Analyse syntaxique multilingue

Traitement du Langage Naturel et Linguistique
Projet

5 novembre 2020

1 Objectif

L'objectif de ce projet est de développer un analyseur multilingue délexicalisé. Un tel analyseur prend en entrée une séquence de parties de discours correspondant à une phrase et produit un arbre de dépendances pour cette phrase. L'analyseur n'a donc pas accès à la représentation orthographique des mots, c'est en ce sens qu'il est délexicalisé. Son autre particularité est qu'il peut prendre en entrée des séquences de parties de discours de langues différentes, c'est en ce sens qu'il est multilingue. L'analyseur ne connait pas exactement la langue à laquelle appartient une phrase particulière, mais il en a une description partielle, abstraite.

Le projet est constitué de deux parties. Dans la première partie, on négligera l'aspect multilingue. Son objectif est la prise en main de l'analyseur et la réalisation d'expériences monolingues (on construit un analyseur pour chacune des langues qui nous intéressent). Dans la seconde partie, on ajoute à l'analyseur une représentation abstraite, partielle, de chaque langue, afin d'étudier les conséquences des différents aspects de cette représentation abstraite sur les performances de l'analyseur.

2 Partie I : analyse monolingue

Cette partie repose sur une implémentation qui vous est fournie d'un analyseur en transition. Le code de l'analyseur se trouve dans le dépôt `git` suivant : `https://gitlab.lis-lab.fr/alexis.nasr/tbp.git`. Les données sur lesquelles les expériences seront réalisées se trouvent à l'adresse suivante : `http://pageperso.lif.univ-mrs.fr/~alexis.nasr/Ens/TLNL/data.tgz`.

Le données sont constituées de fichiers au format `conllu`. Ces fichiers comportent des phrases enrichies de leur analyse syntaxique. Les données

couvrent 32 langues différentes. Pour chacune d'entre elles, trois fichiers sont fournis. Un fichier d'apprentissage, un fichier de développement et un fichier de test. Etant donné la langue dont le code est `fr` (il s'agit du français), ces fichiers ont respectivement pour noms `train_fr.conllu`, `dev_fr.conllu` et `test_fr.conllu`.

L'objectif de cette partie est d'entraîner 32 analyseurs syntaxiques différents (un par langue) dans exactement les mêmes conditions, c'est à dire : quasiment la même quantité de données d'apprentissage, la même fonction de décomposition (feature function) et les mêmes hyperparamètres du perceptron multicouche.

Afin de vous fournir un point de comparaison, la Table 1 reporte les résultats obtenus sur l'ensemble des langues pour un modèle appris sur 10.000 mots, avec un corpus de développement de 5.000 mots et un corpus de test de 700 mots. La structure du classifieur permettant de prédire une action à effectuer étant donné une configuration est la suivante :

```
model = Sequential()
model.add(Dense(units=128, activation='relu', input_dim=inputSize))
model.add(Dropout(0.4))
model.add(Dense(units=outputSize, activation='softmax'))
model.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])

model.fit(x_train, y_train, epochs=10, batch_size=32,
                  validation_data=(x_dev,y_dev))
```

La fonction de décomposition, qui correspond au fichier `basic.fm` est la suivante :

```
W B -2 POS
W B -1 POS
W B 0 POS
W B 1 POS
W B 2 POS
W S 0 POS
W S 1 POS
```

2.1 Ce qu'il faut faire

Votre travail pour cette partie consiste à constituer votre *baseline*. Il s'agit de votre système de référence qui vous servira, dans la seconde partie du

L	LAS	UAS	L	LAS	UAS	L	LAS	UAS
ar	60.14	68.75	fa	51.26	60.92	nl	54.00	62.93
bg	68.86	80.03	fr	69.52	74.57	nno	73.74	78.93
ca	61.83	68.59	he	63.32	68.86	nob	67.25	75.55
cs	70.83	79.23	hi	63.73	72.38	pl	71.58	85.08
da	66.17	73.06	hr	57.54	67.45	pt	70.97	75.51
de	56.44	64.49	hu	58.09	68.49	ro	53.72	64.96
el	71.13	78.64	id	69.09	74.14	sl	47.91	59.19
en	67.58	72.67	it	75.25	81.00	sv	63.47	71.55
es	66.52	72.75	ja	75.11	85.39	vi	49.46	50.83
et	59.64	73.24	ko	46.36	55.90	zh	45.42	54.86
eu	48.39	59.84	lv	54.69	62.24			

