MODELOS DE LENGUAJE BASADOS EN GRAMÁTICAS INCONTEXTUALES PROBABILÍSTICAS

PAOLA EUGENIA GUERRERO HURTADO

HAROLD ARMANDO CALVACHE MUÑOZ

JUAN PABLO MEJIA MEDINA

TRABAJO DE GRADO

En la modalidad de seminario presentado como requisito parcial para optar al título de licenciados en educación con especialidad matemáticas

Director

Dr. FREDY ANGEL AMAYA

UNIVERSIDAD DEL CAUCA

FACULTAD DE CIENCIAS NATURALES, EXACTAS Y DE LA EDUCACIÓN

DEPARTAMENTO DE MATEMÁTICAS

POPAYÁN

2004

MODELOS DE LENGUAJE BASADOS EN GRAMATICAS INCONTEXTUALES PROBABILISTICAS

PAOLA EUGENIA GUERRERO HURTADO

HAROLD ARMANDO CALVACHE MUÑOZ

JUAN PABLO MEJÍA MEDINA

UNIVERSIDAD DEL CAUCA
FACULTAD DE CIENCIAS NATURALES, EXACTAS Y DE LA EDUCACIÓN
DEPARTAMENTO DE MATEMÁTICAS
POPAYÁN
2004

Nota de aceptación

_____

_____

_____

Director

_____

Doctor Fredy Angel Amaya

Comité evaluador

_____

Doctor Luis Eduardo Montoya

_____

Magister Mauricio Maca

Fecha de sustentación: Popayán, octubre 29 de 2004

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

En este documento se presenta el trabajo titulado Modelos de Lenguaje Basados en Gramáticas Incontextuales Probabilísticas desarrollado en el seminario de grado orientado por el Doctor Freddy Amaya. Está enmarcado en la línea de Matemática Computacional perteneciente al Grupo de Estudio y Desarrollo Investigativo en Matemática Aplicada, exigido como requisito parcial para la obtención del título de Licenciado en Matemáticas otorgado por la Universidad del Cauca. En el se tratan dos modelos de lenguaje híbridos obtenidos a partir de la combinación de un modelo de n-gramas y una gramática incontextual, además dos algoritmos que permiten separar en clases las palabras de un vocabulario dado e incorporar tal clasificación a la estimación de probabilidades de un modelo de lenguaje basado en gramáticas. Para ello se revisarán brevemente los conceptos de: Gramáticas, Gramáticas Incontextuales (GIC), Gramáticas Incontextuales Probabilisticas (GIP), Modelos de Lenguaje Probabilísticos. Se presenta el algoritmo de estimación estándar (Inside-Outside IO o Viterbi), a través del cual se pueden estimar las probabilidades de las reglas de una gramática; se presentan dos algoritmos para agrupar en clases el vocabulario, Algoritmo de Della Pietra y Algoritmo de Ney, y los resultados experimentales del uso de la clasificación en el modelo de lenguaje.

INTRODUCCIÓN

El Reconocimiento del habla es un proceso de clasificación de patrones, cuyo objetivo es clasificar la señal de entrada (onda acústica) en una secuencia de patrones previamente aprendidos y almacenados en unos diccionarios de modelos acústicos y de lenguaje, que permitirá obtener automáticamente una secuencia de caracteres que representan la señal de entrada, similar planteamiento se hace para otras aplicaciones tales como: Traducción automática, reconocimiento óptico de caracteres, procesamiento de lenguaje natural, etc.

Este proceso de clasificación supone que se puede asignar a cada segmento o conjuntos consecutivos de segmentos una unidad con significado lingüístico y finalmente mediante un procesador lingüístico se puede dar significado a las secuencias de unidades. Este último paso del sistema supone incorporar al sistema de Reconocimiento Automático del Habla (RAH) conocimiento acerca de la estructura sintáctica, semántica y pragmática del lenguaje de la aplicación.

Sin embargo, los sistemas actuales de RAH solo incorporan estas fuentes de conocimiento sobre tareas muy restringidas y controladas, estando la mayoría de ellos en experimentación en condiciones de laboratorio. Matemáticamente, el problema del reconocimiento automático del habla se puede formular desde un punto de vista estadístico. Para ello se supone que;

O : Representa una secuencia de T medidas de la señal de voz
$O = O_1 O_2 ... O_T$, (datos acústicos).

