Implantation et analyse d'un modèle graphique à entraînement supervisé, semi-supervisé et non-supervisé pour la désambiguïsation sémantique

Yoshua Bengio et Hugo Larochelle

Abstract

La désambiguïsation sémantique est un sujet qui suscite beaucoup d'intérêt dans la communauté scientifique en apprentissage machine. Quoique cette tâche ait été abordée depuis les débuts du traitement automatique de la langue, peu de progrès ont été accomplis jusqu'à maintenant. Je présente ici une application de désambiguïsation basée sur un modèle graphique probabiliste. Ce modèle a été appris sur des données étiquetées, non-étiquetées, et sur la hiérarchie WordNet. Avec peu d'examples d'apprentissage, ses performances sont comparables à celles de l'algorithme de Bayes naïf. Il pourrait éventuellement être adapté à des corpus bi-textes.

1 Introduction

La désambiguïsation attire actuellement beaucoup d'attention en traitement automatique de la langue. Cette tâche, quoique n'étant pas une fin en soi [1], a été identifiée comme étant IA complète [2], i.e. qu'elle nécessiterait la résolution de tous les problèmes courants en apprentissage machine. Elle consiste simplement à identifier le sens correct d'un mot ambigüe (i.e. ayant plus d'un sens), à l'aide d'information contextuelle. Elle peut donc être vue comme une tâche de classification multi-classe, mais où l'ensemble des classes possibles est fonction du mot cible. Des percées dans ce domaine auraient un impact bénéfique dans d'autres tâches, comme la traduction automatique, l'extraction de terminologie, la recherche d'information et la catégorisation de textes.

Plusieurs approches variées ont été proposées jusqu'ici. Il y a les méthodes Lesk [3] datant des années 80 et qui utilisent directement les mots de la définition des sens afin de déterminer l'étiquette sémantique d'un mot cible. Il y a aussi les méthodes connectionnistes (e.g. [4]), les listes de décisions [5], les méthodes d'apprentissage basé sur des exemples ou "instance based learning" [6], et encore beaucoup d'autres.

Cependant, chacune des méthodes citées ci-dessus utilisent seulement de l'information lexicale ou syntaxique, e.g. en considérant seulement les mots contenus dans une fenêtre de longueur donnée (normalement 5 ou 7 mots) centrée sur le mot cible. La modélisation en est alors une de la relation entre les mots et leurs sens, et non pas des sens entre eux.

Pourtant, on image facilement qu'il y ait un lien entre les sens des mots d'un texte. C'est à ce lien qu'on fait implicitement référence lorsqu'on parle du sujet d'un texte. On propose donc une approche utilisant un modèle graphique, qui tente de modéliser une relation sémantique liée à ce concept de sujet ("topic").

2 Présentation du modèle

Le modèle graphique contient trois types de variables: les mots ou lemmes l_i contenus dans une fenêtre de longueur n (valeur à calibrer), les sens associés s_i et un sujet t. On ne considère donc que les mots ayant un ou plusieurs sens, soit les noms, verbes, adverbes et adjectifs. Comme on n'observe pas le sujet, alors on devra aussi calibrer le nombre T de valeurs possibles pour t. On peut voir sur la figure 1 que les seules relations de dépendances sont entrent les mots et leur sens respectif, puis entre les sens et le sujet. Bien sûr, il est irréaliste de croire qu'aucun lien direct n'existe entre chacuns des mots, mais on croit que ceux-ci ne sont pas nécessaires pour une modélisation sémantique raisonnable.

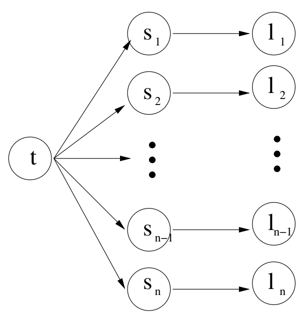


Figure 1: Modèle graphique

Le modèle graphique implique alors que la probabilité jointe de toutes les variables du modèle peut être écrite comme suit:

$$P(t, s_1^n, l_1^n) \quad = \quad P(t) \prod_{i=1}^{n} P(l_i|s_i) P(s_i|t) \tag{1}$$

où $P(t)$ et $P(s|t)$ sont les tables de probabilités à apprendre, et où $P(l|s)$ est estimée directement sur les données d'entraînement par simple comptage. On note aussi que les tables $P(s|t)$ et $P(l|s)$ ne dépendent pas de la position i.

3 Désambiguïsation avec le modèle

Afin de choisir les sens s_1^n corrects, on peut simplement choisir, pour chaque position i, le sens qui maximise $P(s_i|l_1^n)$. De plus, puisqu'on a accès aux étiquettes grammaticales ou "part of speech" (POS) des mots dans la plupart des corpus disponibles, on peut réduire directement le nombre de sens possibles en fonction de cette information.

On aimerait cependant profiter de la notion de sujet, qui a inspirée l'approche courante. On doit donc désambiguïser en choisissant la séquence de sens s_1^n qui maximise:

$$P(s_1^n|l_1^n) = (1/Z)P(s_1^n, l_1^n) = (1/Z)\sum_t P(t, s_1^n, l_1^n)$$

avec $Z = \sum_{s_1^n} P(s_1^n, l_1^n)$. Puisque Z a la même valeur pour chaque s_1^n, alors on peut simplement maximiser $P(s_1^n, l_1^n)$.

Malheureusement, s'il y a en moyenne M sens possibles pour chaque lemme, alors il y a en moyenne M^n séquences possibles. Le temps de calcul est donc trop coûteux.

Une alternative est d'utiliser une heuristique, afin de sélectionner seulement quelques unes des séquences possibles, pour ensuite comparer leur valeur de $P(s_1^n, l_1^n)$ associée.

On propose ici de faire cette sélection en restreignant les sens possibles de chacun des mots l_i. Pour ce faire, on va se baser sur l'ambiguïté a posteriori individuelle des l_i, i.e. la probabilité $P(s_i|l_1^n)$. Les sens possibles du mot cible l_i peuvent alors être classés par ordre de leur probabilité $P(s_i|l_1^n)$, et on peut ensuite sélectionner un sous-ensemble de ceux-ci, e.g. en prenant les k meilleurs sens dont la probabilité est plus grande que la moyenne (i.e. $1/M$ s'il y a M sens possibles). Les séquences s_i^n retenues sont alors celles qui peuvent être générées par les sous-ensembles créés.

