MT-DNN-ensemble (Liu et al., 2019)

Qu'est ce que GLUE ?

GLUE (General Language Understanding Evaluation) est une plate-forme de référence et d'analyse multi-tâches pour la compréhension du langage naturel.

Il faut obtenir ainsi le meilleur score possible dans les 11 tâches d'évaluation (voir lien ci-dessous) imposé par la plate-forme.

- https://gluebenchmark.com/leaderboard
- https://gluebenchmark.com/tasks

Classement GLUE de MT-DNN-ensemble (Liu et al., 2019) : 20

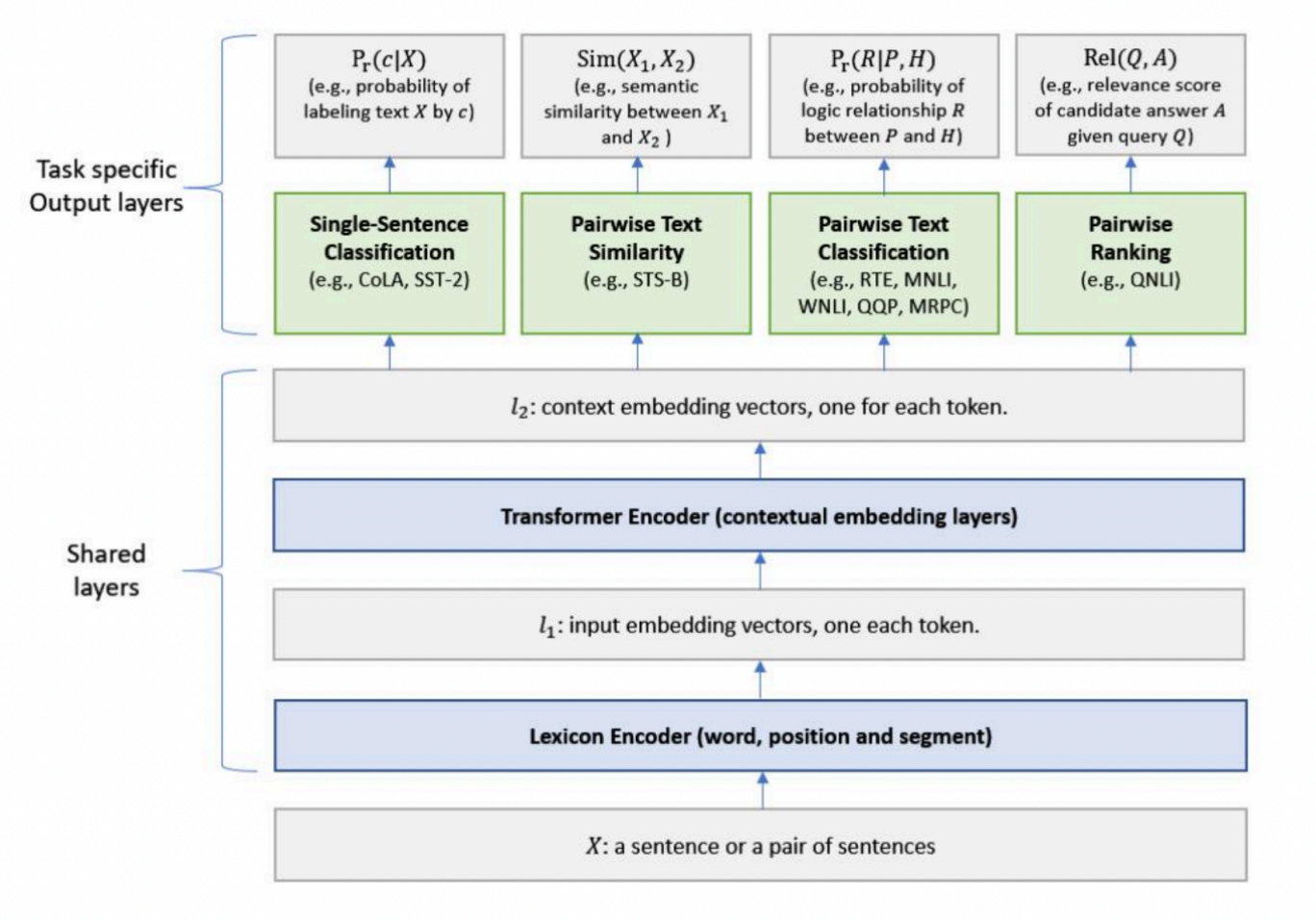
Score obtenus : 87.6

L'implémentation de MT-DNN est basée sur les implémentations PyTorch de MT-DNN3 et BERT4 .De plus, MT-DNN utilise Adamax comme optimiseur.