W : Es una secuencia de **N** palabras que pertenecen a un vocabulario

conocido W = $W_1 W_2 ... W_N$.

P (W | O) : Es la probabilidad de que la secuencia de palabras **W** se haya pronunciado dada la observación de los datos acústicos **O**.

El sistema de reconocimiento debe decidir la secuencia de palabras **W** que maximice la probabilidad **P (W | O)**

$$\boldsymbol{W}' = \arg\max_{\boldsymbol{W} \in \boldsymbol{V}^+} \boldsymbol{P}(\boldsymbol{W} \mid \boldsymbol{O})$$

Utilizando la fórmula de Bayes se obtiene:

$$\boldsymbol{P}(\boldsymbol{W} \mid \boldsymbol{O}) = \frac{\boldsymbol{P}(\boldsymbol{O} \mid \boldsymbol{W})\, \boldsymbol{P}(\boldsymbol{W})}{\boldsymbol{P}(\boldsymbol{O})}$$

Sin embargo, como la probabilidad de la secuencia de datos acústicos **P(O)** es la misma independientemente de la secuencia de palabras pronunciada, en el proceso de maximización, esta probabilidad puede ser eliminada, obteniendose así la ***fórmula fundamental del reconocimiento automático del habla***

$$\boldsymbol{W}' = \arg\max_{\boldsymbol{W} \in \boldsymbol{V}^+} \boldsymbol{P}(\boldsymbol{O} \mid \boldsymbol{W}) \boldsymbol{P}(\boldsymbol{W})$$

Es decir, la secuencia de palabras reconocida es aquella que maximiza el producto de dos probabilidades, una, $\boldsymbol{P}(\boldsymbol{O} \mid \boldsymbol{W})$, que relaciona los datos acústicos con la secuencia de palabras y que se denominará modelo acústico y $\boldsymbol{P}(\boldsymbol{W})$ que únicamente depende de la secuencia de palabras y que se denominará ***modelo de lenguaje***.

Un modelo de lenguaje permite asignar una probabilidad, $P(w_1 ... w_n)$ a cada posible frase de un determinado lenguaje. Se han propuesto diversidad de modelos para calcular dicha probabilidad. El modelo más popular y utilizado es el modelo de n-gramas que calcula la probabilidad de la i-ésima palabra con base en las n-1 anteriores. Si $n = 3$, se llama *modelo de trigrama*. Si $n = 2$, *modelo de bigrama*. Si $n = 1$, *modelo de unigrama*. Para el caso de bigramas, se define $P(w_i | w_{i-1})$ como la probabilidad de que la palabra w_i siga a la palabra w_{i-1}. Usualmente estas probabilidades suelen estimarse

utilizando una gran colección de texto denominado corpus de entrenamiento siguiendo diferentes estrategias [CG98].

El procesamiento del lenguaje natural es un área de la informática que tiene que ver con los conocimientos y técnicas necesarios, para que las computadoras puedan procesar el mismo lenguaje que usamos los humanos en el habla cotidiana. La palabra "procesar"se refiere a las habilidades de comprender oraciones de un lenguaje humano como por ejemplo, el español, y de generar oraciones en el mismo o en otro lenguaje humano.

Entre las diversas herramientas adecuadas para aplicaciones de procesamiento del lenguaje natural, las gramáticas formales permiten caracterizar con precisión las oraciones de un lenguaje, así como distinguir su estructura y componentes. Dada una gramática y una oración, es posible determinar con precisión si la oración pertenece o no al lenguaje descrito por la gramática. Las Gramáticas son una herramienta de especificación para representar eficientemente las restricciones sintácticas existentes entre las palabras que constituyen las oraciones de un lenguaje natural. Las gramáticas son útiles en otras aplicaciones: Reconocimiento del Habla, Traducción Automática, etc. Sin embargo el costo computacional del entrenamiento de una gramática tanto para la inferencia gramatical como para la estimación de parámetros es muy grande.

