Méthodes Empiriques en Linguistique Informatique

Paola Merlo

University of Geneva

année académique 2005-2006

Rappel

- Un de principaux défis en TAL consiste à fournir aux ordinateurs les connaissances linguistiques nécessaires pour effectuer avec succès des tâches langagières.
- Solution A la place d'un expert qui fournit à l'ordinateur des informations sur le langage, le programme apprend lui-même à partir des données textuelles.

Rappel

- Un de principaux défis en TAL consiste à fournir aux ordinateurs les connaissances linguistiques nécessaires pour effectuer avec succès des tâches langagières.
- Solution A la place d'un expert qui fournit à l'ordinateur des informations sur le langage, le programme apprend lui-même à partir des données textuelles.

Apprentissage : Définition

On dit qu'on programme informatique apprend à partir d'une expérience empirique E par rapport à la tâche T et par rapport à une mèsure de performance P, si sa performance P à la tâche T s'améliore à la suite de E.

Exemple

- Tâche : classer des verbes anglais dans des classes prédéfinies.
- Mésure de performance P : % des verbes classés correctement par rapport à une classification définie par des experts (gold standard).
- Expérience d'entraînement E : base de données de couples de verbes (et leurs propriétes) et classe correcte.

Apprentissage : Définition

On dit qu'on programme informatique apprend à partir d'une expérience empirique E par rapport à la tâche T et par rapport à une mèsure de performance P, si sa performance P à la tâche T s'améliore à la suite de E.

Exemple

- Tâche : classer des verbes anglais dans des classes prédéfinies.
- Mésure de performance P : % des verbes classés correctement par rapport à une classification définie par des experts (gold standard).
- Expérience d'entraînement E : base de données de couples de verbes (et leurs propriétes) et classe correcte.

Apprentissage : Définition

On dit qu'on programme informatique apprend à partir d'une expérience empirique E par rapport à la tâche T et par rapport à une mèsure de performance P, si sa performance P à la tâche T s'améliore à la suite de E.

Exemple

- Tâche : classer des verbes anglais dans des classes prédéfinies.
- Mésure de performance P : % des verbes classés correctement par rapport à une classification définie par des experts (gold standard).
- Expérience d'entraînement E : base de données de couples de verbes (et leurs propriétes) et classe correcte.

Apprentissage par classification

La tâche la plus étudiée en apprentissage automatique (machine learning) consiste à inférer une fonction classant des exemples representés comme vecteurs de traits distinctifs dans une catégorie parmi un ensemble fini de catégories données.

Exemple

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Tous les transparents qui suivent sont mis à disposition par le professeur Tom Mitchell de Carnegie Mellon University et il sont à utiliser avec le livre
Tom Mitchell,*Machine Learning*, McGraw Hill, 1997.

Apprentissage Bayésian

Les méthodes bayésiennes fournissent des algorithmes très performants, mais très simples, qui combinent les connaissance préalables (probabilité a priori) avec les données observées. L'algorithme plus simple s'appelle Naive Bayes.

Théorème de Bayes

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- $P(h)$ = probabilité a priori de l'hypothèse h
- $P(D)$ = probabilité a priori des données d'entraînement D
- $P(h|D)$ = probabilité de h sachant D
- $P(D|h)$ = probabilité de D sachant h

Théorème de Bayes

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- $P(h)$ = probabilité a priori de l'hypothèse h
- $P(D)$ = probabilité a priori des données d'entraînement D
- $P(h|D)$ = probabilité de h sachant D
- $P(D|h)$ = probabilité de D sachant h

Théorème de Bayes

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- $P(h)$ = probabilité a priori de l'hypothèse h
- $P(D)$ = probabilité a priori des données d'entraînement D
- $P(h|D)$ = probabilité de h sachant D
- $P(D|h)$ = probabilité de D sachant h

Théorème de Bayes

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- $P(h)$ = probabilité a priori de l'hypothèse h
- $P(D)$ = probabilité a priori des données d'entraînement D
- $P(h|D)$ = probabilité de h sachant D
- $P(D|h)$ = probabilité de D sachant h

Comment choisir une hypothèse

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

On veut l'hypothèse la plus probable sachant les données d'entraînement
Hypothèse maximum a posteriori (*Maximum a posteriori* hypothesis) h_{MAP} :

$$\begin{aligned} h_{MAP} &= \arg\max_{h \in H} P(h|D) \\ &= \arg\max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\ &= \arg\max_{h \in H} P(D|h)P(h) \end{aligned}$$

Si les hypothèses ont toutes la même probabilité $P(h_i) = P(h_j)$, on peut simplifier encore plus et choisir l'hypothèse avec la vraisamblance maximale (*Maximum likelihood* (ML) hypothesis). $P(D|h)$ s'appelle la vraisamblance des données sachant l'hypothèse.

$$h_{ML} = \arg\max_{h_i \in H} P(D|h_i)$$

Bayes Theorem

Does patient have cancer or not ?

