Test
top-16 000	1 113 973	137 521	140 552

Table 1: Nombre d'exemples positifs par ensemble

Néanmoins, il faut aussi montrer des exemples négatifs de citation au modèle, c'est-à-dire des paires de documents (A,B) qui ne possèdent pas la relation de citation. Ainsi pour chaque document cité, on peut échantillonner un document citant qui lui est antérieur chronologiquement. Par contre, pour un document cité, on doit déterminer le ratio entre le nombre d'exemples positifs et négatifs. La stratégie 1:1 consiste à avoir un ratio un positif pour un négatif. La stratégie 1% consiste à échantillonner 1% du nombre de documents citants qui respectent l'ordre chronologique et qui n'ont pas de relation de citation.

Stratégie	Entraînement	Validation
1%	94 627 331	11 822 694

Table 3: Nombre d'exemples par ensemble pour le top-16000

Par contre, pour l'évaluation sur le test set, nous utilisons la distribution naturelle des données. En raison du nombre important de paires de documents respectant la contrainte chronologique, nous évaluons le modèle avec un sous-ensemble des 16 000 documents cités utilisés pendant l'entraînement. Nous échantillons sans remplacement 1000 documents cités avec une probabilité proportionnelle à son nombre de citations. Ceci donne 98 918 000 exemples dans l'ensemble de test dont 140 552 positifs (0.14% d'exemples positifs).

Modèle

Nous proposons une architecture à deux branches, grandement inspiré de [2], qui compare les représentations latentes des deux documents. Cette architecture possède trois sections: l'encodeur des documents cité et citant, la partie de projection dans un espace métrique et une section qui calcule une distance paramétrée par la date de citation. La partie encodeur est probablement l'une des plus importantes, car elle doit capturer l'information importante dans des documents de plusieurs pages. Pour notre baseline, nous utilisons l'algorithme de embedding de document: doc2vec [3]. Bien que cet algorithme soit contesté dans la littérature, son utilisation est simple à l'aide de la bibliothèque Gensim. Une fois que nous avons des vecteurs qui représentent les documents, nous concaténons les méta-informations à ces vecteurs. Ces vecteurs sont ensuite fusionnés avec les opérations de l'article [2]. Enfin, puisque la relation de citation n'est pas invariante dans le temps, nous paramétrons le calcul du score final avec la date à laquelle la citation a été faite. En effet, un document récent peut rendre des citations du passé obsolète. Ce nouveau document, publié par une cour importante, masquera les documents qui étaient cités auparavant.