On calcule $P(s_i|l_1^n)$ en utilisant le modèle, pour chaque position i, comme suit:

$$P(s_i|l_1^n) = \sum_t P(s_i|l_i, t) P(t|l_1^n) \tag{2}$$

où $P(s_i|l_i, t)$ peut être calculée comme suit:

$$P(s_i|l_i, t) = \frac{P(l_i|s_i) P(s_i|t)}{\sum_{s_i \in sens(l_i)} P(l_i|s_i) P(s_i|t)} \tag{3}$$

et où on peut calculer d'avance la probabilité a posteriori de t

$$P(t|l_1^n) = \frac{P(t) \prod_{i=1}^n P(l_i|t)}{\sum_t P(t) \prod_{i=1}^n P(l_i|t)} \tag{4}$$

en utilisant

$$P(l_i|t) = \sum_{s \in sens(l_i)} P(l_i|s_i = s) P(s_i = s|t) \tag{5}$$

En bref, on doit:

- calculer (5) de $i = 1$ à n, $t = 1$ à T, en temps $O(MnT)$, et sauvegarder alors les résultats pour chacune des $T \times n$ paires (l_i, t);
- calculer (4) de $t = 1$ à T, en temps $O(Tn)$, et sauvegarder les résultats pour chaque valeur de T;
- calculer (3), puis (2) de $i = 1$ à n, pour chaque sens possible s de l_i, en temps $O(MnT)$, et sauvegarder les résultats pour chacun des $M \times n$ sens possibles.

4 Apprentissage dans le modèle

Pour apprendre $P(s_i|t)$ et $P(t)$, on va utiliser l'algorithme EM, qui peut être adapté à toute sorte de données (du moins en principe), e.g.:

- des données totalement étiquetées avec s_1^n et l_1^n;
- des données partiellement étiquetées avec des sens sur seulement un sous-ensemble des lemmes;
- des données totalement non-étiquetées l_1^n;
- des "pseudo-données" qui renforcent le lissage des probabilités $P(s_i, s_j)$ ou $P(l_i, s_j)$ à l'aide de poids de pseudo-occurences $q(s_i, s_j)$ ou $q(l_i, s_j)$.

À titre de données totalement et partiellement étiquetées, on a à notre disposition:

- le corpus SemCor, qui comporte 230 000 mots étiquetés;
- le corpus eXtended WordNet, qui est simplement l'ensemble des définitions de sens de la version 1.7 de WordNet ayant été désambiguïsées. Ceci représente 560 000 mots étiquetés;
- le corpus de la tâche "English All Words" de la compétition Senseval-2, soit 2400 mots étiquetés;

Le corpus d'entraînement supervisé sera formé du corpus d'eXtended WordNet et de la majeur partie du corpus SemCor, soit 760 000 exemples. Le corpus de validation comporte la partie restante de SemCor, et le corpus de test sera simplement le corpus Senseval-2. Les étiquettes sémantiques sont celles de la version 1.7.1 de WordNet [7].

À titre de données non-étiquetées, on utilise le corpus Brown, qui comporte 480 000 mots à classe ouverte (noms, verbes, adjectifs et adverbes).

À titre de "pseudo-données", on va utiliser la hiérarchie de WordNet, en considérant l'ensemble des couples de sens (s_i, s_j) ayant le même parent. Ces couples sont alors traités comme s'ils avaient été observés dans un corpus, mais on devra calibrer le poids qu'ils auront dans l'apprentissage.

La procédure générale consiste donc, pour chaque observation "réelle" ou "pseudo" notée obs, à faire la mise à jour de la façon suivante:

$$P(t|obs) \rightarrow \text{pour réestimer } P(t)$$

$$P(s_i|t, obs) \rightarrow \text{pour réestimer } P(s|t)$$

Lorsque ces probabilités a posteriori sont calculées, et en supposant que chaque "observation" obs possède un poids $w(obs)$ (ou fréquence), les formules EM de réestimation sont simplement:

$$P(t) \leftarrow \frac{\sum_{obs} P(t|obs) * w(obs)}{\sum_{obs} w(obs)}$$

$$P(s|t) \leftarrow \frac{\sum_{obs} \sum_i P(s_i = s|t, obs) P(t|obs) * w(obs)}{\sum_{obs} n P(t|obs) * w(obs)}$$

L'algorithem EM itère sur les mises à jour ci-dessus, jusqu'à ce qu'un critère de convergence soit atteint (e.g. la performance en validation atteint son maximum) ou pour un nombre prédéfini d'itérations.

5 Calcul des probabilités a posteriori

Voici maintenant comment on calcule ces probabilités a posteriori.

1. Considérons premièrement le cas avec données totalement étiquetées:

$$\begin{aligned} P(t|l_1^n, s_1^n) &= P(t|s_1^n) \\ &= \frac{P(s_1^n|t)P(t)}{\sum_t P(s_1^n|t)P(t)} \\ &= \frac{P(t)\prod_{i=1}^n P(s_i|t)}{\sum_t P(t)\prod_{i=1}^n P(s_i|t)} \end{aligned}$$

$$P(S_i = s|t, l_1^n, s_1^n) = 1_{s_i=s}$$

   i.e. 1 si s_i est s, 0 sinon.

   Ces probabilités peuvent être calculées efficacement, en temp $O(Tn)$.

2. Deuxièmement, considérons le cas avec données partiellement étiquetées. Sans perte de généralité, on note $1...m < n$ les indices des sens observés:

$$\begin{aligned} P(t|l_1^n, s_1^m) &= P(t|s_1^m, l_{m+1}^n) \\ &= \frac{P(s_1^m, l_{m+1}^n|t)P(t)}{\sum_t P(s_1^m, l_{m+1}^n|t)P(t)} \\ &= \frac{P(t)\prod_{i=1}^m P(s_i|t)\prod_{i=m+1}^n \sum_{s_i} P(l_i|s_i)P(s_i|t)}{Z} \end{aligned}$$

   où Z est la somme du numérateur sur t

   pour $i \leq m$:
   $$P(S_i = s|t, l_1^n, s_1^m) = 1_{s_i=s}$$
   pour $i > m$:
   $$P(S_i = s|t, l_1^n, s_1^m) = P(S_i = s|t, l_i) = P(l_i|S_i = s)P(S_i = s|t)/Z$$

   où Z est la somme du numérateur sur les sens possibles s compatibles avec l_i.

   Ces calculs peuvent tous être exécutés en temps $O(TnM)$.

3. Troisièmement, considérons le cas avec données totalement non-étiquetées. Ce cas est trivial. Il suffit d'appliquer le cas avec données partiellement étiquetées, mais avec $m = 0$.

4. Quatrièmement, considérons le cas où on tente de lisser le modèle par la probabilité jointe de s_i et s_j à l'aide de pseudo-données du type $q(s_i, s_j)$. Dans les cas précédents, le poids de chaque observation était simplement 1. Ici, chaque observation sera une paire de valeur (s_i, s_j) associée à un poids $w(q(s_i, s_j))$. On calcule alors $P(t|s_i, s_j)$ et $P(S_i = s|t, s_i, s_j)$ de la façon suivante:

$$\begin{aligned} P(t|s_i, s_j) &= \frac{P(s_i, s_j|t)P(t)}{\sum_t P(s_i, s_j|t)P(t)} \\ &= \frac{P(s_i|t)P(s_j|t)P(t)}{\sum_t P(s_i|t)P(s_j|t)P(t)} \end{aligned}$$

   et

$$P(S_i = s|t, s_i, s_j) = 1_{s_i=s}$$

   et de même pour $S_j = s$.