A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% of the cases in which the disease is actually present, and a correct negative result in only 97% of the cases in which the disease is not present. Furthermore, .008 of the entire population have this cancer.

$$P(cancer) = \qquad P(\neg cancer) =$$
$$P(+|cancer) = \qquad P(-|cancer) =$$
$$P(+|\neg cancer) = \qquad P(-|\neg cancer) =$$

Naive Bayes Classifier

Les classifieurs Naive Bayes sont, avec les arbres à decision et les reseaux neuronaux, parmi les algorithmes d'apprentissage les plus pratiques et les plus performants.
On les utilise lorsque

- Les données d'entraînement sont nombreuses
- Les attributs qui decrivent la classification sont indépendants étant donné la classe

Applications appropriées :

- Diagnostic
- Classification de documents textuels

Naive Bayes Classifier

Les classifieurs Naive Bayes sont, avec les arbres à decision et les reseaux neuronaux, parmi les algorithmes d'apprentissage les plus pratiques et les plus performants.
On les utilise lorsque

- Les données d'entraînement sont nombreuses
- Les attributs qui decrivent la classification sont indépendants étant donné la classe

Applications appropriées :

- Diagnostic
- Classification de documents textuels

Naive Bayes Classifier

Les classifieurs Naive Bayes sont, avec les arbres à decision et les reseaux neuronaux, parmi les algorithmes d'apprentissage les plus pratiques et les plus performants.
On les utilise lorsque

- Les données d'entraînement sont nombreuses
- Les attributs qui decrivent la classification sont indépendants étant donné la classe

Applications appropriées :

- Diagnostic
- Classification de documents textuels

Naive Bayes Classifier

Les classifieurs Naive Bayes sont, avec les arbres à decision et les reseaux neuronaux, parmi les algorithmes d'apprentissage les plus pratiques et les plus performants.
On les utilise lorsque

- Les données d'entraînement sont nombreuses
- Les attributs qui decrivent la classification sont indépendants étant donné la classe

Applications appropriées :

- Diagnostic
- Classification de documents textuels

Naive Bayes Classifier

Soit la fonction cible $f : X \rightarrow V$, où chaque instance x est décrite par les attributs $\langle a_1, a_2 \ldots a_n \rangle$.
La valeur la plus probable $f(x)$ est :

$$\begin{aligned} v_{MAP} &= \underset{v_j \in V}{\operatorname{argmax}} P(v_j | a_1, a_2 \ldots a_n) \\ v_{MAP} &= \underset{v_j \in V}{\operatorname{argmax}} \frac{P(a_1, a_2 \ldots a_n | v_j) P(v_j)}{P(a_1, a_2 \ldots a_n)} \\ &= \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2 \ldots a_n | v_j) P(v_j) \end{aligned}$$

Hypothèse Naive Bayes :

$$P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$$

cela donne

Naive Bayes classifieur : $v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j)$

Algorithme Naive Bayes

Naive_Bayes_Learn(*examples*)

Pour chaque valeur cible v_j

$\hat{P}(v_j) \leftarrow$ estimation de $P(v_j)$

Pour chaque valeur a_i de chaque attribut a

$\hat{P}(a_i|v_j) \leftarrow$ estimation de $P(a_i|v_j)$

Classify_New_Instance(x)

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j)$$

Algorithme Naive Bayes

Naive_Bayes_Learn(*examples*)

Pour chaque valeur cible v_j

$\hat{P}(v_j) \leftarrow$ estimation de $P(v_j)$

Pour chaque valeur a_i de chaque attribut a

$\hat{P}(a_i|v_j) \leftarrow$ estimation de $P(a_i|v_j)$

Classify_New_Instance(x)

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j)$$

Algorithme Naive Bayes

Naive_Bayes_Learn(*examples*)