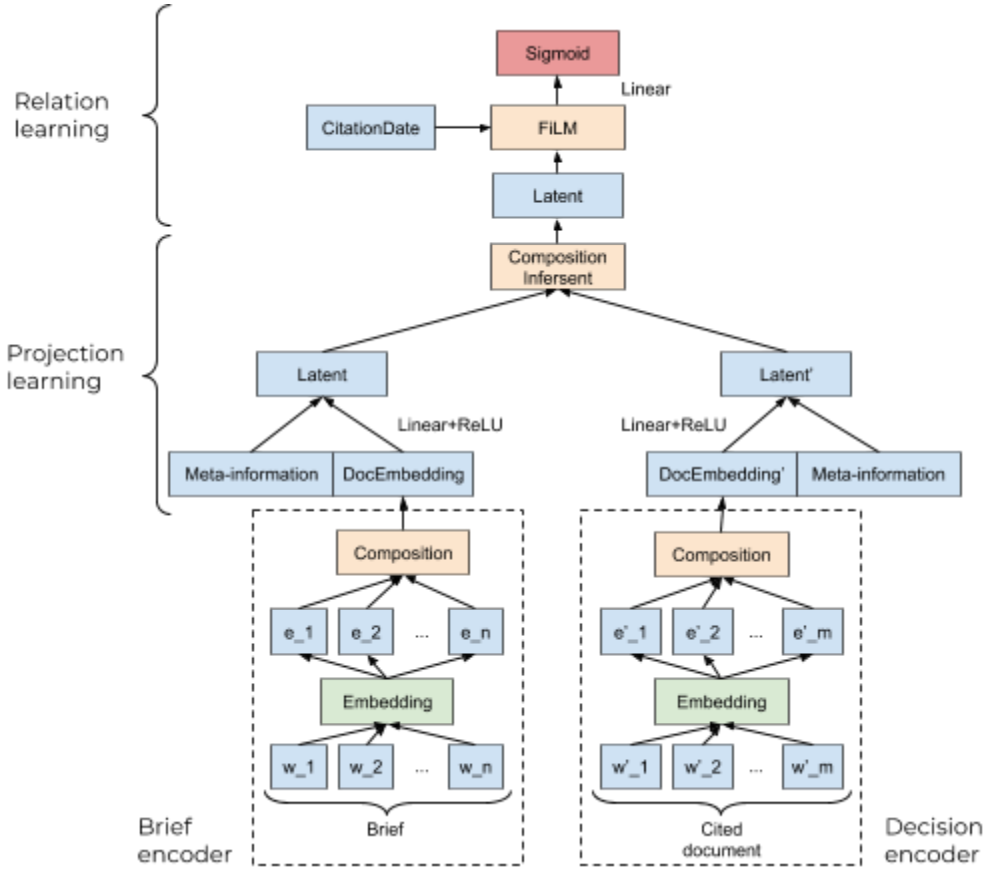

Figure 1: Modèle à deux branches pour l'apprentissage de citations

Base d'apprentissage non balancée

Puisque la probabilité que deux documents choisis aléatoirement possèdent une relation de citation est très faible, la procédure d'entraînement doit dévier du cadre *indépendant et identiquement distribué* pour prendre en compte davantage d'exemples positifs. Pour ce faire, deux techniques sont couramment utilisées: la repondération des exemples d'entraînement et l'échantillonnage biaisé (*negative sampling*). La procédure adoptée est d'utiliser tous les exemples positifs et de faire un échantillonnage des exemples négatifs parmi 1% des décisions éligibles. La fonction de coût adoptée est le maximum de vraisemblance (*negative log-likelihood*).

Résultats empiriques

Precision	Recall	F0.5	Seuil
0.4539	0.0750	0.4194	0.993935

Table 1: Évaluation du modèle sans prendre en compte un nombre maximal de requêtes. Le seuil choisi maximise le F0.5 score sur un ensemble de validation.

@k	Precision	Recall	True pos.	False pos.
1	0.4756	0.0690	615	678
5	0.4545	0.0750	669	803
10	0.4539	0.0750	669	805
30	0.4539	0.0750	669	805

Table 2: Évaluation du modèle en prenant en compte un nombre maximal de requêtes @k avec le seuil optimal.

Seuil @5	Precision	Recall	True pos.	False pos.
0.96	0.2631	0.1782	1589	4450
0.97	0.2908	0.1580	1409	3436
0.98	0.3274	0.1320	1177	2418
0.99	0.4070	0.0974	869	1266

Table 3: Évaluation du modèle en faisant varier le seuil entre 0.96 et 0.99

Nous ne donnons pas les résultats pour le NDCG en raison de la difficulté à interpréter le score. La prise en compte du seuil dans la mesure NDCG devrait être vérifiée.

Discussion

Les résultats de la table 2 montrent le potentiel de l'architecture proposée pour le problème de découverte des citations. En effet, même avec aussi peu que 0.14% d'exemples positifs et une méthode de embeddings génériques, le modèle est capable d'atteindre une précision proche de 50% en étant très conservateur (seuil très élevé). Par contre, ces performances fluctuent beaucoup avec le seuil choisi comme le montre la table 3. Même en relâchant le seuil à 0.96, le nombre de recommandations par document reste très faible (1-2 recommandation par requête).

Au niveau du protocole expérimental, les métriques de performance doivent encore être raffinées pour prendre en compte l'expérience utilisateur qui sera évaluée en aval du projet. Il est plus facile d'orienter les choix lorsque nous savons que les métriques sont les bonnes. Ensuite, une étude *d'ablation* devrait être faite pour évaluer l'apport de chacune des composantes de l'architecture. Par exemple, on devrait pouvoir évaluer l'apport de nouvelles méta-données ou l'apport du module FILM dans les résultats.