En este trabajo se estudian algunos aspectos de las Gramáticas Incontextuales Probabilísticas y su aplicación en Modelización del Lenguaje (ML). Las Gramáticas Incontextuales (GI) constituyen un tipo importante de modelos ya que permiten representar dependencias a largo término entre las palabras. Una Gramática Incontextual Probabilística (GIP) es una extensión natural de una Gramática Incontextual que se compone de dos partes: Un conjunto de reglas que conforman la parte estructural de la misma y unas funciones de distribución de probabilidad asociadas a estas reglas. La aplicación de las GIP para modelizar el lenguaje introduce dos problemas principales, en primer lugar, cómo determinar la relación entre las palabras del lenguaje y en segundo lugar cómo realizar la interpretación en forma eficiente.

El problema de Modelización del Lenguaje consiste en definir e implementar los métodos para calcular las probabilidades $P(w_1...w_n)$, un tipo de modelo es el basado en gramáticas en las que dicha probabilidad se calcula mediante una GIP, otro tipo de modelo es mediante el modelo de n-gramas que calcula esta probabilidad basándose en el supuesto de que la probabilidad de una palabra está condicionada por las $n-1$ palabras anteriores.

Son dos los problemas que se consideran al utilizar las GIP como modelo de lenguaje: su aprendizaje o la obtención de una GIP que represente un lenguaje; y su integración como modelo de interpretación en tareas complejas de Modelización de Lenguaje. En este documento se abordará el proceso de aprendizaje de una GIP, el cual puede descomponerse en el aprendizaje de una gramática característica y/o el aprendizaje de las probabilidades asociadas a las reglas. El aprendizaje de la gramática consiste en recoger la información estructural presente en una muestra de manera que se puedan aprender las reglas de derivación, el aprendizaje de la información probabilísitica de la GIP consiste en la estimación de los parámetros de la GIP.

En cuanto al aprendizaje de las probabilidades de una GIP, se realiza optimizando una función criterio, a partir de un texto recolectado denominado corpus de entrenamiento, mediante la utilización de los algoritmos Inside-Outside y Viterbi. El corpus está formado por frases pertenecientes al lenguaje o por cadenas pertenecientes al lenguaje. En este trabajo se ha utilizado como corpus de aprendizaje de las probabilidades de la GIP, una muestra compuesta de cadenas extraídas del corpus denominado *Wall Street Journal* [MSM93].

Ya se ha hablado que el objetivo de un modelo de lenguaje es proporcionar una estimación de la probabilidad $P(w_1...w_n)$ de una secuencia de palabras $w_1...w_n$. Esta probabilidad se puede calcular como el producto de la secuencia de probabilidades condicionales $P(w_i|w_1...w_{i-1})$. Dicho producto, en el caso de utilizar un modelo de lenguaje

de bigramas, queda expresado de la siguiente manera, $P(w_1...w_n) \approx \prod_{i=1}^{n} P(w_i|w_{i-1})$.

El modelo de n - gramas recoge muy bien las relaciones locales en la frase, las probabilidades de sus eventos elementales se estiman con facilidad, es fácil de implementar y su formulación es muy sencilla. Pero también se presentan algunas limitaciones, por razones computacionales el valor de n se debe mantener bajo ($n \leq 3$), los n-gramas no se acomodan a los cambios dinámicos en el discurso, se tiene problemas de dispersión: hay una gran cantidad de eventos, en el espacio de eventos del modelo, que no son vistos durante el entrenamiento y por tanto la frecuencia relativa es cero lo cual implicará que se les asigne probabilidad 0.

Algunas desventajas que presenta la estimación de un modelo de lenguaje es la dispersión de los datos disponibles para su entrenamiento. Esto es así aún para el modelo relativamente sencillo de bigramas [BS03]. Existen tantas posibles combinaciones de pares de palabras que es imposible observarlas todas el suficiente número de veces. El agrupamiento de aquellas palabras que son similares entre sí es una forma de combatir la dispersión de datos de entrenamiento disponibles. Si se agruparan satisfactoriamente palabras similares, se podrían realizar predicciones más razonables de aquellas secuencias de palabras que no se han visto asumiendo que son similares a otras ya vistas.