Pour chaque valeur cible v_j

$\hat{P}(v_j) \leftarrow$ estimation de $P(v_j)$

Pour chaque valeur a_i de chaque attribut a

$\hat{P}(a_i|v_j) \leftarrow$ estimation de $P(a_i|v_j)$

Classify_New_Instance(x)

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j)$$

Algorithme Naive Bayes

Naive_Bayes_Learn(*examples*)

Pour chaque valeur cible v_j

$\hat{P}(v_j) \leftarrow$ estimation de $P(v_j)$

Pour chaque valeur a_i de chaque attribut a

$\hat{P}(a_i|v_j) \leftarrow$ estimation de $P(a_i|v_j)$

Classify_New_Instance(x)

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j)$$

Play Tennis ?

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Naive Bayes : Exemple

Considerons *PlayTennis* à nouveau, et la nouvelle instance

$$\langle Outlk = sun, Temp = cool, Humid = high, Wind = strong \rangle$$

Nous voulons calculer

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i|v_j)$$

$$P(y)\ P(sun|y)\ P(cool|y)\ P(high|y)\ P(strong|y) = .005$$

$$P(n)\ P(sun|n)\ P(cool|n)\ P(high|n)\ P(strong|n) = .021$$

$$\rightarrow v_{NB} = n$$

Naive Bayes : Détails

1. L'hypothèse d'indépendance conditionnelle souvent n'est pas vérifiée

$$P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$$

   - ...mais l'algorithme marche également très bien.

Naive Bayes : Détails

1. L'hypothèse d'indépendance conditionnelle souvent n'est pas vérifiée

$$P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$$

   - ...mais l'algorithme marche également très bien.

Naive Bayes : Détails

2. Si aucun des exemples d'entraînement avec valeur cible v_j a valeur a_i pour l'attribut a, Alors

$$\hat{P}(a_i|v_j) = 0, \text{ et...}$$

$$\hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = 0$$

La solution typique à ce problème c'est l'estimation bayesienne pour $\hat{P}(a_i|v_j)$

$$\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n + m}$$

où

- n est le nombre d'exemples d'entraînement pour lesquelles $v = v_j$,
- n_c le nombre d'exemple pour lesquelles $v = v_j$ et $a = a_i$
- p est l'estimation a priori $\hat{P}(a_i|v_j)$
- m est la ponderation donnée à la distribution a priori (c-à-d le nombre d'exemples virtuels)

Naive Bayes : Détails

2. Si aucun des exemples d'entraînement avec valeur cible v_j a valeur a_i pour l'attribut a, Alors

$$\hat{P}(a_i|v_j) = 0, \text{ et...}$$

$$\hat{P}(v_j)\prod_i \hat{P}(a_i|v_j) = 0$$

La solution typique à ce problème c'est l'estimation bayesienne pour $\hat{P}(a_i|v_j)$

$$\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n + m}$$

où

- n est le nombre d'exemples d'entraînement pour lesquelles $v = v_j$,
- n_c le nombre d'exemple pour lesquelles $v = v_j$ et $a = a_i$
- p est l'estimation a priori $\hat{P}(a_i|v_j)$
- m est la ponderation donnée à la distribution a priori (c-à-d le nombre d'exemples virtuels)

Naive Bayes : Détails

2. Si aucun des exemples d'entraînement avec valeur cible v_j a valeur a_i pour l'attribut a, Alors

$$\hat{P}(a_i|v_j) = 0, \text{ et...}$$

$$\hat{P}(v_j)\prod_i \hat{P}(a_i|v_j) = 0$$

La solution typique à ce problème c'est l'estimation bayesienne pour $\hat{P}(a_i|v_j)$

$$\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n + m}$$

où

- n est le nombre d'exemples d'entraînement pour lesquelles $v = v_j$,
- n_c le nombre d'exemple pour lesquelles $v = v_j$ et $a = a_i$
- p est l'estimation a priori $\hat{P}(a_i|v_j)$
- m est la ponderation donnée à la distribution a priori (c-à-d le nombre d'exemples virtuels)

Naive Bayes : Détails

2. Si aucun des exemples d'entraînement avec valeur cible v_j a valeur a_i pour l'attribut a, Alors

$$\hat{P}(a_i|v_j) = 0, \text{ et...}$$

$$\hat{P}(v_j)\prod_i \hat{P}(a_i|v_j) = 0$$

La solution typique à ce problème c'est l'estimation bayesienne pour $\hat{P}(a_i|v_j)$

$$\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n + m}$$

où

- n est le nombre d'exemples d'entraînement pour lesquelles $v = v_j$,
- n_c le nombre d'exemple pour lesquelles $v = v_j$ et $a = a_i$
- p est l'estimation a priori $\hat{P}(a_i|v_j)$
- m est la ponderation donnée à la distribution a priori (c-à-d le nombre d'exemples virtuels)

