Projet PSTALN 2023-2024

Carlos Ramisch, Alexis Nasr

30/11/2023

Le projet de PSTALN consiste à *développer* et *évaluer* un système de *prédiction de structures linguistiques* à partir de textes. Ces structures linguistiques peuvent être très diverses, allant des parties du discours de chaque mot (nom, verbe, adjectif, etc.) aux expressions idiomatiques contenues dans une phrase (*Les étudiant.e.s de PSTALN ont du pain sur la planche !*) Par exemple, à partir de la phrase ci-dessous, votre système devra prédire les informations du tableau en dessous :

Entrée ⇒	Les	étudiants	et	étudiantes	de	PSTALN	ont	de	le	pain	sur	la	planche	!
POS ⇒	DET	NOUN	CCONJ	NOUN	ADP	PROPN	VERB	ADP	DET	NOUN	ADP	DET	NOUN	PUNCT
Lemmes ⇒	*le*	*étudiant*	*et*	*étudiante*	*de*	*PSTALN*	*avoir*	*de*	*le*	*pain*	*sur*	*le*	*planche*	*!*
Morphologie ⇒	*Definite=Def Number=Plur PronType=Art*	*Gender=Masc Number=Plur*	–	*Gender=Fem Number=Plur*	–	–	*Mood=Ind Number=Plur Person=3 Tense=Pres VerbForm=Fin*	–	*Definite=Def Gender=Masc Number=Sing PronType=Art*	*Gender=Masc Number=Sing*	–	*Definite=Def Gender=Fem Number=Sing PronType=Art*	*Gender=Fem Number=Sing*	–
Expressions ⇒	_	_	_	_	_	_	EXP	EXP	EXP	EXP	EXP	EXP	EXP	_

Le système sera fondé sur un modèle *d'apprentissage* à base de réseaux de neurones, développé à l'aide de la bibliothèque `pytorch`. Vous travaillerez en groupes de **3 personnes**. Vous serez évalué.e par un *rapport écrit* et une *soutenances orale*. Ci-dessous, nous décrivons un certain nombre de choix possibles chacun des 6 aspects du système (tâches, combinaison de tâches, architectures, langues, encodage des entrées, et encodage des sorties). Chaque groupe composera son sujet "à la carte" en choisissant, pour chacun de ces aspects, une ou plusieurs valeurs.

1 Choix à effectuer

1.1 Tâches de prédiction de structures

Parties du discours : dans cette tâche, chaque mot reçoit une étiquette de *partie du discours* (POS, de l'anglais *part of speech*). Ces étiquettes indiquent le rôle grammatical du mot dans la phrase, par exemple : le mot *été* est un nom dans *l'été dernier*. Les défis ici sont : (1) certains mots sont ambigus (p.ex. *été*) est un verbe dans *j'ai été informée*) ; (2) le contexte joue un rôle important pour désambiguïser ; et (3) le système doit être capable de prédire des POS pour des mots non observés à l'entraînement (p.ex. noms propres).

Lemmatisation Étant donné les différentes formes que peut prendre un mot variable : *manger*, *manges*, *mangeront*, *mangeâmes* ..., le **lemme** est une de ces formes choisie arbitrairement. Pour les verbes, il s'agit de l'infinitif (*manger*), dans le cas d'un nom, d'un déterminant ou d'un adjectif, il s'agit du mot au masculin singulier, etc. Le défi ici réside dans le grand nombre de classes à prédire (tous les lemmes possibles). Il faudra trouver des manières astucieuses d'encoder les sorties à prédire (cf. Section 1.6).

Traits morphologiques Les traits morphologiques sont complémentaires aux POS, par exemple, la flexion des noms en nombre (singulier, pluriel), le temps des verbes (passé, présent, etc.), etc. La difficulté ici est surtout technique, car le nombre de traits à prédire est variable : certains mots n'ont aucun trait (p.ex. prépositions) alors que d'autres en ont un grand nombre (p.ex. verbes). Faut-il entraîner un modèle par POS ? Par type de trait ? Un seul modèle qui prédit tous les traits ? Ou prédire un nombre variable de traits ?

Expressions idiomatiques Les expressions idiomatiques (MWE, de l'anglais *multiword expression* sont une séquence d'au moins 2 mots dont le sens n'est pas littéralement celui des mots qui la composent (p.ex. *avoir du pain sur la planche* veut dire *avoir beaucoup de travail*). Cette tâche consiste à repérer les MWE dans les phrases, sachant qu'une MWE n'est pas toujours composée de mots adjacents (*j'ai encore du pain sur la planche*. De plus, la plupart des mots ne font pas partie d'une MWE, donc il faut les prédire avec parcimonie.