Para tratar con el problema de dispersión y hacer mas eficiente la estimación de la gramática se agrupa el vocabulario en clases de palabras. Para el presente trabajo se han utilizado dos algoritmos de agrupamiento, uno (Algoritmo de Della Pietra) basado en la información mutua y el otro (Algoritmo de Ney) basado en la verosimilitud; finalmente se realizará una comparación experimental de los modelos de bi-gramas en estos dos algoritmos.

Para tratar el problema de localidad, se utiliza una GIP combinada con el modelo de n-gramas de clases y los terminales de la GIP son clases en las cuales se ha agrupado el vocabulario, los no terminales de la GIP son las etiquetas semánticas extraídas del

Wall Street Journal. El modelo así definido recoge las dependencias locales a nivel de palabras como también las relaciones estructurales entre las palabras del texto.

Con el fin de alcanzar este objetivo se ha realizado un estudio tanto teórico como experimental, de diferentes aspectos de la Teoría de Lenguajes Formales y Probabilísticos que se enumeran a continuación:

En el capítulo 1 se revisarán brevemente las principales ideas matemáticas necesarias para la comprensión del material de los posteriores capitulos. En el capítulo 2 se estudiarán las gramáticas libres de contexto y los lenguajes libres de contexto que estas describen. En el capítulo 3 se estudian algoritmos para decidir la pertenencia de una cadena a un lenguaje generado por una GI, es decir encontrar una secuencia de derivaciones que mediante reglas de la GI generen la cadena. En el capítulo 4 se presentarán algunas definiciones sobre lenguajes y gramáticas probabilísticas, la introducción de nociones para realizar el análisis sintáctico probabilístico de una cadena, además de dos de los algoritmos clásicos de estimación de las Gramáticas Incontextuales Probabilísticas. En el capítulo 5 se estudiará el cálculo de la probabilidad de una frase deducida desde el punto de vista de teoría de la información. Se revisará el concepto de modelo de lenguaje; se dará una definición de modelos de n-gramas y se presentarán dos algoritmos para agrupar el vocabulario de una aplicación en clases, los cuales agrupan según la pérdida de información mutua y el otro según la verosimilitud. En el capítulo 6 se realiza la integración del modelo basado en GIP y el modelo basado en n-gramas de clase, esperando que tal integración resulte en un modelo de mayor potencia expresiva que el de n-gramas solo. Finalmente en el capítulo 7 se describen los experimentos realizados con los dos algoritmos de clasificación por clases y se muestran los resultados experimentales.

Capítulo 1

PREAMBULO

En este capítulo se revisan brevemente las principales ideas matemáticas necesarias para la comprensión del material de los posteriores capítulos. Estos conceptos incluyen grafos, árboles, conjuntos, relaciones, cadenas, etc.

1.1 Cadenas, Alfabetos Y Lenguajes

Un *símbolo* es una entidad abstracta que, de la misma manera que los conceptos de *punto* y *línea*, no se define rigurosamente. Las letras y los dígitos son ejemplos de símbolos usados con frecuencia. Una cadena (o palabra) es una secuencia finita de símbolos. Por ejemplo, a, b y c son símbolos, y $abcb$ es una cadena. La longitud de una cadena w se denota como $|w|$ y es el número de símbolos que componen la cadena. Por ejemplo, acbba tiene longitud 5. La cadena vacía, denotada por λ, es la cadena que consiste en cero símbolos. Por lo tanto, $|\lambda| = 0$.

Un *alfabeto* es un conjunto finito de símbolos. Un *lenguaje* (formal) es un conjunto de cadenas de símbolos tomados de algún alfabeto fijo $\sum$, el cual se denota como $\sum^*$. $\sum^+$ denotará el conjunto de todas las cadenas de longitud mayor o igual que 1 que se pueden formar con elementos de $\sum$, es decir, $\sum^+ = \sum^* - \{\lambda\}$.

1.2.2 Árboles

Un *árbol* se puede definir de varias maneras. Una forma natural de hacerlo es recusivamente [AM95].