Naive Bayes : Détails

2. Si aucun des exemples d'entraînement avec valeur cible v_j a valeur a_i pour l'attribut a, Alors

$$\hat{P}(a_i|v_j) = 0, \text{ et...}$$

$$\hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = 0$$

La solution typique à ce problème c'est l'estimation bayesienne pour $\hat{P}(a_i|v_j)$

$$\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n + m}$$

où

- n est le nombre d'exemples d'entraînement pour lesquelles $v = v_j$,
- n_c le nombre d'exemple pour lesquelles $v = v_j$ et $a = a_i$
- p est l'estimation a priori $\hat{P}(a_i|v_j)$
- m est la ponderation donnée à la distribution a priori (c-à-d le nombre d'exemples virtuels)

Apprendre à classer du texte

Pourquoi ?

- Apprendre quels textes sont intéressants
- Apprendre à classer les pages web par sujet

Naive Bayes est parmi les algorithmes les plus performants
Quel sont les attributs pour representer de données textuels ?

Apprendre à classer du texte

Pourquoi ?

- Apprendre quels textes sont intéressants
- Apprendre à classer les pages web par sujet

Naive Bayes est parmi les algorithmes les plus performants

Quel sont les attributs pour representer de données textuels ?

Apprendre à classer du texte

Concepte cible *Interessant*? : *Document* $\rightarrow \{+, -\}$

1. On represente un document par un vecteur de mots
   - Un attribut pour chaque position de mot dans le document
2. Apprentissage : Sur la base des exemples d'entraînement, on estime
   - $P(+)$ $P(-)$
   - $P(doc|+)$ $P(doc|-)$

Hypothèse d'indépendance conditionnelle Naive Bayes

$$P(doc|v_j) = \prod_{i=1}^{length(doc)} P(a_i = w_k|v_j)$$

où $P(a_i = w_k|v_j)$ est la probabilité que le mot en position i est w_k, sachant v_j
Une autre hypothèse : $P(a_i = w_k|v_j) = P(a_m = w_k|v_j), \forall i, m$

Apprendre à classer du texte

Concepte cible *Interessant*? : *Document* $\rightarrow \{+, -\}$

1. On represente un document par un vecteur de mots
   - Un attribut pour chaque position de mot dans le document
2. Apprentissage : Sur la base des exemples d'entraînement, on estime
   - $P(+)$ $P(-)$
   - $P(doc|+)$ $P(doc|-)$

Hypothèse d'indépendance conditionnelle Naive Bayes

$$P(doc|v_j) = \prod_{i=1}^{length(doc)} P(a_i = w_k|v_j)$$

où $P(a_i = w_k|v_j)$ est la probabilité que le mot en position i est w_k, sachant v_j
Une autre hypothèse : $P(a_i = w_k|v_j) = P(a_m = w_k|v_j), \forall i, m$

Apprendre à classer du texte

Concepte cible *Interessant*? : $Document \rightarrow \{+, -\}$

1. On represente un document par un vecteur de mots
   - Un attribut pour chaque position de mot dans le document
2. Apprentissage : Sur la base des exemples d'entraînement, on estime
   - $P(+)$ $P(-)$
   - $P(doc|+)$ $P(doc|-)$

Hypothèse d'indépendance conditionnelle Naive Bayes

$$P(doc|v_j) = \prod_{i=1}^{length(doc)} P(a_i = w_k|v_j)$$

où $P(a_i = w_k|v_j)$ est la probabilité que le mot en position i est w_k, sachant v_j
Une autre hypothèse : $P(a_i = w_k|v_j) = P(a_m = w_k|v_j), \forall i, m$

Apprendre à classer du texte

Concepte cible *Interessant*? : *Document* $\rightarrow \{+, -\}$

1. On represente un document par un vecteur de mots
   - Un attribut pour chaque position de mot dans le document
2. Apprentissage : Sur la base des exemples d'entraînement, on estime
   - $P(+) \quad P(-)$
   - $P(doc|+) \quad P(doc|-)$