6 Implémentation du modèle

La librairie PLearn offre une vaste gamme d'outils informatique dédiés à l'apprentissage machine et a été utilisée ici. On a donc créé un objet appelé `Graph_Model`, qui est dérivé de l'objet `Learner` de la librairie. En plus des spécifications de l'héritage, la nouvelle classe définit la méthode `compute_posteriors` qui calcule les probabilités a posteriori $P(s_i|l_1^n)$, et la méthode `em_learning` qui exécute la procédure d'apprentissage sur les différents types de données spécifiés par l'utilisateur.

Afin d'utiliser cette classe, l'utilisateur doit construire un fichier (appelé fichier de modèle) qui précise les paramètres du modèle graphique. Par exemple, il peut y indiquer le nombre de sujet T, le rayon de la fenêtre de mots (qui correspond à $\frac{n}{2}$) ou le nombre d'itérations de l'algorithme EM. C'est aussi par ce fichier qu'il spécifie s'il souhaite que le modèle apprenne sur des données non-étiquetées ou sur des pseudo-données. La figue 2 donne un exemple de fichier de modèle.

```
Graph_Model (

max_n_epochs = 100;
use_pos_reduction = 1;
save_best_model = 0;
n_window = 1;
n_topics = 10;
pseudo_data_learning = 0;
pseudo_data_weight = 0;
consider_all_senses = 0;
use_heuristic = 0;
p_si_cond_topic_factor = 0.5;

voc_path ="/u/larocheh/myUserExp/WSD/features/world4.voc";
synset_path ="/u/larocheh/myUserExp/WSD/features/world4.synsets";
ontology_path ="/u/larocheh/myUserExp/WSD/features/world4.ontology";
sense_key_path ="/u/larocheh/myUserExp/WSD/features/world4.sense_key";
)
```

Figure 2: Exemple de fichier de modèle

L'option `save_best_model` permet d'indiquer si on souhaite que les paramètres du meilleur modèle, i.e. son état lorsqu'il atteint ses meilleures performances en validation, soient enregistrés dans un fichier.

Le programme `useGraphModel` permet d'instancier la classe `Graph_Model` et d'utiliser le modèle. L'appel se fait comme suit:

```
useGraphModel model_file n_sup_train n_unsup_train n_valid model_name
```

où:

- `model_file` est le chemin d'accès du fichier de modèle;
- `n_sup_train` est le nombre d'exemples d'entraînement étiquetés à utiliser (jusqu'à concurrence de la taille du corpus d'entraînement);
- `n_unsup_train` est le nombre d'exemples non-étiquetés à utiliser (idem);
- `n_valid` est le nombre d'exemples en validation à utiliser (idem);
- `model_name` est le nom du modèle, qui servira de nom pour le fichier de modèle.

Le programme imprime alors à l'écran, d'une itération à l'autre, les statistiques liées à la classification et à la log-vraisemblance négative (voir section 8) en entraînement et en validation. La figure 3 montre un exemple de sortie.

Finalement, si l'utilisateur a spécifié dans le fichier de modèle qu'il souhaite enregistrer les meilleurs paramètres de celui-ci, alors le programme donne à la fin les résultats en test associés au meilleur modèle.

```
12:53:32> epoch 1
train classification/recall = 592484 / 734136, 80.7049% (592484)
n_seen_examples = 806409
train negative likelihood = 0.59007291471085
test classification/recall = 17781 / 24712, 71.9528973777922% (17781)
n_seen_examples = 24714
test negative likelihood = 1.05019452880552
----------------------------------------------------------------------------
EM learning
n seen examples in EM learing: 806409
12:57:30> epoch 2
train classification/recall = 592722 / 734136, 80.7373565660859% (238)
n_seen_examples = 806409
train negative likelihood = 0.590075666146437
test classification/recall = 17780 / 24712, 71.948850760764% (-1)
n_seen_examples = 24714
test negative likelihood = 1.0498522766525
----------------------------------------------------------------------------
EM learning
n seen examples in EM learing: 806409
13:01:48> epoch 3
train classification/recall = 592791 / 734136, 80.7467553695773% (69)
n_seen_examples = 806409
train negative likelihood = 0.590069028106246
test classification/recall = 17784 / 24712, 71.9650372288767% (4)
n_seen_examples = 24714
test negative likelihood = 1.04985210930595
----------------------------------------------------------------------------
                                   .
                                   .
                                   .
```

Figure 3: Exemple de sortie du programme

7 Bornes de la désambiguïsation

Avant de commencer à analyser le modèle proposé, il est important d'avoir une idée des performances limites de la tâche.

7.1 Borne inférieure

Comme borne inférieure, on a simplement implémenté deux algorithmes très simples, qui donneront des estimations de la borne inférieure à battre.

Le premier algorithme consiste simplement à choisir le sens le plus fréquent du mot cible. Plus formellement, on note cette décision:

$$d(s_i, l_i) = \underset{s_i \in sens(l_i)}{\operatorname{argmax}} \; P(s_i|l_i)$$

où $P(s_i|l_i)$ est estimée par simple comptage, sur le corpus d'entraînement utilisé par le modèle graphique. On obtient alors 72.3% de bonnes classifications sur le corpus de validation, ce qui est passablement élevé. Cette statistique met d'ailleurs en évidence une difficulté reconnue en désambiguïsation: plusieurs sens sont rarement utilisés pour un même mot.

Le second algorithme considéré utilise l'approche dite de Bayes naïf. Celle-ci consiste à supposer une dépendance seulement entre les mots de contexte et le sens du mot cible. Il n'y a donc pas de dépendance directe entre les mots de contexte et le mot cible. On peut alors calculer la probabilité a posteriori d'un sens comme suit:

$$P(s_i|l_1^n) = \frac{P(l_1^n|s)i)P(s_i)}{P(l_1^n)} = \frac{P(s_i)\prod_{j=1}^n P(l_j|s_i)}{P(l_1^n)}$$

On prend alors une décision comme suit:

$$d(s_i, l_1^n) = \underset{s_i \in sens(l_i)}{\operatorname{argmax}} \frac{P(s_i)\prod_{j=1}^n P(l_j|s_i)}{P(l_1^n)} = \underset{s_i \in sens(l_i)}{\operatorname{argmax}} \; P(s_i) \prod_{j=1}^{n} P(l_j|s_i)$$

Cependant, on remarque expérimentalement que trop d'influence est alors accordée aux mots de contexte $l_j, j \neq i$. On utilise alors le truc suivant, qui donne de meilleurs résultats:

$$d(s_i, l_1^n) = \underset{s_i \in sens(l_i)}{\operatorname{argmax}} \; P(s_i) \sqrt[n-1]{\prod_{j=1}^{n} P(l_j|s_i)}$$

Le mots cible l_i est toujours centré dans la fenêtre de longueur n. Les tables $P(l_j|s_i), j \neq i$ et $P(l_i|s_i)$ sont estimées par simple comptage sur le corpus d'entraînement du modèle graphique.