TABLE 1 – Labeled Accuracy Score (LAS) et Unlabeled Accuracy Score (UAS) pour 32 langues différentes dans des conditions d'apprentissage proches.

projet, à comparer une approche monolingue et multilingue. Pour cela, vous trouverez un jeux d'hyper-paramètres raisonnable (on entend ici par hyper-paramètres, les hyper-paramètres du classifieur proprement dit, mais aussi la taille des données d'apprentissage, de développement et de test, ainsi que la fonction de décomposition).

D'un point de vue lingustique la partie la plus intéressante concerne la mise au point de la fonction de décomposition. C'est en effet elle qui permet de déterminer les variables linguistiques qui semblent intéressantes pour réaliser une analyse syntaxique.

Dans le code qui vous est fourni, seules les *Word Features* sont implémentées, c'est à vous d'implémenter les *Configurational Features*.

A l'issue de cette partie, vous devrez fournir les résultats que vous avez obtenu pour les différentes langues, les hyper-paramètres que vous avez utilisé ainsi que la fonction de décomposition que vous avez définie. Plus une description de la méthode que vous avez suivie pour arriver à vos hyper-paramètres et à votre fonction de décomposition.

3 Partie II : analyse multilingue

La première partie du projet portait sur l'analyse monolingue délexicalisée. La seconde partie porte sur l'analyse multilingue, toujours délexicalisée. L'idée consiste à produire un seul analyseur pour nos 32 langues. Un tel analyseur

prend en entrée une séquence de parties de dicours correspondant à une phrase d'une des 32 langues et produit son analyse.

3.1 Le modèle Σ

La version la plus simple de l'analyseur multilingue consiste tout simplement à concaténer les différents corpus d'apprentissage et à entraîner un seul analyseur dessus. Nous appelerons de modèle, le modèle Σ. Il faut s'attendre à ce que ce modèle obtiennent de très mauvaises performances. En effet, certaines des 32 langues peuvent avoir, par exemple, le sujet situé avant le verbe ordre `SV`, tandis que d'autres ont le sujet situé après le verbe `VS`. L'analyseur n'ayant pas cette information, il peut très bien établir une dépendance sujet gauche (où le sujet est situé avant le verbe) pour une langue `VS` et, inversement, établir une dépendance sujet droite (où le sujet est situé après le verbe) pour une langue `SV`.

3.2 Le modèle ΣID

Une autre version simple de l'analyseur multilingue consiste à ajouter à chaque mot d'une phrae, une *Word Feature* qui représente le code la langue de la phrase (par exemple `fr`) pour le français. Cette information est déjà présente dans la colonne 11 des fichiers `conllu`. A l'aide de cette information, l'analyseur est capable de savoir, lorsqu'il analyse une phrase, la langue de cette dernière. Ce modèle sera appelé ΣID.

3.3 Le modèle ΣW

La troisième version (la plus intéressante) ne connaît pas la langue précise de la phrase analysée, en revanche, elle en connaît un certain nombre de caractéristiques. Ces dernières sont extraites du WALS et le modèle est appelé ΣW. Pour mettre au point ce modèle, vous devez :

— Sélectionner 12 features du WALS qui vous semblent pertientes pour un analyseur syntaxique. Par exemple, la feature `82A` du WALS, pourrait permettre de régler le problème d'ordre entre le sujet et le verbe mentionné ci-dessus pour le modèle Σ. En plus d'être pertinentes, ces features doivent être discriminantes : si elles prennent la même valeur pour chacune des 32 langues, elles n'apporteront aucune information pour aider l'analyseur. Vous trouverez les valeurs des différentes features du WALS pour les différentes langues, dans le fichier `values.csv` qui se trouve sur le site du cours. Les codes utilisés pour les différentes

langues dans le WALS ne sont pas les mêmes que dans UD, vous trouverez la correspondance entre les deux dans le tableau 2.