Architecture du modèle MT-DNN

Les chercheurs de Microsoft ont publié les détails techniques d'un système d'IA qui combine l'apprentissage multi-tâches ainsi que la pré-formation de modèles linguistiques . Le nouveau réseau de neurones profonds multitâches (MT-DNN) est un modèle de traitement du langage naturel (NLP) qui surpasse Google BERT dans neuf des onze tâches NLP de référence.

MT-DNN s'appuie sur un modèle proposé par Microsoft en 2015 et intègre l'architecture réseau de BERT, un modèle de langage de transformateur bidirectionnel pré-entraîné proposé par Google l'année dernière.



Comme le montre la figure ci-dessus, les couches de bas niveau du réseau (c'est-à-dire les couches d'encodage de texte) sont partagées entre toutes les tâches, tandis que les couches supérieures sont spécifiques aux tâches, combinant différents types de tâches NLU (Natural language understanding). Comme le modèle BERT, MT-DNN est formé en deux phases : pré-formation et mise au point. Mais contrairement à BERT, MT-DNN ajoute l'apprentissage multitâche (MTL) dans les phases de réglage fin avec plusieurs couches spécifiques aux tâches dans son architecture de modèle.

L'entrée X, qui est une séquence de mots (soit une phrase soit un ensemble de phrases regroupées) est d'abord représentée comme une séquence de vecteurs d'intégration, un pour chaque mot, dans *l1*. Ensuite, le codeur transformateur capture les informations contextuelles pour chaque mot et génère les vecteurs d'intégration contextuelle partagés dans l2. C'est la représentation sémantique partagée qui est entraînée par nos objectifs multitâches.

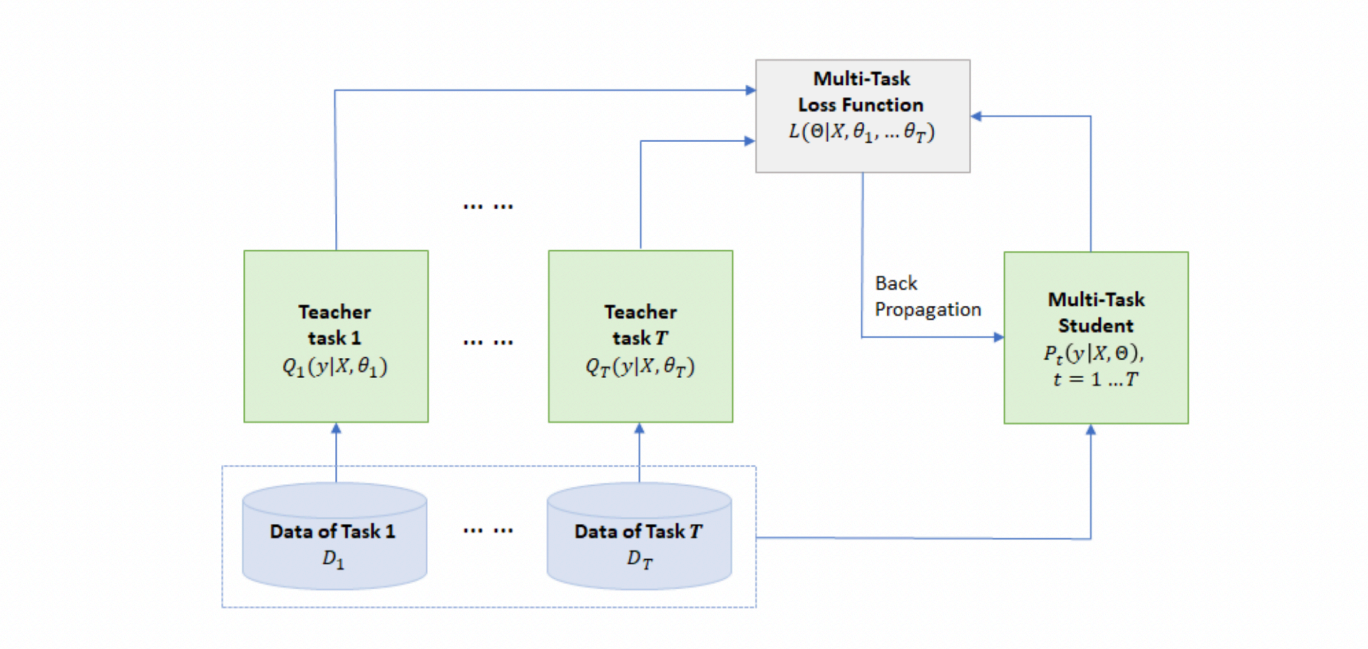
Enfin, pour chaque tâche, des couches supplémentaires spécifiques à la tâche génèrent des représentations spécifiques à la tâche, suivies des opérations nécessaires à la classification, à la notation de similarité ou au classement de pertinence.

