MElt
Manuel de l'utilisateur

Benoît Sagot

Version 2.0

MElt est un système d'étiquetage séquentiel de textes. Plus précisément, l'archive MElt contient :

— Un ensemble de modèles d'étiquetage pour diverses langues, c'est-à-dire les données nécessaires à MElt pour étiqueter des corpus ;
— Le système MElt proprement dit, qui permet l'utilisation de ces modèles pour réaliser effectivement la tâche d'étiquetage, mais également pour construire de nouveaux modèles à partir de données annotées (corpus d'entraînement et, si disponible, lexique) ;
— Un lemmatiseur, qui peut être appelé sur option comme post-traitement après l'étiquetage ;
— Un ensemble de scripts de pré-traitement qui permettent, à la demande de l'utilisateur, de tokeniser un texte, de le segmenter en énoncés et d'y reconnaître certaines entités (URL, adresses e-mail, etc) avant de l'étiqueter ;
— Des outils, qui peuvent également être appelés sur simple option, qui permettent de traiter des données bruitées (textes issus du web, par exemple) en les normalisant à la volée, en étiquetant alors le texte normalisé, puis en redistribuant les étiquettes sur les mots (tokens) d'origine.

Ce document décrit comment installer et utiliser MElt. Il donne aussi quelques indications sur les enjeux scientifiques sous-jacents à son développement.

1 Installation

L'utilisation de base de MElt nécessite un système de type UNIX (par exemple, Linux, MacOS X ou encore Cygwin sous Windows) ainsi que les logiciels `sh`, Perl et Python. MElt peut être téléchargé sous forme d'une archive .tgz sur la page de MElt : `https://www.rocq.inria.fr/alpage-wiki/tiki-index.php?page=MElt`. L'installation se fait alors, comme pour tout logiciel reposant sur une architecture `autotools` (architecture standard pour le déploiement de logiciels sous UNIX), de la façon suivante :

— Dans un terminal (ou invite de commande), décompresser l'archive (`tar xvzf [nom_de_l'archive_téléchargée].tgz`), puis aller dans le dossier ainsi créé (`cd [nom_de_l'archive_téléchargée]`)
— Préparer la compilation en l'adaptant à son ordinateur et à son système d'exploitation : `./configure`
— Compiler : `make`
— Installer le résultat de la compilation dans un endroit accessible au système : `sudo make install`

Cette installation, standard, nécessite d'avoir les droits permettant d'exécuter la commande `sudo`. Une alternative est de remplacer `./configure` par `./configure -prefix=/chemin/vers/un/dossier/d'installation` (dossier où l'on doit avoir les droits d'écriture), de compiler avec `make` puis d'installer simplement par `make install`.

Pour tester l'installation, il suffit de lancer la commande suivante [1] :

```
echo "Ceci est un test" | MElt
```

Si tout s'est correctement installé, une réponse telle que celle ci-dessous doit être renvoyée :

```
Ceci/PRO est/V un/DET test/NC.
```

Si ce n'est pas le cas (« command not found », ou quelque chose de ce type), vérifier que la variable d'environnement `$PATH` contient bien le dossier d'installation de l'exécutable `MElt` (pour une installation par défaut, il s'agit le plus souvent de `/usr/local/bin`). Une autre possibilité est de remplacer les appels à `MElt` par un appel explicite avec le chemin complet [2].

Si l'on souhaite pouvoir entraîner de nouveaux modèles, il faut également installer le logiciel `megam` (`http://www.umiacs.umd.edu/~hal/megam/`), disponible soit sous forme binaire (exécutable UNIX, fichier .exe pour Windows) soit sous forme de fichiers sources, aux adresses suivantes :

- Exécutable Linux : `http://hal3.name/megam/megam_i686.opt.gz`
- Exécutable Mac OS X (processeurs Intel 64 bits, c'est-à-dire toute machine raisonnablement récente) : `http://alpage.inria.fr/~sagot/megam.opt`
- Code source `http://hal3.name/megam/megam_src.tgz`. Attention : la compilation de `megam` à partir des sources nécessite l'installation préalable du compilateur `ocaml` pour le langage `caml` (`http://caml.inria.fr`).

2 Utilisation

2.1 Étiquetage

MElt lit du texte, soit brut soit tokenisé et segmenté, sur l'entrée standard. Il l'étiquette puis écrit le résultat au format Brown sur sa sortie standard.