Hypothèse d'indépendance conditionnelle Naive Bayes

$$P(doc|v_j) = \prod_{i=1}^{length(doc)} P(a_i = w_k|v_j)$$

où $P(a_i = w_k|v_j)$ est la probabilité que le mot en position i est w_k, sachant v_j
Une autre hypothèse : $P(a_i = w_k|v_j) = P(a_m = w_k|v_j), \forall i, m$

Apprendre à classer du texte

Concepte cible *Interessant*? : *Document* $\rightarrow \{+, -\}$

1. On represente un document par un vecteur de mots
   - Un attribut pour chaque position de mot dans le document
2. Apprentissage : Sur la base des exemples d'entraînement, on estime
   - $P(+) \quad P(-)$
   - $P(doc|+) \quad P(doc|-)$

Hypothèse d'indépendance conditionnelle Naive Bayes

$$P(doc|v_j) = \prod_{i=1}^{length(doc)} P(a_i = w_k|v_j)$$

où $P(a_i = w_k|v_j)$ est la probabilité que le mot en position i est w_k, sachant v_j
Une autre hypothèse : $P(a_i = w_k|v_j) = P(a_m = w_k|v_j), \forall i, m$

Algorithme

LEARN_NAIVE_BAYES_TEXT(*Exemples*, V)

1. *recolter tous les mots et autres tokens dans Exemples*
   - *Dictionnaire* $\leftarrow$ tous les mots et autres tokens dans *Exemples*
2. *calculer les termes* $P(v_j)$ *et* $P(w_k|v_j)$
   - Pour chaque valeur cible v_j dans V faire
     - $docs_j \leftarrow$ le sous-ensemble des documents de *Exemples* pour lesquels la valeur cible est v_j
     - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
     - $Text_j \leftarrow$ un seul document créé par concaténation de tous les membres de $docs_j$
     - $n \leftarrow$ nombre total de mots (tokens) dans $Text_j$
     - Pour chaque mot dans w_k in *Dictionnaire*
       - $n_k \leftarrow$ nombre de fois que le mot w_k apparaît dans $Text_j$
       - $P(w_k|v_j) \leftarrow \frac{n_k + 1}{n + |Vocabulaire|}$

Algorithme

LEARN_NAIVE_BAYES_TEXT(*Exemples*, V)

1. *recolter tous les mots et autres tokens dans Exemples*
   - *Dictionnaire* ← tous les mots et autres tokens dans *Exemples*
2. *calculer les termes* $P(v_j)$ *et* $P(w_k|v_j)$
   - Pour chaque valeur cible v_j dans V faire
     - $docs_j \leftarrow$ le sous-ensemble des documents de *Exemples* pour lesquels la valeur cible est v_j
     - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
     - $Text_j \leftarrow$ un seul document créé par concaténation de tous les membres de $docs_j$
     - $n \leftarrow$ nombre total de mots (tokens) dans $Text_j$
     - Pour chaque mot dans w_k in *Dictionnaire*
       - $n_k \leftarrow$ nombre de fois que le mot w_k apparaît dans $Text_j$
       - $P(w_k|v_j) \leftarrow \frac{n_k + 1}{n + |Vocabulaire|}$

Algorithme

LEARN_NAIVE_BAYES_TEXT(*Exemples*, V)

1. *recolter tous les mots et autres tokens dans Exemples*
   - *Dictionnaire* ← tous les mots et autres tokens dans *Exemples*
2. *calculer les termes* $P(v_j)$ *et* $P(w_k|v_j)$
   - Pour chaque valeur cible v_j dans V faire
     - $docs_j \leftarrow$ le sous-ensemble des documents de *Exemples* pour lesquels la valeur cible est v_j
     - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
     - $Text_j \leftarrow$ un seul document créé par concaténation de tous les membres de $docs_j$
     - $n \leftarrow$ nombre total de mots (tokens) dans $Text_j$
     - Pour chaque mot dans w_k in *Dictionnaire*
       - $n_k \leftarrow$ nombre de fois que le mot w_k apparaît dans $Text_j$
       - $P(w_k|v_j) \leftarrow \frac{n_k + 1}{n + |Vocabulaire|}$

Algorithme

LEARN_NAIVE_BAYES_TEXT(*Exemples*, V)