Dans les données fournies pour le projet, d'autres tâches sont incluses. On pourrait imaginer que le système doit aussi découper le texte brut en phrases, et les phrases en mots (tokens). De plus, les données contiennent des arbres syntaxiques (colonnes HEAD et DEPREL) voire des graphes sémantiques (colonne DEPS). Ces tâches requièrent des modèles plus complexes, notamment pour l'évaluation. Elles ne sont donc pas proposées compte tenu des délais du projet.

1.2 Stratégies de combinaison des tâches

Modèles indépendants Classiquement, chacune de ces tâches peut être réalisée par des modèles dédiés et indépendants. Ainsi, vous pouvez développer un classifieur avec des choix (p.ex. d'architecture) spécifiques à chaque tâche choisie. Chaque modèle prend en entrée uniquement les mots de la phrase tokénisée (FORM).

Pipeline / cascade Le découpage modulaire en tâches omet l'inter-dépendance entre les niveaux. Par exemple, le lemme peut souvent être déduit de la POS et des traits morphologiques, les MWE seront mieux prédites à partir des lemmes, etc. La combinaison en pipeline consiste à connecter des modèles indépendants : la prédiction d'un modèle est utilisée en entrée du prochain. La question ici est : dans quel ordre enchaîner les tâches ?

Modèle joint Plus récemment, les avancées du deep learning ont permis de faire des modèles joints, où les prédictions sont effectuées en parallèle. Le modèle joint partage un grand nombre de paramètres avant de se spécialiser, et est entraîné avec une fonction de perte portant sur toutes les tâches en même temps.

Selon les tâches choisies, vous pouvez combiner certaines en pipeline, d'autres avec un modèle joint : à vous de tester vos intuitions linguistiques sur le bénéfice mutuel entre les tâches.

1.3 Architectures des classifieurs

Chacune des tâches ont des caractéristiques différentes. Pour certaines, un contexte assez localisé (p.ex. 3 mots avant, 3 mots après) permet d'obtenir des bonnes prédictions, alors que d'autres requièrent plus de contexte. Vous pourrez tester une ou plusieurs architectures de classifieur selon vos connaissances de la tâche.

MLP / Feedforward Un modèle MLP prend en entrée une fenêtre glissant de k mots avant/après le mot cible, et prédit une ou plusieurs étiquettes pour le mot du milieu. Il peut avoir des couches cachées.

Convolution Une alternative aux couches denses sont les couches de convolution, souvent plus rapides et assez efficaces pour extraire des traits qui font abstraction de la position absolue du mot dans la phrase. De plus, vous pouvez aussi appliquer la convolution au niveau des caractères, si cela vous paraît pertinent.

Réseau récurrent Idéal pour les tâches d'étiquetage de séquences demandant une contextualisation variable, les réseaux récurrents (GRU, LSTM) sont naturellement des bons candidats certaines tâches décrites ci-dessus. Attention à bien traiter la longueur variable des séquences (p.ex. avec du padding) et à bien construire les batchs (p.ex. grouper des phrases de longueur similaire)

Transformer Rendues très populaires grâce aux modèles de langage, les couches d'auto-attention des transformers sont aussi potentiellement pertinentes ici. Si vous partez sur un transformer, n'oubliez pas de rajouter des embeddings de position, car le modèle de base ignore l'ordre des mots. Vous pourrez aussi utiliser des transformers pré-entraînés, p.ex. ceux disponibles dans la bibliothèque `transformers` de Huggingface. Dans ce cas, préférez des versions légères de ces modèles (p.ex. DistilBERT au lieu de BERT-large), sinon vous risquez de passer 99% de votre temps à attendre !

Vous pouvez, bien entendu, combiner les architectures et/ou faire varier leurs hyper-paramètres. Mais attention à "l'effet lasagne" : ne pas rajouter des couches arbitrairement sans justification.

1.4 Langue des données

Les données d'entraînement et d'évaluation sont disponibles en plusieurs langues. Vous pouvez vous concentrer sur une seule langue, ou effectuer des expériences sur plusieurs langues en même temps. De plus, vous pouvez faires des expériences multilingues : entraîner le système sur plusieurs langues, voire l'évaluer sur une langue différente de celle sur laquelle il a été entraîné.