Autrement dit, après installation, la commande

```
echo "Ceci est un test" | MElt
```

doit renvoyer quelque chose comme

```
Ceci/PRO est/V un/DET test/NC.
```

On notera qu'il y a un temps de lancement pour MElt : étiqueter 10 phrases va à peine plus vite qu'en étiqueter une seule.

Par défaut, MElt utilise le modèle standard du français. Il suppose que le texte donné en entrée est déjà segmenté en phrases (une phrase par ligne) et tokenisé (l'espace étant le séparateur de tokens). Il considère de plus qu'il doit identifier les URL, les

1. Attention aux majuscules, l'exécutable s'appelle MElt, avec `ME` en majuscules suivi de `lt` en minuscules.
2. Par exemple `echo "Ceci est un test" | /usr/local/bin/MElt`

adresses e-mail et les smileys dans le texte avant de l'étiqueter, ces entités étant particulièrement problématiques à traiter sans reconnaissance explicite. Enfin, les éléments XML, à condition qu'ils constituent une ligne à eux seuls, sont laissés tels quels (qu'il s'agisse de balises seules, de séquences de balises ou de balises encadrant du contenu, lui-même laissé inchangé). Ainsi,

```
echo "Aller à http://www.inria.fr" | MElt
```

renverra quelque chose comme

```
Aller/VINF à/P {http://www.inria.fr}_URL/NPP.
```

Ainsi, l'URL elle-même a été mise entre accolades, suivies d'un identifiant d'entité, `_URL`. C'est cet identifiant que l'étiqueteur a traité comme un mot et a étiqueté, ici comme un nom propre.

Toutes ces décisions par défaut peuvent être modifiées en donnant des options à MElt. Pour un aperçu complet des options disponibles dans une version donnée de MElt, il est toujours possible de faire apparaître l'aide en ligne par la commande `MElt -h`. Les options principales peuvent être classées comme suit :

- Sélection du modèle :
  - `-m` Chaque modèle d'étiquetage est un dossier comportant plusieurs fichiers. Par l'option `-m [/chemin/vers/un/modèle]`, on spécifie à MElt quel modèle utiliser. Les options `-l` et `-m` sont incompatibles, puisqu'elles constituent deux manières concurrentes de spécifier un modèle.
  - `-l` L'option `-l [langue]`, spécifie à MElt d'utiliser le modèle par défaut pour une langue donnée. Dans le cas d'une installation standard, `-l [langue]` est équivalent à `-l /usr/loca/share/melt/[langue]`.
  - — Exemple : pour étiqueter une phrase en anglais, on peut par exemple utiliser la commande suivante :
    `echo "This is a test" | MElt -l en`
    qui renvoie en principe
    `This/DT is/VBZ a/DT test/NN.`
- — Tokenisation, segmentation et détection de certaines entités (comme indiqué ci-dessus, MElt considère par défaut que le texte est tokenisé, mais identifie malgré tout les URL, adresses e-mails et smileys) :
  - `-t` Spécifie à MElt qu'il doit segmenter en phrases et tokeniser le texte avant de l'étiqueter, et ce au moyen de modules extraits de la chaîne SxPipe et intégrés à MElt. Cette option n'est pas compatible avec l'option `-c` décrite plus bas, car elle l'implique automatiquement. Les éléments XML, à condition qu'ils constituent une ligne à eux seuls, sont laissés tels quels (qu'il s'agisse de balises seules, de séquences de balises ou de balises encadrant du contenu, lui-même laissé inchangé). Incompatible avec `-T`.
  - `-T` Comme `-t`, mais le tokeniseur/segmenteur utilisé est un tokeniseur qui tente d'appliquer les conventions de tokensiation et segmentation utilisées par l'analyseur syntaxique Bonsai. Incompatible avec `-t`.
    Note : pour l'instant, les éléments XML ne sont pas gérés correctement par ce tokeniseur.
  - `-r` Spécifie à MElt qu'il ne doit pas chercher à détecter les URL, adresses e-mail et smileys avant d'appliquer éventuellement le tokeniseur à la Bonsai (option `-T`, soit avant d'étiqueter directement (si ni `-t` ni `-T` ne sont donnés). Si le tokeniseur/segmenteur issu de SxPipe est utilisé, cette option n'a

aucun effet puisque ce dernier détecte nécessairement les URL, adresses e-mail et smileys.