1. *recolter tous les mots et autres tokens dans Exemples*
- *Dictionnaire* $\leftarrow$ tous les mots et autres tokens dans *Exemples*
2. *calculer les termes* $P(v_j)$ *et* $P(w_k|v_j)$
- Pour chaque valeur cible v_j dans V faire
  - $docs_j \leftarrow$ le sous-ensemble des documents de *Exemples* pour lesquels la valeur cible est v_j
  - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
  - $Text_j \leftarrow$ un seul document créé par concaténation de tous les membres de $docs_j$
  - $n \leftarrow$ nombre total de mots (tokens) dans $Text_j$
  - Pour chaque mot dans w_k in *Dictionnaire*
    - $n_k \leftarrow$ nombre de fois que le mot w_k apparaît dans $Text_j$
    - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

Algorithme

LEARN_NAIVE_BAYES_TEXT(*Exemples*, V)

1. *recolter tous les mots et autres tokens dans Exemples*

- *Dictionnaire* $\leftarrow$ tous les mots et autres tokens dans *Exemples*

2. *calculer les termes* $P(v_j)$ *et* $P(w_k|v_j)$

- Pour chaque valeur cible v_j dans V faire
  - $docs_j \leftarrow$ le sous-ensemble des documents de *Exemples* pour lesquels la valeur cible est v_j
  - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
  - $Text_j \leftarrow$ un seul document créé par concaténation de tous les membres de $docs_j$
  - $n \leftarrow$ nombre total de mots (tokens) dans $Text_j$
  - Pour chaque mot dans w_k in *Dictionnaire*
    - $n_k \leftarrow$ nombre de fois que le mot w_k apparaît dans $Text_j$
    - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

Algorithme

LEARN_NAIVE_BAYES_TEXT(*Exemples*, V)

1. *recolter tous les mots et autres tokens dans Exemples*

- *Dictionnaire* $\leftarrow$ tous les mots et autres tokens dans *Exemples*

2. *calculer les termes* $P(v_j)$ *et* $P(w_k|v_j)$

- Pour chaque valeur cible v_j dans V faire
  - $docs_j \leftarrow$ le sous-ensemble des documents de *Exemples* pour lesquels la valeur cible est v_j
  - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
  - $Text_j \leftarrow$ un seul document créé par concaténation de tous les membres de $docs_j$
  - $n \leftarrow$ nombre total de mots (tokens) dans $Text_j$
  - Pour chaque mot dans w_k in *Dictionnaire*
    - $n_k \leftarrow$ nombre de fois que le mot w_k apparaît dans $Text_j$
    - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

Algorithme

LEARN_NAIVE_BAYES_TEXT(*Exemples*, V)

1. *recolter tous les mots et autres tokens dans Exemples*

- *Dictionnaire* $\leftarrow$ tous les mots et autres tokens dans *Exemples*

2. *calculer les termes* $P(v_j)$ *et* $P(w_k|v_j)$

- Pour chaque valeur cible v_j dans V faire
  - $docs_j \leftarrow$ le sous-ensemble des documents de *Exemples* pour lesquels la valeur cible est v_j
  - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
  - $Text_j \leftarrow$ un seul document créé par concaténation de tous les membres de $docs_j$
  - $n \leftarrow$ nombre total de mots (tokens) dans $Text_j$
  - Pour chaque mot dans w_k in *Dictionnaire*
    - $n_k \leftarrow$ nombre de fois que le mot w_k apparaît dans $Text_j$
    - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

Algorithme

LEARN_NAIVE_BAYES_TEXT(*Exemples*, V)

1. *recolter tous les mots et autres tokens dans Exemples*

- *Dictionnaire* ← tous les mots et autres tokens dans *Exemples*

2. *calculer les termes* $P(v_j)$ *et* $P(w_k|v_j)$

- Pour chaque valeur cible v_j dans V faire
  - $docs_j$ ← le sous-ensemble des documents de *Exemples* pour lesquels la valeur cible est v_j
  - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
  - $Text_j$ ← un seul document créé par concaténation de tous les membres de $docs_j$
  - n ← nombre total de mots (tokens) dans $Text_j$
  - Pour chaque mot dans w_k in *Dictionnaire*
    - n_k ← nombre de fois que le mot w_k apparaît dans $Text_j$
    - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

Algorithme

LEARN_NAIVE_BAYES_TEXT(*Exemples*, V)