Definición: Un árbol es una colección o conjunto de nodos (o *vértices*). La colección puede estar vacía, lo que se denotará con Λ. Si no es así, un árbol consiste en un nodo distinguido v, conocido como raíz, y cero o más árboles $A_1, A_2, ..., A_k$, cada uno de los cuales tiene su raíz conectada a v por medio de un *arco* (o *rama*) dirigida. Se denota un arco de v a w como $v \rightarrow w$. Si $v \rightarrow w$ es un arco, se dice que v es un *predecesor* (o *antecesor*) de w y que w es un *sucesor* (o *descendiente*) de v.

Se dice que la raíz de cada $A_1, A_2, ..., A_k$ es un sucesor de v, y que v es el predecesor de cada raíz de los subárboles. La figura 1.1 muestra un árbol representativo según la definición recursiva.

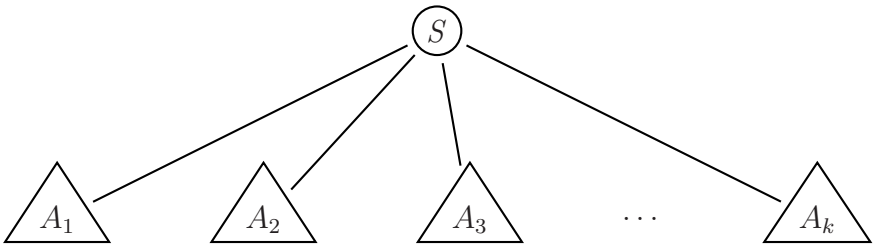


Fig. 1.1 Árbol genérico

Una *trayectoria* (o *camino*) desde un nodo n_1 a otro n_k se define como la secuencia de nodos $n_1, n_2, ..., n_k$ tal que n_i es el predecesor de n_{i+1} para $1 \leq i < k$. La *longitud* de esta trayectoria es el número de arcos que lo forman, $k - 1$. Existe una trayectoria de longitud cero entre cada nodo y él mismo. Además un árbol tiene exactamente una trayectoria de la raíz a cada nodo.

De la definición se deduce que un árbol es un conjunto de n nodos, uno de los cuales es la raíz, y $n - 1$ arcos, lo cual se evidencia del hecho de que cada uno de estos conecta

algún nodo con su predecesor. Además se tiene que todo nodo tiene un predecesor, excepto la raíz. Cada nodo puede tener un número arbitrario de sucesores, incluyendo cero. Los nodos sin sucesores se conocen como términales (u hojas).

Para cualquier nodo n_i la $profundidad$ de n_i es la longitud del camino único entre la raíz y n_i. Así, la raíz tiene una profundidad cero. La $altura$ de n_i es el camino más largo de n_i a una hoja o terminal. Por tanto, todos los terminales son de altura cero. La profundidad del árbol es igual a la profundidad del terminal más profundo.

La Fig. 1.2 muestra el ejemplo de un árbol que es un diagrama de la oración: "El veloz perro amarillo corrió hacia la hermosa casa". Donde los nodos no terminales son categorías léxicas y los terminales son palabras del vocabulario en español. En el árbol se puede observar que, la raíz es "<oración>". EL nodo "<sujeto>"tiene a "<oración>"como predecesor y a "<frase sustantiva>"como sucesor. El árbol tiene una altura de 8 y Los terminales son: el, veloz, perro, amarillo, corrió, hacia, la, hermosa, casa. Que leídos de izquierda a derecha formarán la oración.