`-C` Ne pas étiqueter, sortir le résultat des prétraitements et/ou de la normalisation. Exemple :
`echo "Aller à http://www.inria.fr" | MElt`
doit renvoyer :
`Aller à {http://www.inria.fr}_URL`

— Lemmatisation :

`-L` Utilise le lemmatiseur en post-traitement. Exemple : `echo "Ceci est un test" | MElt -L`
doit renvoyer
`Ceci/PRO/ceci est/V/être un/DET/un test/NC/test`
Les lemmes pour les mots inconnus du lexique sont précédés d'un symbole *

Note : ce lemmatiseur n'est pas disponible dans tous les modèles distribués avec MElt. En effet, les données nécessaires au lemmatiseur pour fonctionner ne sont construites correctement lors de la construction d'un modèle que si le jeu d'étiquette du lexique externe est identique au jeu d'étiquettes du corpus d'entraînement (cette contrainte n'esst pas nécessaire pour la seule construction du modèle d'étiquetage proprement dit).

— Traitement de données bruitées (disponible uniquement pour français et l'anglais) :

`-n` Normaliser le texte avant de l'étiqueter, et restaurer après étiquetage les tokens d'origine. Les étiquettes sont redistribuées sur les tokens d'origine ; l'étiquette `x` est attribuée aux amalgames non-standard (type *chépa*), et tous les composants de composés non-standard sont étiquetés `y` (par exemple, les deux premiers tokens de *c a dire*). Exemple complet :
`echo "c a dire ke c un truc de ouf" | MElt -n`
doit renvoyer :
`c/Y a/Y dire/Y ke/CS c/X un/DET truc/NC de/P ouf/ADJ`

`-N` Normaliser le texte avant de l'étiqueter, et restaurer après étiquetage les tokens d'origine. Les étiquettes sont redistribuées sur les tokens d'origine ; les amalgames non-standard (type *chépa*) sont étiquetés avec la liste des étiquettes des mots amalgamés séparés par le symbole + (ici, `cls+v+advneg`), et les composants de composés non-standard sont étiquetés `y` ou `gw` (par exemple, les deux premiers tokens de *c a dire*), sauf le dernier composant qui reçoit l'étiquette du composé complet. Exemple complet :
`echo "c a dire ke c un truc de ouf" | MElt -n`
doit renvoyer :
`c/Y a/Y dire/CC ke/CS c/CLS+V un/DET truc/NC de/P ouf/ADJ`

`-C` (rappel) Ne pas étiqueter, sortir le résultat des prétraitements et/ou de la normalisation. Exemple :
`echo "c un truc de ouf" | MElt -n -C`
doit renvoyer :
`{c} c' {} est un truc de {ouf} fou`
On notera que la correspondance entre tokens d'origine et tokens normalisés est donnée via les contenus entre accolades : un token modifié est placé entre accolades avant le ou les tokens modifés qui lui correspondent (s'il y en a plusieurs, les tokens modifiés suivants sont précédés d'une paire d'accolades sans contenu).

— Options diverses :

`-c` Gère les « commentaires » à la SXPipe. Autrement dit, toute séquence entre accolades est ignorée. Implique `-r`, incompatible avec `-T` ou `-t`.

`-d` Toute phrase intégralement en majuscules et minusculisée avant étiquetage.

`-P` Ajouter derrière chaque étiquette la probabilité que lui a assignée le modèle d'étiquetage.

`-e` Ne pas exécuter MElt, sortir la commande complète telle qu'elle aurait été exécutée sans cette option.

`-h` Renvoie un message d'aide décrivant toutes les options disponibles.

`-v` Renvoie la version de MElt.

2.2 Entraînement de nouveaux modèles

L'entraînement de nouveaux modèles nécessite d'avoir préalablement installé `megam` (voir plus haut). Pour entraîner un nouveau modèle MElt, il faut disposer des données suivantes (des exemples sont donnés ci-dessous) :

— un **corpus d'entraînement** étiqueté au format Brown, c'est-à-dire qui respecte les conventions suivantes :
  — une phrase par ligne ;
  — le caractère d'espace est le séparateur de mots (tokens) ;
  — chaque mot est immédiatement suivi du symbole « / » puis d'une étiquette, qui ne doit pas comporter de symbole « / ».