Voici quelques questions intéressantes : (a) Peut-on faire un système unique capable de prédire des structures linguistiques pour plusieurs langues ? (b) Peut-on bénéficier de données d'autres langues (proches) pour

améliorer le système d'une langue donnée ? (c) Peut-on apprendre des embeddings qui marchent pour plusieurs langues ? (d) Est-ce que certaines paires (tâche, langue) sont plus difficiles que d'autres ?

1.5 Encodage des entrées

Votre système prend en entrée du texte découpé au préalable en phrases et en mots (texte tokénisé). Il faudra ensuite convertir chaque mot en une représentation compatible avec les classifieurs, donc en un vecteur (*embedding*) de nombres réels. Pour cela, vous pouvez utiliser des embeddings initialisés aléatoirement, sous la forme d'une couche `Embedding` directement dans `pytorch`. Alternativement, vous pouvez utiliser des embeddings pré-entraînés, et là encore vous avez plusieurs options : embeddings **statiques** tels que ceux générés par Word2vec ou Fasttext, embeddings **contextuels** tels que ceux générés par BERT, ou embeddings **contextuels multilingues** tels ceux de mBERT. Vous pouvez comparer ces encodages, notamment sur leur capacité à générer des embeddings pour les mots ambigus (p.ex. *été*) et pour les mots hors vocabulaire.

1.6 Encodage des sorties

La tâche de prédiction de POS a un encodage trivial : chaque mot reçoit une étiquette unique à prédire parmi un ensemble réduit d'étiquettes possibles. Cependant, c'est plus compliqué pour les autres tâches.

Lemmatisation En théorie, vous pouvez adopter la même stratégie et considérer tous les lemmes à prédire comme des étiquettes. Cependant, cela n'est pas très efficace. L'alternative consiste à prédire une "règle" de remplacement de suffixe qui permet d'aller de la forme au lemme. Par exemple, la forme *mangeront* peut être transformée en *manger* en remplaçant le suffixe *ont* par rien, donc sa règle s'écrit *ont@ε*. Pour la forme *mangeais*, la règle est *ais@r*. Certaines exceptions auront des règles très spécifiques, p.ex. *eu@avoir*, que le système devra tout simplement mémoriser. Une autre stratégie consiste à voir cette tâche comme une transduction caractère à caractère : pour chaque caractère de la forme, il faudra prédire le caractère du lemme correspondant. Dans ce cas, une architecture récurrente opérant sur les caractères est recommandée.

Traits morphologiques La première stratégie consiste à considérer la concaténation de tous les traits comme une seule "super-étiquette" à prédire, comme les POS. Alternativement, vous pouvez faire des classifieurs séparés par POS, avec les traits pertinents pour chaque POS. Aussi, il est possible de séparer les traits selon les clés, et prédire une valeur spéciale `None` quand le trait n'est pas présent. Vous pouvez aussi voir la tâche comme un problème de génération auto-regressive, comme pour les modèles de langage.

Expressions idiomatiques Cette tâche peut être abordée avec un schéma adapté du Begin-Inside-Outside, par exemple, en ajoutant une étiquette Gap pour les "trous" dans les expressions discontinues. Les expressions ont aussi des catégories, que vous pouvez utiliser pour compléter les étiquettes. Attention, si vous utilisez une prédiction gloutonne, vous risquez de générer des séquences invalides (p.ex. Outide-Inside). Il vaut mieux privilégier un décodage par Viterbi, par exemple.

2 Données

Vous utiliserez les données des projets Universal Dependencies (UD) [1] et PARSEME [2] pour entraîner vos modèles. Si vous avez choisi la tâche de prédiction d'expressions idiomatiques parmi vos tâches, vous devez prendre les données PARSEME. Si vous ne l'avez pas choisie, préférez les données UD.

Ces données sont fournies au format tabulaire. Les fichiers texte UTF-8 contiennent un mot par ligne, avec des lignes blanches entre les phrases. Les meta-informations (commençant par un `#`) peuvent être ignorées. Chaque mot sur une ligne contient 10 ou 11 colonnes séparées par tabulation. L'entrée de votre système est la deuxième colonne : FORM. Le système doit prédire les colonnes 3 (LEMMA), 4 (UPOS), 6 (FEATS), ou 11 (PARSEME :MWE). Vous trouverez la description des formats CoNLL-U [3] et CUPT [4] sur les sites des projets. Le format CUPT est une extension de CoNLL-U, identique sur les 10 premières colonnes, et avec une 11ème colonne pour les expressions idiomatiques.