Les meilleurs résultats en validation sont obtenus avec $n = 3$, ce qui correspond à 73.2% de bonnes classifications. Là aussi, les performances sont plutôt élevées. D'ailleurs, lors de la compétition Senseval-2, pour la tâche "English All-Words", très peu d'applications ont réussi à faire mieux que l'algorithme de Bayes naïf.

7.2 Borne supérieure

Dans la littérature scientifique, on distingue deux types de bornes supérieures, soient l'accord entre étiqueteur ("inter-tagger agreement" ou ITA) et la reproductibilité ("replicability").

L'ITA est simplement la proportion des exemples à désambiguïser pour lesquels deux humains sont en accord sur l'étiquette sémantique à assigner. Les chiffres varient beaucoup cependant, allant de 57% [8] à 85% [9] (pour les noms). Cette mesure est cependant beaucoup moins précise, car moins encadrée. En effet, certaines études ont utilisé des étudiants plutôt que des experts, et la gestion des cas où les étiqueteurs sont eux-mêmes incertains sur un étiquette unique pour un mot cible est libre.

Une mesure beaucoup plus fiable est celle de la reproductibilité. Elle consiste en la proportion des exemples à désambiguïser pour lesquels deux équipes d'experts produisent le même étiquette. Un protocole bien défini, incluant une procédure d'arbitration lorsque l'accord n'est pas parfait à l'intérieur du groupe, doit être suivi. Les experts sont aussi encouragés à assigner plus d'une étiquette lorsqu'il y en a plusieurs raisonnables, ou même aucune lorsque toutes les étiquettes possibles ne sont pas adéquates. On obtient alors 95% d'accord, et c'est ce chiffre qui sera retenu.

8 Analyse des résultats

8.1 Apprentissage totalement et partiellement supervisé

Pour débuter, afin de trouver les valeurs optimales de n et T, on va utiliser la fonction de coût de la log-vraisemblance moyenne négative, ou plus formellement:

$$J(D,G) = (1/n) \sum_{(l_1^n, s_1^n) \in D} \sum_{i=1}^{n} -log(P(s_i | l_1^n))$$

où D est le corpus, qui peut être vu comme un ensemble de couples (l_1^n, s_1^n), et G est le modèle graphique. On observera donc la progression de la valeur prise par cette fonction sur le corpus d'entraînement et de validation, d'une itération de EM à l'autre.

Aussi, plutôt qu'utiliser directement la probabilité $P(s_i|t)$, on va faire un mélange avec la probabilité $P(s_i)$, i.e.:

$$P_m(s_i|t) = (1-\beta)P(s_i|t) + \beta P(s_i)$$

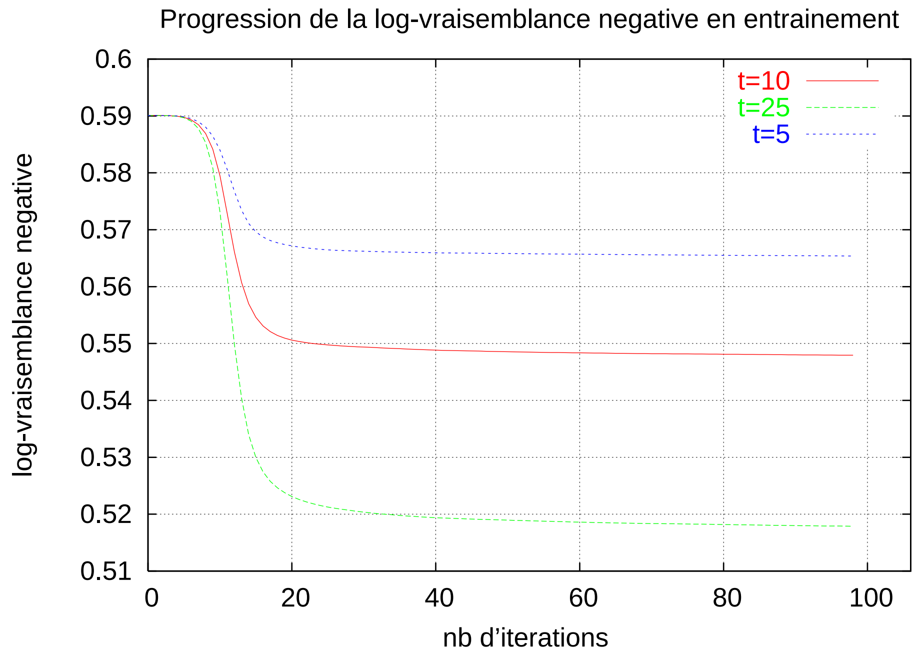
où β est calibré à l'aide d'un ensemble de validation. Ce changement est motivé par quelques essais préliminaires, qui ont montrés un problème flagrant de sur-apprentissage. Donc, après avoir observé les courbes de progression de la fonction de coût pour différentes valeurs de n, T et β, on obtient les valeurs optimales $n = 3$, $T = 10$ et $\beta = 0.5$.

Les figures 4(a) et 4(b) montrent les courbes de progression en entraînement et en validation pour différentes valeurs de T, lorsque $n = 3$ et $\beta = 0.5$ sont fixés [1].

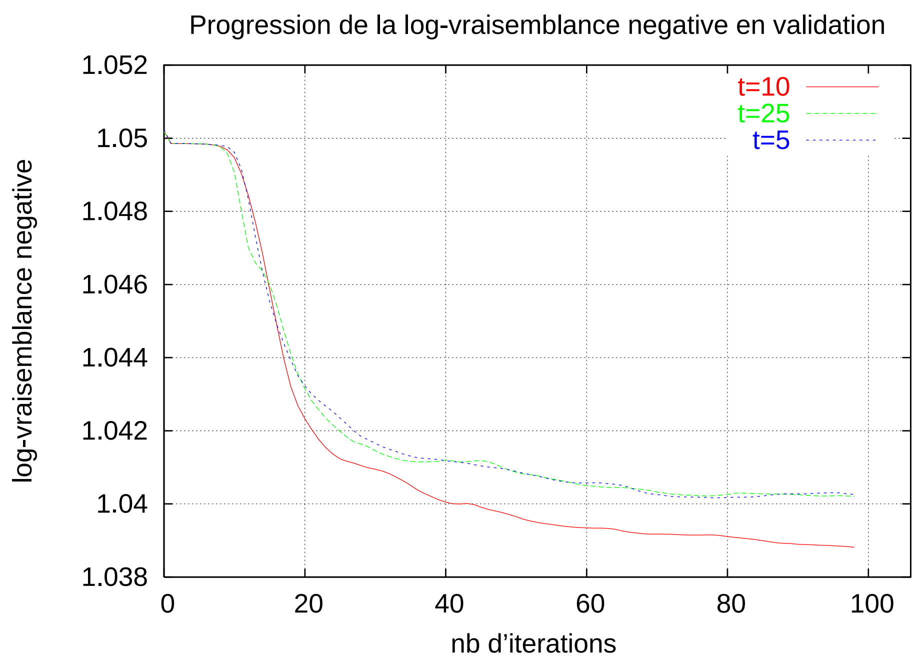
On y observe des comportements normaux, i.e. une amélioration continuelle en entraînement et en validation jusqu'à 80 itérations, où le modèle ne réussit plus à améliorer ses performances en validation (sauf peut-être pour $T = 10$, qui stagne un peu plus loin). Le phénomène de sur-apprentissage est très peu visible. On remarque aussi que la log-vraisemblance négative est beaucoup plus élevée en validation qu'en entraînement. Cette caractéristique reflète normalement un manque de données d'apprentissage par rapport à l'espace associé, les tables de probabilités étant alors beaucoup plus représentatives des données d'entraînement. En effet, le nombre de possibilités pour une fenêtre de 3 mots est extrêmement grand vu la taille du vocabulaire anglais qui est de plus de 100 000 mots. Aussi, l'amélioration en validation est très petite, allant de 1.05 à seulement 1.0385.