— un **lexique externe**, qui peut être un fichier vide (pour le cas où l'on n'a pas ou ne souhaite pas utiliser de lexique externe), mais qui est typiquement un lexique au format texte à 3 ou 4 colonnes (la quatrième étant facultative) séparées par des tabulations. Chaque ligne est une entrée lexicale, répartie sur les 3 ou 4 colonnes comme suit :

  colonne 1 Le mot (token). Il ne doit comporter aucun espace (les composants de mots composés doivent être séparés par des blancs soulignés)

  colonne 2 Une catégorie (note : pour que l'entraînement du modèle construise également un lemmatiseur, cette catégorie doit être cohérente avec les catégories utilisées dans le corpus ; pour le seul entraînement du modèle d'étiquetage, il n'est pas nécessaire de respecter cette contrainte, puisque le fait que le mot ait cette catégorie dans le lexique externe est utilisé comme trait et non comme contrainte (par exemple, à aucun moment de l'étiquetage les catégories issues du lexiques ne sont comparées à celles issues du corpus ; en revanche, une telle comparaison est nécessaire pour le lemmatiseur).

  colonne 3 Un lemme. Si l'on ne souhaite pas ou ne peut pas entraîner de lemmatiseur en même temps qu'un étiqueteur, on peut systématiquement remplir cette colonne par un faux lemme (l'usage est de mettre $\star$ à toutes les entrées).

  colonne 4 (facultative et expérimentale) Soit 0 soit 1, selon que l'entrée lexicale est morphologiquement régulière. Des travaux en cours visent à construire automatiquement une telle information et à en étudier l'utilité pour un étiqueteur, notamment pour améliorer l'étiquetage des mots inconnus. Il est fortement déconseillé d'utiliser des lexiques à 4 colonnes à ce stade du développement de MElt.

Exemple de corpus :

```
Ceci/PRO est/V un/DET petit/ADJ corpus/NC ./PONCT
Il/CLS ne/ADV comprend/V que/ADV deux/DET phrases/NC ./PONCT
```

Exemple de lexique :

```
pondérerions   V    pondérer
domptions      VS   dompter
domptions      V    dompter
cahotements    NC   cahotement
translatés     VPP  translater
schématique    ADJ  schématique
```

3 Principes de fonctionnement

De nombreux systèmes pour l'étiquetage automatique en parties du discours ont été développés pour un large éventail de langues. Parmi les systèmes les plus performants, on trouve ceux qui s'appuient sur des techniques d'apprentissage automatique [3]. Pour certaines langues comme l'anglais ou d'autres langues très étudiées, ces systèmes ont atteint des niveaux de performance proches des niveaux humains. Il est intéressant de constater que la majorité de ces systèmes n'ont pas recours à une source externe d'informations lexicales, et se contentent d'un lexique extrait « à la volée » à partir du corpus d'apprentissage (cf. cependant [Hajič, 2000]).

L'un des objectifs scientifiques ayant conduit au développement de MElt est précisément l'étude de la faisabilité et de l'impact de l'utilisation de telles ressources lexicales externes. Nous avons pu démontrer [Denis and Sagot, 2009, Denis and Sagot, 2010, Denis and Sagot, 2012] qu'il y a là matière à améliorations dans diverses directions :

— les étiqueteurs ainsi construits permettent en un meilleur traitement des mots « inconnus », c'est-à-dire des mots absents du corpus d'apprentissage, dès lors qu'ils sont présents dans le lexique externe ;

— le temps de développement des ressources (corpus d'entraînement, lexique externe) nécessaire pour atteindre un niveau de performances fixé à l'avance est moins élevé si l'on développe en parallèle un corpus annoté et un lexique externe ; autrement dit, si l'on ne dispose pas à l'avance d'un lexique externe, ne développer qu'un corpus annoté ne constitue pas la solution optimale en temps pour avoir construite en un temps donné le meilleur étiqueteur.