Lexicon Encoder : L'entrée X = {x1, ..., xm} est une séquence de jetons de longueur m. Si X est emballé par un ensemble de phrases (X1, X2), nous séparons ces phrases avec des jetons spéciaux [SEP]. L'encodeur de lexique mappe X en une séquence de vecteurs d'intégration d'entrée, un pour chaque jeton, construit en additionnant les intégrations de mot, de segment et de position correspondantes.

Transformer Encoder : Le modèle utilise un "multilayer bidirectional Transformer encoder" pour mapper les vecteurs de représentation d'entrée (l1) en une séquence de vecteurs d'intégration contextuels C. Il s'agit de la représentation partagée entre différentes tâches.

Couches de sortie spécifiques à la tâche : Le modèle incorpore des tâches arbitraires en langage naturel, chacune avec ses couches de sortie spécifiques à la tâche. Par exemple, nous implémentons les couches de sortie comme un décodeur neuronal pour la génération de texte, un classement neuronal pour le classement par pertinence, une régression logistique pour la classification du texte, etc.

Processus de distillation des connaissances pour l'apprentissage multi-tâches



Un ensemble de tâches où il y a des données de formation étiquetées spécifiques à la tâche est sélectionné. Ensuite, pour chaque tâche, un ensemble de réseaux de neurones différents (enseignant) est formé. L'enseignant est utilisé pour générer pour chaque échantillon de formation spécifique à une tâche un ensemble de cibles souples. Compte

tenu des cibles souples des ensembles de données de formation sur plusieurs tâches, un seul MT-DNN (étudiant) est formé à l'aide de l'apprentissage multitâche et de la rétropropagation comme décrit dans l'algorithme, sauf que si la tâche *t* a un enseignant, la perte spécifique à la tâche dans la ligne 3 (voir algorithme ci-dessous) est la moyenne de deux fonctions objectives, l'une pour les cibles correctes et l'autre pour les cibles souples assignées par l'enseignant.

Une fois que MT-DNN est formé via MTL, il peut être affiné (ou adapté) à l'aide de données de formation étiquetées spécifiques à la tâche pour effectuer une prédiction sur n'importe quelle tâche individuelle, qui peut être une tâche utilisée dans l'étape MTL ou une nouvelle tâche qui est liés à ceux utilisés dans MTL. Liu et al. (2019) ont montré que les couches partagées de MT-DNN produisent des représentations textuelles plus universelles que celles de BERT. En conséquence, MT-DNN permet un réglage fin ou une adaptation avec beaucoup moins d'étiquettes spécifiques aux tâches.

Lorsque les cibles correctes sont connues, les performances du modèle peuvent être considérablement améliorées en entraînant le modèle distillé sur une combinaison de cibles souples et dures. Nous le faisons en définissant une fonction de perte pour chaque tâche qui prend une moyenne pondérée entre la perte d'entropie croisée avec les cibles correctes.