Ensuite, une approche à deux niveaux, comme utilisée dans [4], devrait améliorer les résultats. Le premier niveau commence par filtrer les documents qui n'ont aucun lien afin de simplifier l'analyse du modèle du second niveau. Le filtre peut être implémenté à l'aide des embeddings de documents. De plus, avec un algorithme efficace du plus proche voisin comme FAISS [5], ce filtre pourra accélérer grandement les recherches des utilisateurs.

Plusieurs autres améliorations du modèle sont possibles. Nous pouvons tester différentes variantes pour l'encodeur des documents. La méthode de *paragraph vector* utilisée est maintenant obsolète et des alternatives plus performantes existent, mais aussi plus complexes à mettre en oeuvre. Une alternative tout aussi simple à mettre en oeuvre est le *Universal Sentence Encoder* [6]. Ensuite, plusieurs méthodes d'agrégation des word embeddings ont été proposées récemment. Ces méthodes incluent des moyennes pondérées [7], des architectures

récurrentes sans apprentissage (poids aléatoires) [8], des réseaux récurrents hiérarchiques [9][10], des réseaux convolutionnels [11] ainsi que la nouvelle famille de modèles transformer [12] [13] [14]. Plusieurs de ces méthodes sont implémentées dans la librairie Spacy [15] et FAIRSEQ[1]. Ces méthodes spécifient aussi le niveau d'encodage des symboles des documents. Certaines approches utilisent les caractères, d'autres prennent des word embeddings pré-entrainées (word2vec, Glove) et enfin, certains utilisent des tokens qui sont entre le niveau des caractères et des mots (WordPiece). Ces choix peuvent aussi influencer les résultats. Enfin, il existe aussi des algorithmes de similarité de documents basés sur une notion de transport optimal [16].

[1] https://github.com/pytorch/fairseq

Références

[1] Flach, Peter, and Meelis Kull. "Precision-recall-gain curves: PR analysis done right." Advances in Neural Information Processing Systems. (2015).
[2] Conneau, Alexis, et al. "Supervised learning of universal sentence representations from natural language inference data." arXiv:1705.02364 (2017).
[3] Dai, Andrew M., Christopher Olah, and Quoc V. Le. "Document embedding with paragraph vectors." arXiv:1507.07998 (2015).
[4] Bhagavatula, Chandra, et al. "Content-based citation recommendation." arXiv preprint arXiv:1802.08301 (2018).
[5] Johnson, Jeff, Matthijs Douze, and Hervé Jégou. "Billion-scale similarity search with GPUs." arXiv preprint arXiv:1702.08734 (2017).
[6] Cer, Daniel, et al. "Universal sentence encoder." arXiv:1803.11175, https://tfhub.dev/google/universal-sentence-encoder/2 (2018).
[7] Arora, Sanjeev, Yingyu Liang, and Tengyu Ma. "A simple but tough-to-beat baseline for sentence embeddings." (2016).
[8] Wieting, John, and Douwe Kiela. "No Training Required: Exploring Random Encoders for Sentence Classification." arXiv:1901.10444 (2019).
[9] Peters, Matthew E., et al. "Deep contextualized word representations." arXiv preprint arXiv:1802.05365 (2018).
[10] Jiang, Jyun-Yu, et al. "Semantic Text Matching for Long-Form Documents." (2019).
[11] Dauphin, Yann N., et al. "Language modeling with gated convolutional networks." Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017.
[12] Vaswani, Ashish, et al. "Attention is all you need." Advances in Neural Information Processing Systems. (2017).
[13] Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018).
[14] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[15] Honnibal, Matthew and Montani, Ines. "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing", to appear, https://spacy.io/ (2017).
[16] Kusner, Matt, et al. "From word embeddings to document distances." International Conference on Machine Learning. 2015.
https://markroxor.github.io/gensim/static/notebooks/WMD_tutorial.html

Remerciement

Nous remercions Benjamin Cerat, Daniel Shane et Marc-André Morissette qui ont implémenté le protocole expérimental et produit les résultats. Ces travaux ont été financés par le PARI-CNRC.