Nous avons également montré la pertinence de notre approche sur plusieurs langues. Si l'on entraîne le système MElt sur le Corpus Arboré de Paris 7 [Abeillé et al., 2003] et en utilisant le lexique Le*fff* [Sagot, 2010] comme lexique externe, l'étiqueteur obtenu est état-de-l'art pour le français. À ce jour avons entraîné avec succès des modèles d'étiquetage (c'est-à-dire des « instances » de MElt) pour l'anglais, l'espagnol, l'italien, l'allemand, le néerlandais, le portugais, le polonais et le persan. La plupart de ces modèles sont distribués avec le système MElt proprement dit.

Pour présenter les principes sous-jacents au système d'étiquetage MElt, nous présenterons tout d'abord le modèle dit « de base », qui n'exploite pas de lexique externe. Puis nous montrerons comment nous intégrons les informations provenant d'un lexique externe.

3. Se reporter à [Manning and Schütze, 1999] pour un panorama complet.

De manière générale, MElt repose sur des modèles discriminants linéaires et séquentiels. L'idée sous-jacente à de tels modèles est que les décisions (ici, les choix d'étiquettes) se font de gauche à droite, en s'appuyant sur un certain nombre de traits (*features*). Ces traits peuvent être de diverses natures : le ou les mots qui précèdent (2 au plus, sinon les statistiques que l'on fait sur le corpus d'entraînement ne sont plus significatives), le ou les mots qui suivent (même remarque), les préfixes et suffixes de ces mots (au sens informatique), mais également les étiquettes déjà attribuées aux mots qui précèdent (les mots suivants n'ayant pas encore reçu d'étiquette). Dans le cas de l'utilisation d'un lexique externe, des traits issus de ce lexique pourront être utilisés également. Ces traits sont choisis comme étant tous booléens : chaque trait est ou bien présent ou bien absent (par exemple, il n'y a pas de trait « étiquette du mot précédent », qui pourrait valoir `v` ou `det`, mais plusieurs traits de la forme « l'étiquette du mot précédent est `v` », qui peut donc valoir Vrai ou Faux). L'idée générale est alors la suivante : pour chaque trait, l'algorithme d'apprentissage calcule pour chaque étiquette un poids, qui indique combien ce trait, s'il est Vrai, renforce l'hypothèse selon laquelle cette étiquette est la bonne. Par exemple, le trait « l'étiquette du mot précédent est `det` » est susceptible d'avoir un poids positif relativement élevé en faveur des étiquettes `nc` et `adj`, mais un poids très négatif pour l'étiquette `v` : en effet, il est vraisemblable qu'un mot précédé d'un déterminant soit un nom ou un adjectif, et très peu vraisemblable qu'il soit un verbe.

Plusieurs types de modèles permettent de calculer de tels poids à partir de données d'entraînement (corpus d'apprentissage annoté manuellement et, le cas échéant, lexique externe). Les premières versions de MElt ne permettaient que l'entraînement de modèles markoviens à maximum d'entropie (Maximum Entropy Markov Models, MEMM). Désormais, lors de l'entraînement d'un nouveau modèle, l'utilisateur peut choisir d'utiliser un algorithme d'apprentissage de type MEMM ou un perceptron multiclasse, ce dernier étant bien plus rapide à entraîner mais conduisant en général à des modèles très légèrement moins précis que les MEMM.

3.1 Le modèle de base

Étant donné un jeu d'étiquettes T et une chaîne de mots (tokens) w_1^n, on définit la tâche d'étiquetage comme le processus consistant à assigner à w_1^n la séquence d'étiquettes $\hat{t}_1^n \in T^n$ de vraisemblance maximale. Suivant [Ratnaparkhi, 1996], on peut alors approcher la probabilité conditionnelle $P(t_1^n|w_1^n)$ de sorte que :

$$\hat{t}_1^n = \arg\max_{t_1^n \in T^n} P(t_1^n|w_1^n) \approx \arg\max_{t_1^n \in T^n} \prod_{i=1}^{n} P(t_i|h_i), \tag{1}$$

où t_i est l'étiquette du mot w_i et h_i est le *contexte* de (w_i, t_i), qui comprend la séquence des étiquettes déjà assignées t_1^{i-1} et la séquence des mots w_1^n.