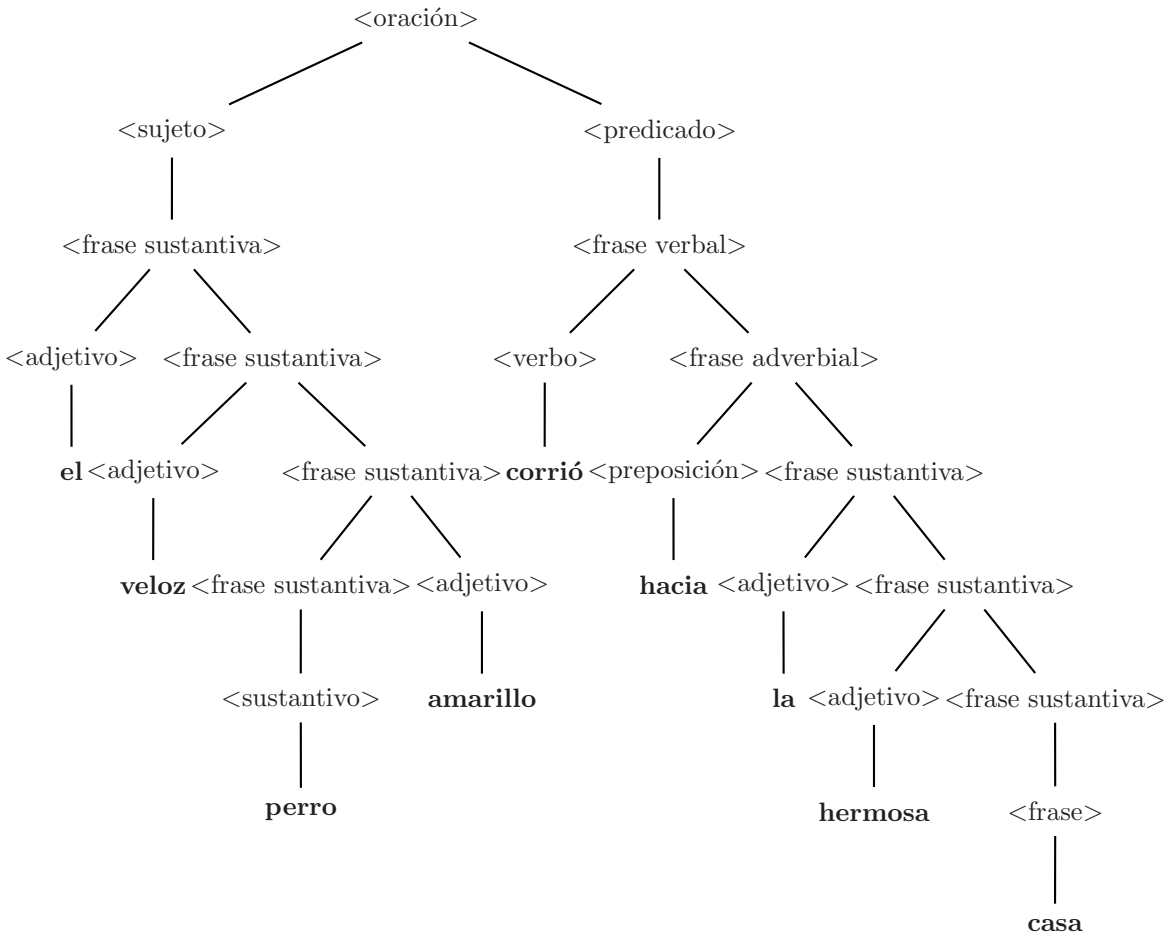


Fig.1.2 Un árbol sintáctico de una oración en castellano

Capítulo 2

GRAMÁTICAS

En este capítulo se estudian las gramáticas libres de contexto y los lenguajes libres de contexto que estas describen. Los lenguajes libres de contexto son de gran importancia práctica, sobre todo en la definición de lenguajes de programación, en la formalización del concepto de análisis gramatical, simplificación de la traducción de lenguajes de programación y en otras aplicaciones del procesamiento de cadenas.

2.1 Definición de Gramáticas

La necesidad de describir los lenguajes naturales originó el desarrollo de las gramáticas. Aunque por ciertas razones, las gramáticas, en general, no se consideran como adecuadas para la descripción de los lenguajes naturales, como el español. Por ejemplo, si extendemos reglas sintácticas como las siguientes:

<oración> → <frase sustantiva> <frase verbal>
<frase sustantiva> → <adjetivo> <frase sustantiva>
<frase sustantiva> → <sustantivo>
<sustantivo> → niño
<adjetivo> → pequeño

a todo el idioma, podríamos tomar "la mesa" como una frase sustantiva y "camina" como una frase verbal. Por consiguiente, "la mesa camina" sería una oración, que carece de significado aunque es correcta sintácticamente. Es claro que se necesita alguna información semántica para reglamentar cadenas sin significado que son sintácticamente correctas.