1. `http://hdl.handle.net/11234/1-5287`
2. `http://hdl.handle.net/11234/1-3367`
3. `https://universaldependencies.org/format`
4. `https://multiword.sourceforge.net/cupt-format/`

3 Évaluation du système

Les tâches de prédiction de POS, lemmes et traits morphologiques s'évaluent avec la métrique *accuracy* (exactitude, taux de prédictions correctes). Cependant, cette métrique étant sensible au déséquilibre des classes, vous pouvez aussi moyenner le F-score par étiquette. Vous trouverez des scripts vous permettant de calculer ces scores sur la page de la campagne d'évaluation UD (pour les 3 premières tâches) : `http://universaldependencies.org/conll18/evaluation.html`. Utilisez l'option `-v` et regardez uniquement les lignes concernant les tâches `UPOS`, `UFeats` et `Lemmas`.

La tâche de prédiction des expressions idiomatiques a des métriques spécifiques de F-score (par expression et par token). Vous trouverez un script d'évaluation `evaluate.py` dans le dépôt `https://gitlab.com/parseme/sharedtask-data/-/tree/master/1.2/bin` (attention : télécharger tout le dossier, pas uniquement le script).

4 Travail à effectuer

Parmi les multiples questions que l'on peut se poser en combinant les aspects ci-dessus, le groupe devra choisir **une question exploratoire** à développer en profondeur, et faire des expériences pour essayer d'y répondre. La réalisation de ces expériences pour répondre à la question fait partie des compétences évaluées. Les étapes à suivre sont :

1. **Choix du sujet :** Découverte des problèmes proposés, téléchargement et exploration des données, lecture de la documentation, discussion avec les enseignants pour le choix du sujet et de la question exploratoire, organisation du groupe, création d'un dépôt git, et partage des tâches.
2. **Baseline :** Développement d'un système baseline simple : p.ex. prédiction par classe majoritaire. Mise en place du protocole allant de l'entraînement jusqu'à l'évaluation, obtention des scores pour la baseline.
3. **Modèle proposé :** Développement d'un modèle neuronal à l'aide de `pytorch` selon le sujet choisi. Premiers tests sur des petits jeux de données. Choix de l'architecture et des hyper-paramètres.
4. **Évaluation et analyse des résultats** : Évaluation du modèle proposé sur les données de test, comparaison avec les résultats obtenus par la baseline et/ou par plusieurs modèles, points forts et faibles, perspectives (si on avait eu plus de temps on aurait...)

Ces étapes correspondent plus ou moins aux sections de votre rapport. Le rapport couvrira votre compréhension du sujet et de la question exploratoire choisie, une description du système développé, la description des expériences, l'analyse des résultats du modèle, (en particulier par rapport à la baseline), et vos conclusions sur la question exploratoire. Il faut inclure le lien vers le code du modèle, mais le code-source ne sera pas évalué directement. Le rapport ne doit pas dépasser 15 pages, il n'y a pas de minimum.

Critères d'évaluation Selon le sujet choisi, il peut être plus ou moins facile d'obtenir des bons scores lors de l'évaluation. Nous nous attendons à ce que votre modèle obtienne des résultats supérieurs à une baseline simple, mais obtenir les meilleurs scores ne doit pas être votre priorité. En particulier, l'évaluation du travail n'est pas corrélée avec les performances de votre système.

Notamment, vous ne disposez pas forcément de milliers d'heures GPU pour entraîner pendant des centaines d'époques, charger des gigas en mémoire, ou optimiser des milliards de paramètres. Vous pourrez faire des simplifications qui vous permettront tout de même d'apporter des éléments de réponse à votre question.

Votre rapport et soutenance seront évalués selon les critères suivants :
— Clarté et pédagogie, utilisation d'exemples, tableaux, graphiques, qualité des slides
— Méthodologie expérimentale, choix justifiés (p.ex. architecture du réseau de neurones)
— Quantité de travail, diversité d'approches, profondeur des analyses
— Structure du rapport/présentation, travail en équipe, organisation

Dates importantes
— 11/12/2023 - Constitution des groupes (3 personnes) et choix du sujet
— 10/01/2024 - Rendu du rapport écrit
— 15/01/2024 - Soutenance orale (≈15min de présentation + 5min de questions)