Dans un modèle à maximisation d'entropie, on estime les paramètres d'un modèle exponentiel qui a la forme suivante :

$$P(t_i|h_i) = \frac{1}{Z(h)} \cdot \exp\left(\sum_{i=j}^{m} \lambda_j f_j(h_i, t_i)\right) \tag{2}$$

Les f_1^m sont des traits, fonctions définies sur l'ensemble des étiquettes t_i et des historiques h_i (avec $f(h_i, t_i) \in \{0, 1\}$), les λ_1^m sont les paramètres associés aux f_1^m, et

$Z(h)$ est un facteur de normalisation sur les différentes étiquettes. Dans ce type de modèle, le choix des paramètres est assujetti à des contraintes garantissant que l'espérance de chaque trait soit égale à son espérance empirique telle que mesurée sur le corpus d'apprentissage [Berger et al., 1996]. Le système MElt estime ces paramètres en utilisant l'algorithme dit *Limited Memory Variable Metric Algorithm* [Malouf, 2002] implémenté au sein du système Megam [4] [Daumé III, 2004].

Les classes de traits que nous avons utilisées pour la conception du modèle de base, c'est-à-dire du modèle n'utilisant pas de lexique externe, est un sur-ensemble des traits utilisé par [Ratnaparkhi, 1996] et [Toutanova and Manning, 2000] pour l'anglais (qui étaient largement indépendants de la langue). Ces traits peuvent être regroupés en deux sous-ensembles. Le premier rassemble des traits dits *internes* qui essaient de capturer les caractéristiques du mot à étiqueter. Il s'agit notamment du mot w_i lui-même, de ses préfixes et suffixes de longueur 1 à 4, ainsi que de traits booléens qui testent si w_i contient ou non certains caractères particuliers comme les chiffres, le tiret ou les majuscules. Le deuxième ensemble de traits, dits *externes*, modélise le contexte du mot à étiqueter. Il s'agit tout d'abord des mots qui sont dans les contextes gauche et droit de w_i (à une distance d'au plus 2). Ensuite, nous intégrons comme traits l'étiquette t_{i-1} assignée au mot précédent, ainsi que la concaténation des étiquettes t_{i-1} et t_{i-2} pour les deux mots précédents w_i. La liste détaillée des classes de traits utilisées dans le système de base est indiquée à la table 1.

Traits internes	
$w_i = X$	& $t_i = T$
Préfixe de $w_i = P, \|P\| < 5$	& $t_i = T$
Suffixe de $w_i = S, \|S\| < 5$	& $t_i = T$
w_i contient un nombre	& $t_i = T$
w_i contient un tiret	& $t_i = T$
w_i contient une majuscule	& $t_i = T$
w_i contient uniquement des majuscules	& $t_i = T$
w_i contient une majuscule et n'est pas le premier mot d'une phrase	& $t_i = T$
Traits externes	
$t_{i-1} = X$	& $t_i = T$
$t_{i-2}t_{i-1} = XY$	& $t_i = T$
$w_{i+j} = X, j \in \{-2, -1, 1, 2\}$	& $t_i = T$

TABLE 1 – Traits de base utilisés par le système de base

Une différence importante avec le jeu de traits de [Ratnaparkhi, 1996] vient du fait que nous n'avons pas restreint l'application des traits de type préfixes et suffixes aux mots qui sont rares dans le corpus d'apprentissage. Dans notre modèle, ces traits sont toujours construits, même pour les mots fréquents. En effet, nous avons constaté lors du développement [5] que la prise en compte systématique de ces traits conduit à de meilleurs résultats, notamment sur les mots inconnus. De plus, ces traits sont probablement plus discriminants sur des langues à morphologie plus riche que l'anglais, comme le français ou l'italien, que sur l'anglais, langue traitée par [Ratnaparkhi, 1996]. Une autre différence entre notre modèle de base et les travaux antérieurs concerne le lissage. [Ratnaparkhi, 1996] et [Toutanova and Manning, 2000] seuillent leurs traits à un nombre d'occurrence de 10 pour éviter les données statistiquement non significatives.

4. Librement disponible à l'adresse `http://www.cs.utah.edu/~hal/megam/`.

5. Plus précisément, lors du développement du MElt du français entraîné sur le Corpus Arboré de Paris 7 et sur le lexique L*efff*.

Nous n'avons pas seuillé nos traits mais avons utilisé à la place une régularisation gaussienne sur les poids [6], ce qui est une technique de lissage plus motivée statistiquement [7].