— Une fois les 12 features choisies, une langue L peut être représentée par un vecteur $W(L)$ de features de dimension 12 : les valeurs des 12 features pour L. Attention : toutes les features du WALS ne sont pas renseignées pour les 32 langues. Il faut donc trouver un moyen pour donner une valeur aux features non renseignées. A minima, on indiquera une valeur *joker*. A l'issue de cette étape, vous obtiendrez une matrice 32×12 dont chaque ligne correspond à un vecteur $W(L)$.

— Une fois les vecteurs des langues déterminés, il faut remplacer dans les fichiers de données (les fichiers `mcf`) le code d'une langue L par son vecteur $W(L)$, sous la forme de 12 nouvelles colonnes et modifier les fichiers `mcd` et `fm` en conséquence. Le fichier `fm` comportera maintenant 12 nouvelles *Word Features* correspondant aux 12 features du WALS que vous aurez sélectionné.

UD	WALS	Language	UD	WALS	Language	UD	WALS	Language
ar	ams	Arabic	fa	prs	Persian	nl	dut	Dutch
bg	bul	Bulgarian	fr	fre	French	nno	nor	Norwegian
ca	ctl	Catalan	he	heb	Hebrew	nob	nor	Norwegian
cs	cze	Czech	hi	hin	Hindi	pl	pol	Polish
da	dsh	Danish	hr	scr	Croatian	pt	por	Portugese
de	ger	German	hu	hun	Hungarian	ro	rom	Romanian
el	grk	Greek	id	ind	Indonesian	sl	slo	Solvenian
en	eng	English	it	ita	Italian	sv	swe	Swedish
es	spa	Spanish	ja	jpn	Japanese	vi	vie	Vietnamese
et	est	Estonian	ko	kor	Korean	zh	mnd	Chinese
eu	bsq	Basque	lv	lat	Latvian			

TABLE 2 – Correspondance entre les codes des langues pour UD et pour WALS.

3.3.1 Méthodologie pour la sélection des features du WALS

Nous avons évoqué ci-dessus que les features du WALS à utiliser devaient être pertinentes et discriminantes. Pour vous aider à déterminer quelles features sont pertinentes, vous pouvez vous concentrer sur un petit nombre de dépendances syntaxiques et vous poser la question de savoir quelles features peuvent être intéressantes pour mieux prédire ces dépendances. Pour reprendre l'exemple ci-dessus, on peut émettre l'hypothèse que la feature `82A` devrait améliorer la qualité de la prédiction de la dépendance sujet (`nsubj` et `csubj`). On peut tester cette hypothèse en introduisant cette feature dans

le modèle ΣW et en comparant les résultats de Σ et de ΣW. Il est possible d'étudier précisément les conséquences de l'introduction d'une feature en calculant le rappel, la précision et la F-mesure des dépendances auxquelles vous aurez décidé de vous intéresser sur le corpus de test. En comparant le rappel, la précision et la F-mesure de ces dépendances sur Σ et sur ΣW vous pourrez valider (ou invalider) votre hypothèse.

4 Ce qu'il faut rendre

Vous présenterez votre travail sous la forme d'un rapport d'une dizaine de pages.

Ce rapport devra nécessairement contenir les éléments suivants :

1. Les résultats des analyseurs mono-lingues. Ces résultats se présenterons sous la forme d'un tableau tel que tableau 1. Vous indiquerez de plus les conditions dans lesquelles ces résultats ont été obtenus : hyper-paramètres du classifieur et fonction de décomposition (fichier `fm`).
2. Les résulats des modèles Σ et ΣID dans les mêmes conditions que les expériences mono-lingues.
3. Pour le modèle ΣW :
   (a) La description des phénomènes (ensemble de dépendances syntaxiques $\mathcal{D}$) auxquelles vous avez décidé de vous intéresser.
   (b) Les features du WALS que vous avez sélectionné en expliquant la raison de ce choix (vos hypothèses). La matrice 32×12 des valeurs des différentes valeurs des features pour les 32 langues.
   (c) Les résultats, sous la forme d'un tableau identique au tableau 1, la précision, le rappel et la F-mesure des dépendances de $\mathcal{D}$ obtenues avec les différents modèles (principalement L (analyse mono-lingue), Σ et ΣW).
   (d) Une analyse qualitative des résultats et la validation (ou l'invalidation) des hypohtèses que vous aurez faites.

Bon courage !