On peut alors essayer de désambiguïser avec la règle $\mathrm{argmax}_i\, P(s_i|l_1^n)$ (figures 5(a) et 5(b)). Après seulement 30 itérations en entraînement et un peu plus de 40 itérations en validation, ce qui correspond aux itérations où il y a le plus d'amélioration dans la

[1] Plusieurs configurations ont été tentées. On a commencé par trouver le β optimal avec n et T fixés, puis toutes les combinaisons de $n \in \{1, 2, 3\}$ et $T \in \{1, 5, 10, 25, 75\}$ ont été essayées. On présente ici les courbes les plus pertinentes.



(a)



(b)

Figure 4: Progression de la log-vraisemblance négative moyenne

vraisemblance des données, le nombre de bonnes classifications cessent d'augmenter. Malheureusement, le nombre de bonnes classifications en validation n'atteint pas les performances de l'algorithme du sens le plus fréquent (noté "baseline"). L'algorithme de Bayes naïf avec atténuation du context, pour une fenêtre de $n = 3$ mots, fait encore mieux, à plus de 73% de bonnes classifications.

Comme cité plus haut, le manque de données par rapport à la taille de l'espace est certainement à blâmer. Le problème est que pour un mot à désambiguïser, il est fort probable qu'aucun des mots du contexte n'ait déjà été vu dans un autre contexte appris de ce même mot cible. En fait, pour une fenêtre de 3 mots de diamètre, seulement 55.9% des mots cibles du corpus de validation ont dans leur contexte un mot qui a déjà été vu dans le contexte du même mot, dans l'ensemble d'entraînement. En d'autres mots, dans près de la moitié des cas, on a affaire à un contexte totalement différent de ceux rencontrés en entraînement, ce qui est beaucoup. Si on s'intéresse à cette même mesure, mais pour les sens, on obtient 21.5%, soit encore très peu. Lorsqu'on élargit la fenêtre à $n = 5$, ces pourcentages augmentent, mais les fenêtres observées sont alors encore plus propices à contenir des mots qui n'aident en rien à la désambiguïsation. C'est pourquoi la taille de la fenêtre optimale est plutôt petite.

Lorsqu'on utilise l'heuristique décrite plus haut avec $k = 2$, pour sélectionner un sous-ensemble des séquences s_1^n possibles et approximer $\mathrm{argmax}_i\, P(s_1^n|l_1^n)$, les performances en validation sont encore moins bonnes (figures 6).

Par contre, ceci n'est pas très surprenant puisqu'il semble que l'information contextuelle contenue dans le corpus d'entraînement ne soit pas suffisante. On aimerait donc ajouter des données d'apprentissage, afin de pallier ce problème.

8.2 Apprentissage non-supervisé

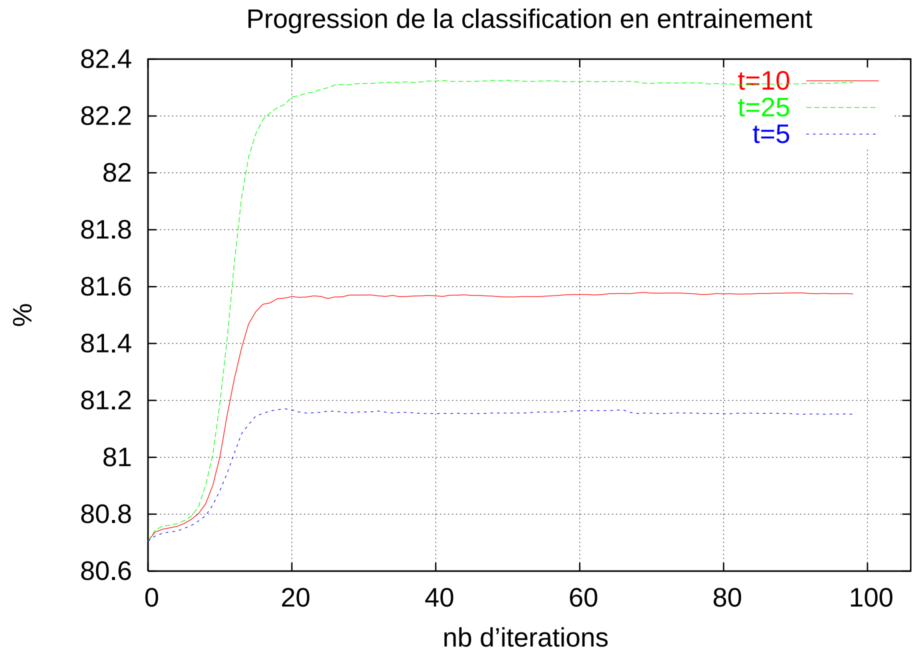
Malheureusement, l'algorithme ne réussit pas à profiter des données non-étiquetées (figures 7(a) et 7(b)). Après 75 itérations, lorsque la procédure d'apprentissage commence à utiliser les exemples non-étiquetés, les performances empirent, quelque soit le poids (noté uf) donné à ces exemples. Encore une fois, le manque de données totalement ou partiellement étiquetées est probablement en cause, car une bonne base d'apprentissage supervisé est souvent nécessaire à de bons résultats en non-supervisé.