3.2 Intégration des informations lexicales dans l'étiqueteur

L'avantage du modèle de base est de permettre un ajout aisé de traits supplémentaires, y compris de traits dont les valeurs sont calculées à partir d'une ressource externe au corpus d'apprentissage.

Pour chaque mot w_i, nous générons une nouvelle série de traits internes basés sur la présence (ou non) de w_i dans le lexique externe et, le cas échéant, les étiquettes associées à w_i par ce même lexique. Si w_i est associé à une étiquette unique t_i, nous générons un trait qui encode l'association non ambiguë entre w_i et $t_{i,0}$. Lorsque w_i est associé à plusieurs étiquettes $t_{i,0}, \ldots, t_{i,m}$ par le lexique externe, nous générons un trait interne pour chacune de ses étiquettes possibles $t_{i,j}$, ainsi qu'un trait interne qui représente la disjonction de m étiquettes. Enfin, si w_i n'est pas recensé dans le lexique externe, nous créons un trait spécifique qui encode le statut d'inconnu du lexique [8].

De même, nous utilisons le lexique externe pour construire de nouveaux traits externes : nous construisons l'équivalent des traits internes pour les mots des contextes gauche et droit à une distance de moins de 2 du mot courant. Nous générons également des traits bigrammes correspondant à la concaténation des étiquettes du lexique externe pour les 2 mots à gauche, les 2 mots à droite, et les deux mots qui entourent w_i. Lorsque ces mots sont ambigus pour le lexique externe, seule leur disjonction contribue au bigramme, et si l'un ce ces mots est inconnu, la valeur *unk* tient lieu d'étiquette [9].

La liste détaillée des classes de traits utilisées par le modèle complet, en plus de ceux de la table 1, est indiquée à la table 2. Ce jeu de traits, présenté dans [Denis and Sagot, 2010], étend légèrement celui présenté par [Denis and Sagot, 2009].

Traits lexicaux internes	
$t_i =_{uni} X$, if $\mathsf{lex\_ext}(w_i) = \{X\}$	& $t_i = T$
$t_i = X, \forall X \in \mathsf{lex\_ext}(w_i)$ if $\lvert\mathsf{lex\_ext}(w_i)\rvert > 1$	& $t_i = T$
$t_i = \bigvee \mathsf{lex\_ext}(w_i)$ if $\lvert\mathsf{lex\_ext}(w_i)\rvert > 1$	& $t_i = T$
$t_i = unk$, if $\mathsf{lex\_ext}(w_i) = \emptyset$	& $t_i = T$
Traits lexicaux externes	
$t_{i+j} = \bigvee \mathsf{lex\_ext}(w_{i+1}), j \in \{-2, -1, 1, 2\}$	& $t_i = T$
$t_{i+j}t_{i+k} = \bigvee \mathsf{lex\_ext}(w_{i+j}) \bigvee \mathsf{lex\_ext}(w_{i+k}), (j,k) \in \{(-2,-1),(+1,+2),(-1,+1)\}$	& $t_i = T$

TABLE 2 – Traits lexicaux ajoutés au modèle de base dans le modèle complet

Ces différents traits permettent d'avoir une information, ne serait-ce qu'ambiguë, sur les étiquettes dans le contexte droit du mot, ce que ne permettent pas les traits de base. Ceux-ci n'incluent que les étiquettes sur le contexte gauche, les seules à pouvoir

6. Plus précisément, nous utilisons un distribution gaussienne avec une précision (c'est-à-dire une variance inverse) de 1, ce qui est la valeur par défaut dans Megam ; d'autres valeurs ont été testées pendant la phase de développement, mais n'ont pas amélioré les résultats.

7. Informellement, l'effet de ce type de régluarisation est de pénaliser les poids artificiellement élevés en forçant l'ensemble des poids à respecter une certaine distribution *a priori*, ici une distribution gaussienne centrée sur zéro.

8. Pour les mots qui apparaissent en position initiale dans la phrase, nous vérifions au prélalable que la version décapitalisée du mot n'est pas présente non plus dans le lexique externe.

9. Différentes valeurs de « fenêtre » ont été essayées lors de la phase de développement : 1, 2 et 3. Bien que le passage de 1 à 2 mène à une amélioration significative, le passage de 2 à 3 mène à une légère dégradation.