Ce modèle a pour but de montrer la distillation de connaissances étendu à MTL pour former un MT-DNN pour la compréhension du langage naturel. Nous avons montré que la distillation fonctionne très bien pour transférer les connaissances d'un ensemble de modèles (enseignants) vers un seul MT-DNN distillé (étudiant).

Algorithme

Algorithm 1: Training a MT-DNN model.

Initialize model parameters Θ randomly.
Initialize the shared layers (i.e., the lexicon encoder and the transformer encoder) using a pre-trained BERT model.
Set the max number of epoch: $epoch_{max}$.
//Prepare the data for T tasks.
for t *in* $1, 2, ..., T$ **do**
  Pack the dataset t into mini-batch: D_t.
end
for $epoch$ *in* $1, 2, ..., epoch_{max}$ **do**
  1. Merge all the datasets:
    $D = D_1 \cup D_2 ... \cup D_T$
  2. Shuffle D
  for b_t *in* D **do**
    //b_t is a mini-batch of task t.
    3. Compute task-specific loss : $L_t(\Theta)$
    4. Compute gradient: $\nabla(\Theta)$
    5. Update model: $\Theta = \Theta - \epsilon\nabla(\Theta)$
  end
end

Résultat

MT-DNNKD : Il s'agit du modèle MT-DNN entraîné en utilisant la distillation des connaissances. MT-DNNKD utilise la même architecture de modèle que celle de MT-DNN. Mais il est entraîné avec l'aide de quatre ensembles spécifiques à la tâche (enseignants). Le MT-DNNKD est optimisé pour les objectifs multitâches qui sont basés sur les cibles correctes dures, ainsi que sur les cibles douces produites par les enseignants si elles sont disponibles.

Model	CoLA 8.5k	SST-2 67k	MRPC 3.7k	STS-B 7k	QQP 364k	MNLI-m/mm 393k	QNLI 108k	RTE 2.5k	WNLI 634	AX	**Score**
BiLSTM+ELMo+Attn [1]	36.0	90.4	84.9/77.9	75.1/73.3	64.8/84.7	76.4/76.1	79.8	56.8	65.1	26.5	70.0
Singletask Pretrain Transformer [2]	45.4	91.3	82.3/75.7	82.0/80.0	70.3/88.5	82.1/81.4	87.4	56.0	53.4	29.8	72.8
GPT on STILTs [3]	47.2	93.1	87.7/83.7	85.3/84.8	70.1/88.1	80.8/80.6	-	69.1	65.1	29.4	76.9
$BERT_{LARGE}$ [4]	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7/85.9	92.7	70.1	65.1	39.6	80.5
MT-DNN[5]	61.5	95.6	90.0/86.7	88.3/87.7	72.4/89.6	86.7/86.0	-	75.5	65.1	40.3	82.2
Snorkel MeTaL [6]	63.8	**96.2**	91.5/88.5	**90.1/89.7**	73.1/89.9	87.6/87.2	93.9	80.9	65.1	39.9	83.2
ALICE *	63.5	95.2	**91.8/89.0**	89.8/88.8	**74.0/90.4**	**87.9/87.4**	95.7	80.9	65.1	40.7	83.3
$MT\text{-}DNN_{KD}$	**65.4**	95.6	91.1/88.2	89.6/89.0	72.7/89.6	87.5/86.7	**96.0**	**85.1**	65.1	**42.8**	**83.7**
Human Performance	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0/92.8	91.2	93.6	95.9	-	87.1

Le tableau montre que MT-DNNKD surpasse significativement MT-DNN non seulement dans le score global mais aussi sur 7 des 9 tâches GLUE, y compris les tâches sans enseignant. Puisque MT-DNNKD et MT-DNN utilisent la même architecture de réseau, et sont entraînés avec la même initialisation et sur les mêmes jeux de données, l'amélioration de MT-DNNKD est uniquement attribuée à l'utilisation de la distillation des connaissances dans MTL.

Nous montrons que le MT-DNN distillé conserve presque toutes les améliorations obtenues par les modèles d'ensemble, tout en gardant la même taille de modèle comme le modèle MT-DNN de base.