8.3 Apprentissage sur pseudo-données

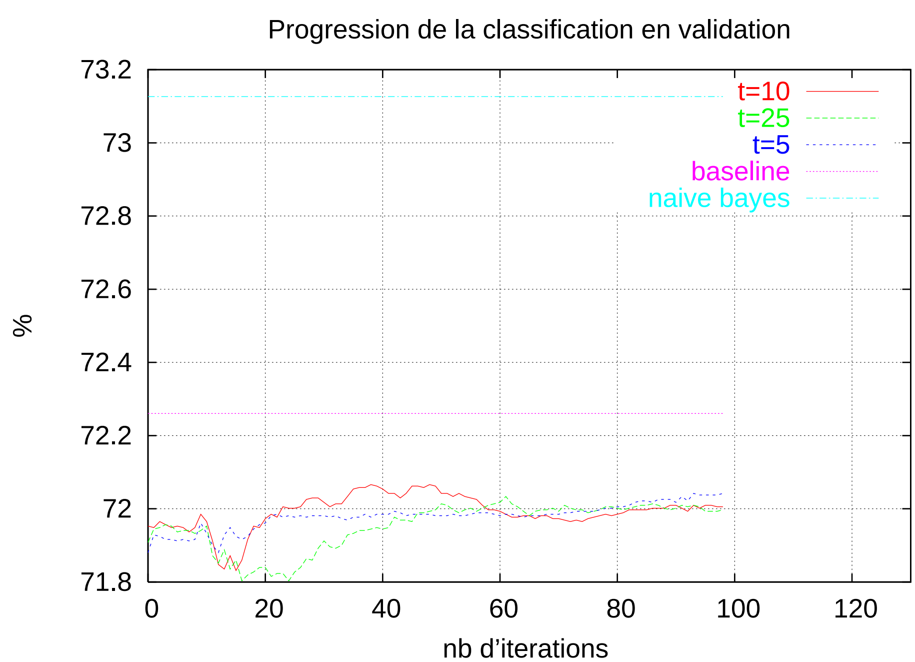
L'apprentissage sur WordNet permet de faire des gains au niveau de la log-vraisemblance négative (figure 8(b)), lorsqu'on donne un poids $w(q(s_i, s_j)) = 0.1$ aux observations. Cependant, on ne retrouve pas les mêmes gains avec la classification (figure 8(a)).

Ceci est plutôt surprenant, puisqu'on s'attend à ce que l'arborescence WordNet soit une source d'information fiable. Par contre, le problème ne réside pas vraiment là. En effet, les distributions de la distance minimale (dans la hiérarchie WordNet) du sens cible correct (figure 9(a)) et des autres sens possibles pour le mot cible (figure 9(b)) avec les sens corrects des autres mots de la fenêtre de longueur $n = 3$ sont plutôt indistinguables. Les couples (s_i, s_j) de sens partageant le même parent n'offrent donc pas d'information utile à la désambiguïsation, et considérer d'autres couples (s_i, s_j) plus éloignés ne semble pas être une voie prometteuse.

On retiendra tout de même l'apprentissage sur pseudo-données, car la différence en classification est très petite, et probablement pas significative.



(a)



(b)

Figure 5: Progression de la classification

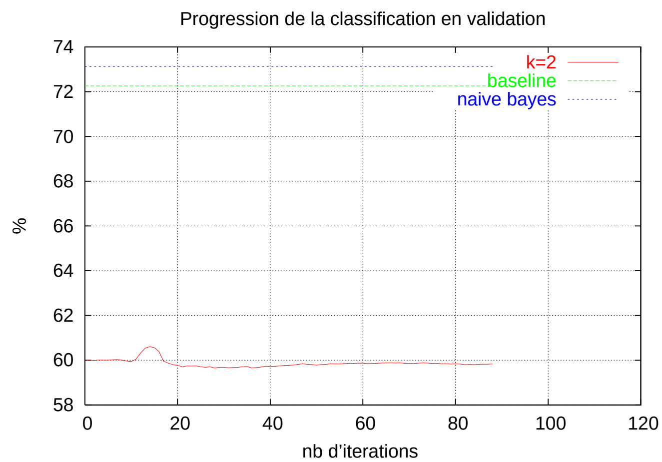


Figure 6: Progression de la classification en test avec heuristique

Table 1: Résultats finaux

Algorithme	**% de bonnes classifications**					**McNemar**
	Tout	**Noms**	**Verbes**	**Adjectifs**	**Adverbes**	
Modèle graphique	64.6	72.8	45.6	70.1	81.4	-
Sens le plus fréquent	64.4	72.6	45.3	70.1	81.7	0.125
Bayes naïf	64.7	71.9	47.0	70.1	82.4	0

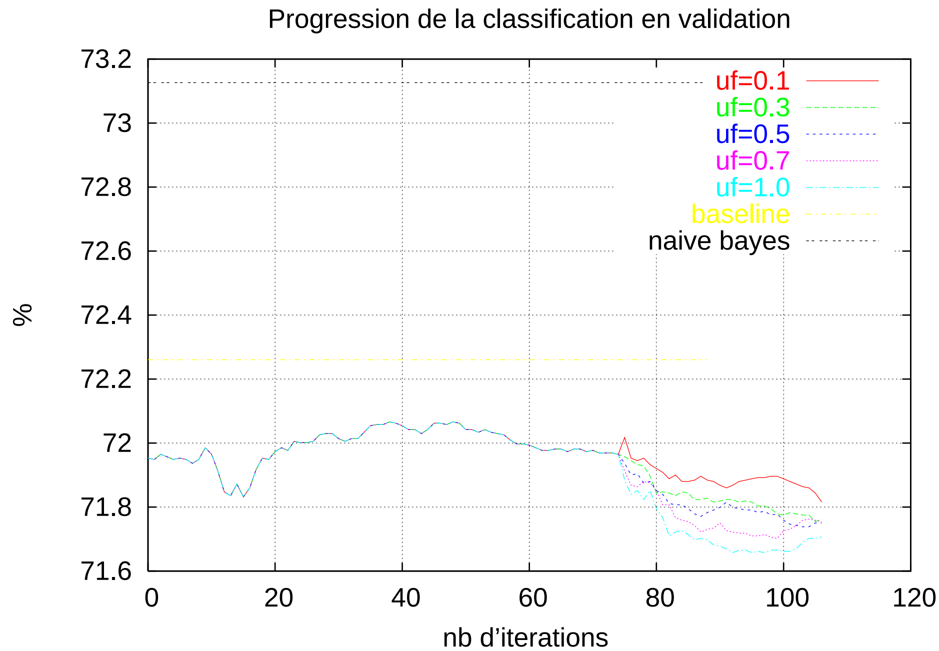
8.4 Résultats finaux

Suite à la calibration des paramètres sur le corpus de validation, on choisit le modèle avec apprentissage sur données totalement/partiellement étiquetées et sur pseudo-données, où $n = 3$, $T = 10$, $\beta = 0.5$ et $w(q(s_i, s_j)) = 0.1$. Le sens est choisi selon la règle $\text{argmax}_i\, P(s_i|l_1^n)$. Les résultats[2] sur le corpus de test (corpus de Senseval-2) sont présentés dans la table 1.