être intégrées dans un décodage gauche-droite. Par ailleurs, cette manière d'intégrer les informations issues du lexique au modèle sous forme de traits supplémentaires a l'avantage de ne pas ajouter de contraintes *fortes*, et d'être ainsi robuste à d'éventuelles erreurs ou incomplétudes du lexique.

Une autre façon, plus directe, d'exploiter une ressource lexicale exogène consiste en effet à utiliser les informations lexicales comme *filtre*. A savoir, on contraint l'étiqueteur à choisir pour un mot w une étiquette correspondant soit à une occurrence de w dans le corpus, soit à une entrée du lexique pour w. C'est l'approche employée par exemple par [Hajič, 2000] pour des langues à morphologie très riche, et notamment pour le tchèque. Dans [Denis and Sagot, 2009], nous avons montré sur notre étiqueteur MElt du français que cette stratégie ne permet d'améliorer que marginalement les performances du modèle de base, et restent largement en-deçà de celles du modèle complet tel que présenté ci-dessus.

3.3 Décodage

La procédure de décodage (c'est-à-dire l'étiquetage proprement dit une fois le modèle construit) repose sur un algorithme de type *beam search* pour trouver la séquence d'étiquettes la plus probable pour une phrase donnée. Autrement dit, chaque phrase est décodée de gauche à droite, et l'on conserve pour chaque mot w_i les n séquences d'étiquettes candidates les plus probables du début de la phrase jusqu'à la position i. Pour nos expériences, nous avons utilisé un *beam* de taille 3 [10]. De plus, la procédure de test utilise un *dictionnaire d'étiquettes*) qui liste pour chaque mot les étiquettes qui lui sont associées dans le corpus d'apprentissage. Ceci réduit considérablement l'ensemble des étiquettes parmi lesquelles l'étiqueteur peut choisir pour étiqueter un mot donné, ce qui conduit, comme le montrent nos expériences, à de meilleures performances tant en termes de précision que d'efficacité en temps.

3.4 Normalisation et étiquetage de textes bruités

à rédiger

Références

[Abeillé et al., 2003] Abeillé, A., Clément, L., and Toussenel, F. (2003). Building a treebank for French. In Abeillé, A., editor, *Treebanks*. Kluwer, Dordrecht.

[Berger et al., 1996] Berger, A., Pietra, S. D., and Pietra, V. D. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1) :39–71.

[Daumé III, 2004] Daumé III, H. (2004). Notes on CG and LM-BFGS optimization of logistic regression. Paper available at `http://pub.hal3.name#daume04cg-bfgs`, implementation available at `http://hal3.name/megam/`.

[Denis and Sagot, 2009] Denis, P. and Sagot, B. (2009). Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art POS tagging with less human effort. In *Proceedings of PACLIC 2009*, Hong Kong.

10. Nous avons essayé d'autres valeurs (5, 10, 20) pendant le développement du MElt du français, mais ces valeurs n'ont pas conduit à des variations significatives de performance.

[Denis and Sagot, 2010] Denis, P. and Sagot, B. (2010). Exploitation d'une ressource lexicale pour la construction d'un étiqueteur morphosyntaxique état-de-l'art du français. In *Traitement Automatique des Langues Naturelles : TALN 2010*, Montréal, Canada.

[Denis and Sagot, 2012] Denis, P. and Sagot, B. (2012). Coupling an annotated corpus and a lexicon for state-of-the-art POS tagging. *Language Resources and Evaluation*.

[Hajič, 2000] Hajič, J. (2000). Morphological Tagging : Data vs. Dictionaries. In *Proceedings of ANLP'00*, pages 94–101, Seattle, WA, USA.

[Malouf, 2002] Malouf, R. (2002). A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Workshop on Natural Language Learning*, pages 49–55, Taipei, Taiwan.

[Manning and Schütze, 1999] Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

[Ratnaparkhi, 1996] Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Proceedings of International Conference on Empirical Methods in Natural Language Processing*, pages 133–142.

[Sagot, 2010] Sagot, B. (2010). The Le*fff*, a freely available, accurate and large-coverage lexicon for french. In *Proceedings of the 7th Language Resource and Evaluation Conference*, La Valette, Malte. à paraître.

[Toutanova and Manning, 2000] Toutanova, K. and Manning, C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of International Conference on New Methods in Language Processing*, pages 63–70, Hong Kong.