1. recolter tous les mots et autres tokens dans Exemples

- *Dictionnaire* $\leftarrow$ tous les mots et autres tokens dans *Exemples*

2. calculer les termes $P(v_j)$ *et* $P(w_k|v_j)$

- Pour chaque valeur cible v_j dans V faire
  - $docs_j \leftarrow$ le sous-ensemble des documents de *Exemples* pour lesquels la valeur cible est v_j
  - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
  - $Text_j \leftarrow$ un seul document créé par concaténation de tous les membres de $docs_j$
  - $n \leftarrow$ nombre total de mots (tokens) dans $Text_j$
  - Pour chaque mot dans w_k in *Dictionnaire*
    - $n_k \leftarrow$ nombre de fois que le mot w_k apparaît dans $Text_j$
    - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

Algorithme

LEARN_NAIVE_BAYES_TEXT(*Exemples*, V)

1. *recolter tous les mots et autres tokens dans Exemples*

- *Dictionnaire* ← tous les mots et autres tokens dans *Exemples*

2. *calculer les termes* $P(v_j)$ *et* $P(w_k|v_j)$

- Pour chaque valeur cible v_j dans V faire
  - $docs_j$ ← le sous-ensemble des documents de *Exemples* pour lesquels la valeur cible est v_j
  - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
  - $Text_j$ ← un seul document créé par concaténation de tous les membres de $docs_j$
  - n ← nombre total de mots (tokens) dans $Text_j$
  - Pour chaque mot dans w_k in *Dictionnaire*
    - n_k ← nombre de fois que le mot w_k apparaît dans $Text_j$
    - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

Algorithme

LEARN_NAIVE_BAYES_TEXT(*Exemples*, V)

1. *recolter tous les mots et autres tokens dans Exemples*

- *Dictionnaire* ← tous les mots et autres tokens dans *Exemples*

2. *calculer les termes* $P(v_j)$ *et* $P(w_k|v_j)$

- Pour chaque valeur cible v_j dans V faire
  - $docs_j$ ← le sous-ensemble des documents de *Exemples* pour lesquels la valeur cible est v_j
  - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
  - $Text_j$ ← un seul document créé par concaténation de tous les membres de $docs_j$
  - n ← nombre total de mots (tokens) dans $Text_j$
  - Pour chaque mot dans w_k in *Dictionnaire*
    - n_k ← nombre de fois que le mot w_k apparaît dans $Text_j$
    - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

Text classification

CLASSIFY_NAIVE_BAYES_TEXT(*Doc*)

- *positions* ← toutes les position de mots dans *Doc* contenant de tokens dans *Dictionnaire*
- Returner v_{NB}, où

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_{i \in positions} P(a_i|v_j)$$

Text classification

CLASSIFY_NAIVE_BAYES_TEXT(*Doc*)

- *positions* ← toutes les position de mots dans *Doc* contenant de tokens dans *Dictionnaire*
- Returner v_{NB}, où

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_{i \in positions} P(a_i|v_j)$$

Twenty NewsGroups

Entraînement : 1000 document dans chaque groupe
Apprendre à classer de nouveaux documents sur la base du groupe auquel ils appartiennent

comp.graphics	misc.forsale
comp.os.ms-windows.misc	rec.autos
comp.sys.ibm.pc.hardware	rec.motorcycles
comp.sys.mac.hardware	rec.sport.baseball
comp.windows.x	rec.sport.hockey
alt.atheism	sci.space
soc.religion.christian	sci.crypt
talk.religion.misc	sci.electronics
talk.politics.mideast	sci.med
talk.politics.misc	
talk.politics.guns	

Naive Bayes : 89% exactitude de classification

Articles tiré de rec.sport.hockey

```
Path: cantaloupe.srv.cs.cmu.edu!das-news.harvard.edu!ogicse!uw
From: xxx@yyy.zzz.edu (John Doe)
Subject: Re: This year's biggest and worst (opinion)...
Date: 5 Apr 93 09:53:39 GMT

I can only comment on the Kings, but the most
obvious candidate for pleasant surprise is Alex
Zhitnik. He came highly touted as a defensive
defenseman, but he's clearly much more than that.
Great skater and hard shot (though wish he were
more accurate). In fact, he pretty much allowed
the Kings to trade away that huge defensive
liability Paul Coffey. Kelly Hrudey is only the
biggest disappointment if you thought he was any
good to begin with. But, at best, he's only a
mediocre goaltender. A better choice would be
Tomas Sandstrom, though not through any fault of
his own, but because some thugs in Toronto decided
```