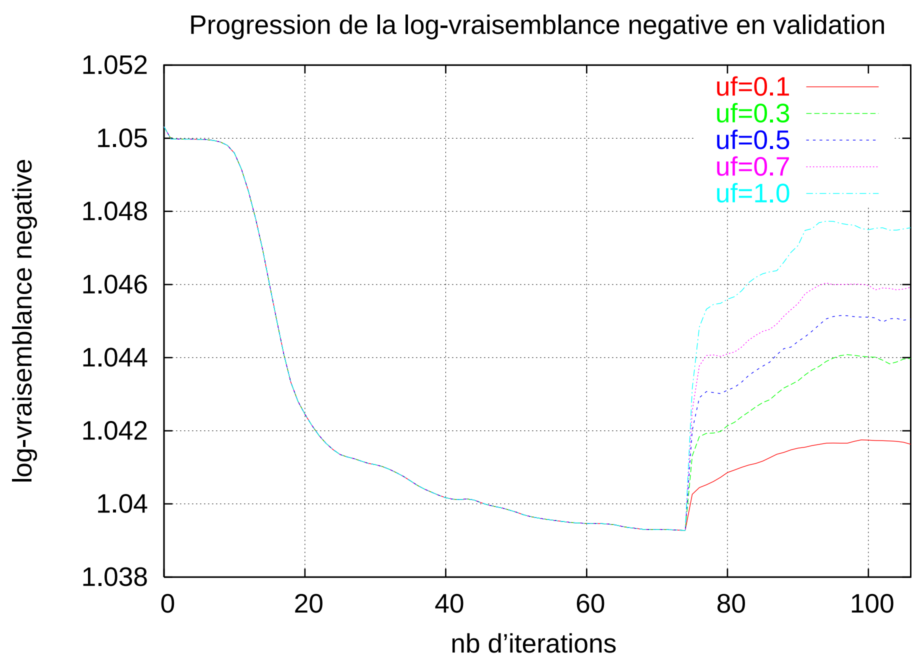
La colonne McNemar donne la valeur de la statistique du test du même nom qui compare le modèle graphique aux algorithmes du sens le plus fréquent et de Bayes naïf. Si cette valeur est plus petite que $\chi^2_{1;0.95} = 3.841$, alors les algorithmes ont statistiquement une performance équivalente, ce qui est le cas ici. Ce test est conseillé [10] lorsqu'un des algorithmes à comparer est coûteux en temps d'apprentissage, ce qui est le cas pour le modèle graphique. Celui-ci peut prendre plus de 10 heures avant d'atteindre un minimum en validation avec la log-vraisemblance négative.

Dans le corpus de test, 38.7% des mots cibles ont dans leur contexte un mot qui a déjà été vu dans le contexte du même mot dans l'ensemble d'entraînement, ce qui est beaucoup moins que pour l'ensemble de validation. Pour les sens, c'est encore pire, avec 9.6%. Ceci explique pourquoi les performances sont beaucoup plus faibles sur le corpus de test que sur le corpus de validation.

[2] On a ici supposé qu'on avait accès aux classes syntaxiques (POS) correctes, afin que les performances ne dépendent pas de la qualité de l'extraction de celles-ci.

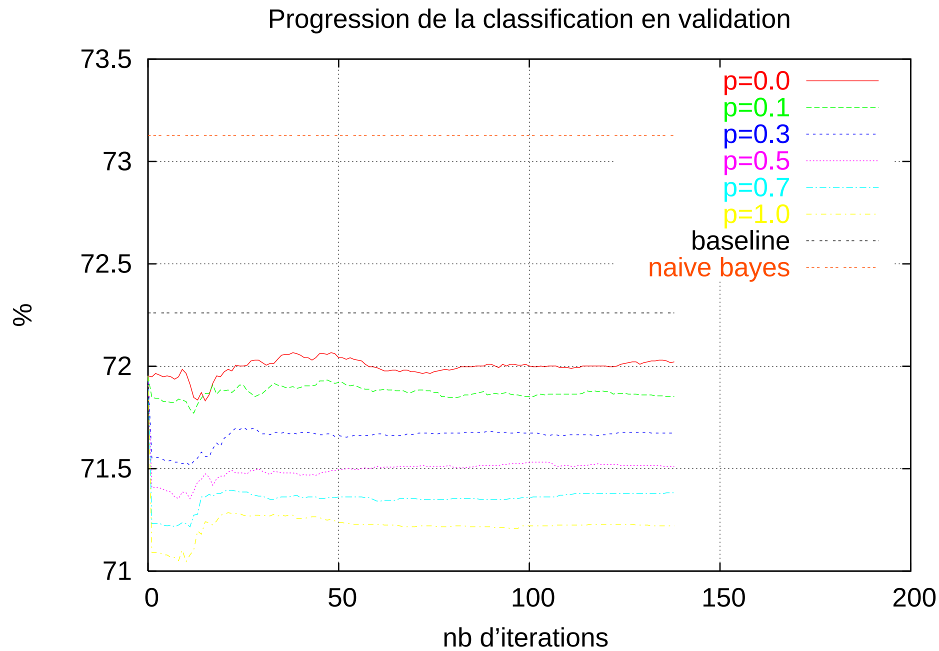


(a)

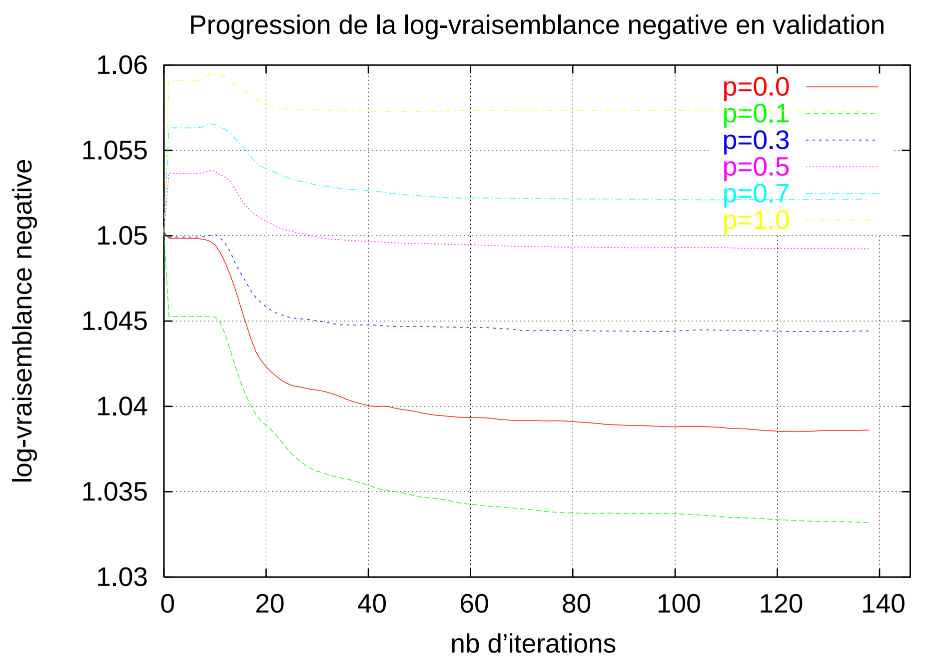


(b)

Figure 7: Progression en validation avec données non-étiquetées

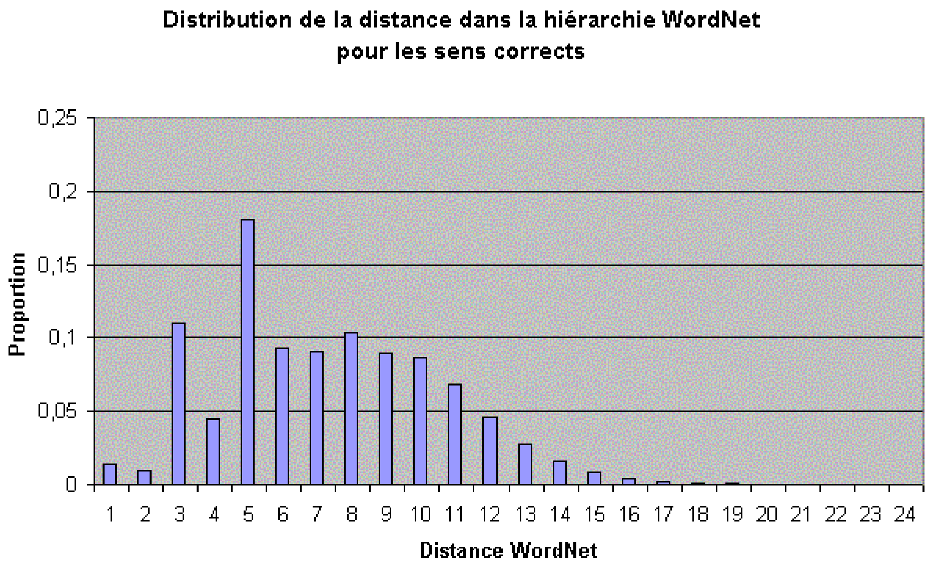


(a)

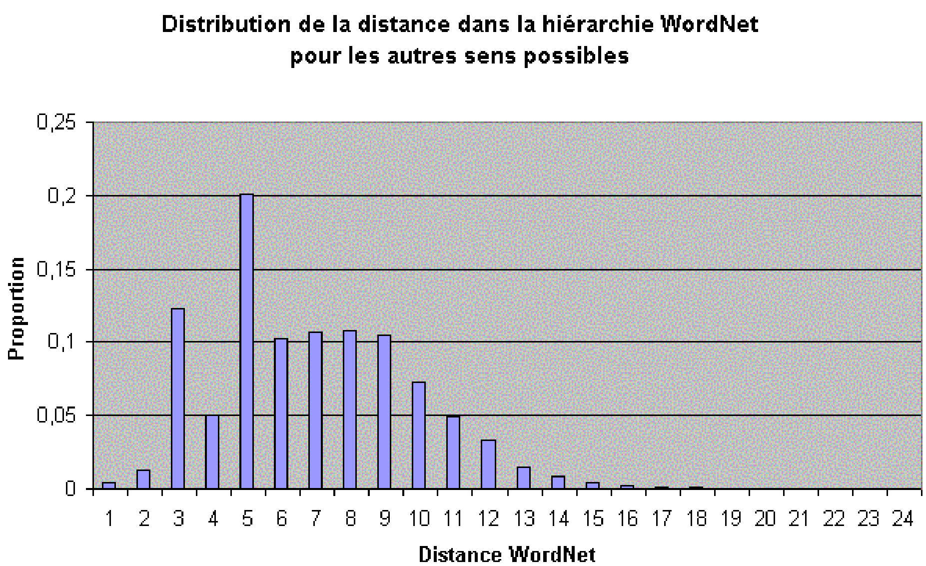


(b)

Figure 8: Progression en validation avec pseudo-données



(a)



(b)

Figure 9: Distributions des distances WordNet pour les sens cibles et pour les autres sens possibles

9 Conclusion

Le modèle graphique proposé ne réussit pas à saisir une relation sémantique sophistiquée et complète. Cependant, sur un corpus largement utilisé pour comparer les performances de désambiguïsation, i.e. le corpus de Senseval-2, sa performance est statistiquement équivalente à celle de l'algorithme de Bayes naïf et du sens le plus fréquent. On atteint donc la borne inférieure qu'on s'était fixée. De plus, il est à noter que peu d'algorithmes participant à la compétition Senseval-2 ont réussi à faire aussi bien que l'algorithme de Bayes, ou même que l'algorithme du sens le plus fréquent.

Tout semble indiquer que le manque de données par rapport à la taille l'espace échantillonal, un problème reconnu en désambiguïsation, en soit la cause.

En effet, très peu de corpus étiquetés sont disponibles, et on estime à 3.2 millions le nombre d'exemples étiquetés nécessaire à l'atteinte de performances satisfaisantes (voir [11]). Cependant, il existe un type de corpus très abondant et potentiellement utile en désambiguïsation (e.g. voir [12]), qui pourrait permettre de vaincre ce problème: les bi-textes.

Les bi-textes, qui sont formés de deux corpus traduits et alignés, forment une base de données imposante de mots sur lesquels une désambiguïsation implicite liée à la traduction a été appliquée manuellement. Le corpus Hansard est un des plus utilisés, avec près de 235 millions de mots. Ce type de données peut facilement être intégré à une procédure d'apprentissage de type EM pour un modèle graphique.

Effectivement, lorsque deux mots alignés ne partagent qu'un seul sens, alors on a un exemple étiqueté qui peut être directement utilisé par EM. Si les deux mots ont plus d'un sens s en commun, on peut simplement pondérer l'exemple avec sa probabilité relative a posteriori par rapport aux autres sens en commun en posant:

$$w(l_i, s) = \frac{P(s|l_i)}{\sum_{s \in S^*} P(s|l_i)}$$

où S^* est l'ensemble des sens en commun. Je suggère donc que cette voie soit explorée.

References

[1] Yorick Wilks and Mark Stevenson. The grammar of sense: Is word sense tagging much more than part-of-speech tagging? Technical report, University of Sheffield, 1996.

[2] Nancy Ide and Jean Veronis. Introduction to the special issue on word sense disambiguation: The state of the art. *Computational Linguistics*, Vol. 24(No. 1):1–40, 1998.

[3] Michael E. Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *SIGDOC Conference*, Toronto, Canada, 1980.

[4] Jean Veronis and Nancy Ide. Word sense disambiguation with very large neural networks extracted from machine readable dictionaries. In *COLING'90*, 1990.

[5] David Yarowsky. Decision lists for lexical ambiguity resolution: Application to accent restoration in spanish and french. In *Meeting of the Association for Computational Linguistics*, pages 88–95, 1994.

[6] Rada Mihalcea. Instance based learning with automatic feature selection applied to word. In *Proceedings of the 6th Conference on Natural Language Learning*, 2002.

[7] Christine Fellbaum. *WordNet: An Electronic Lexical Database and Some of its Application*. MIT Press, 1996.

[8] Hwee Tou Ng and Hian Beng Lee. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 40–47, San Francisco, 1996. Morgan Kaufmann Publishers.

[9] Adam Kilgarriff. English lexical sample task description. In *Proceedings of Senseval-2*. ACL workshop, 2002.

[10] Thomas G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[11] Hwee Tou Ng. Getting serious about word sense disambiguation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, pages 1–7, Washington, 1997.

[12] Christopher Kermorvant Yoshua Bengio. Extracting hidden sense probabilities from bitexts. Technical Report 1231, Université de Montréal, 2003.