Definición 1: Una gramática es una cuatrupla $G = (V, T, P, S)$, donde:

V : Conjunto finito de variables o no terminales

T : Conjunto finito de terminales. V y T son disjuntos

P : Conjunto finito de producciones; cada producción [1] es de la forma $\boldsymbol{\alpha} \rightarrow \boldsymbol{\gamma}$, en donde $\boldsymbol{\alpha}$ y $\boldsymbol{\gamma}$ son cadenas de símbolos tomados de $(V \cup T)^*$

S : es una variable especial de V conocida como el símbolo inicial

Ejemplo 2.1

Sea $G = (V, T, P, S)$, donde: $V = \{\text{A, B, S}\}$, $T = \{a,\ b\ \}$, S es el símbolo inicial y P consiste en:

$$S \rightarrow aA$$
$$aB \rightarrow bBAa$$
$$ABb \rightarrow abB$$

Ejemplo 2.2

Sea $G = (V, T, P, S)$, donde: $V = \{\text{A, B, C, S}\}$, $T = \{a,\ b,\ c\ \}$, S es el símbolo inicial y P consiste en:

$$S \rightarrow aAb$$
$$aBA \rightarrow BACab$$
$$ACb \rightarrow aCbB$$
$$CBb \rightarrow cAbB$$

2.2 Gramáticas Incontextuales

Las gramáticas incontextuales (GI) juegan un importante papel en lingüística computacional. El uso de las gramáticas incontextuales ha simplificado considerablemente la definición de lenguajes de computación y la construcción de compiladores. Esto se debió en parte a la forma en que la mayoría de las construcciones de programas de

[1] intuitivamente una producción es una regla que permite sustituir la cadena del lado izquierdo por la cadena del lado derecho

computación son descritas por gramáticas. En el ejemplo 2.3 considerese la gramática $G = (V, T, P, S)$, donde: $V = \{E, S\}$, $T = \{*, +,), (, id\}$, S es el símbolo inicial.

Ejemplo 2.3

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow (E)$
4. $E \rightarrow$ id

P consiste en el conjunto de producciones numerado de 1 - 4, que define las expresiones aritméticas que contienen operadores + y $*$, y con los operandos representado por el símbolo id. Aquí E es la única variable, y los terminales son +, $*$,),(e id. Las dos primeras producciones nos dicen que "E " puede componerse de dos "E" conectadas por un signo de adición o de multiplicación. La tercera producción dice que "E" puede ser otra "E" rodeada por paréntesis. La última dice que un sólo operador es una "E".

Mediante la aplicación de estas producciones podemos obtener otras un poco más complicadas.

Ejemplo 2.4

$$\begin{aligned} E &\Rightarrow E + E \\ &\Rightarrow (E) + E \\ &\Rightarrow (E * E) + E \\ &\Rightarrow (E * E) + \mathbf{id} \\ &\Rightarrow (E * \mathbf{id}) + \mathbf{id} \\ &\Rightarrow (\mathbf{id} * \mathbf{id}) + \mathbf{id} \end{aligned}$$

El símbolo $\Rightarrow$ denota la derivación, es decir la sustitución de una variable por el lado derecho de una producción para esa variable (adelante en este texto se especificará formalmente lo que significa la derivación). La primera línea se obtiene de la primera

producción. La segunda línea se obtiene sustituyendo la primera "E" en la línea 1 por el lado izquierdo de la tercera producción. Las líneas restantes son el resultado de aplicar las producciones (2), (4),(4) y (4). La última línea (id $*$ id) + id, consiste solamente en símbolos términales y por tanto es una palabra en el lenguaje de "E".

Definición 2: Una gramática incontextual (GI) es una cuatrupla $G = (V, T, P, S)$, donde: Cada producción en P es de la forma $\boldsymbol{A \longrightarrow \alpha}$, en donde A es una variable y $\boldsymbol{\alpha}$ es una cadena de símbolos tomados de $(V \cup T)^*$.

Ejemplo 2.5

Sea $G = (V, T, P, S)$, donde : $V = \{S, A\}$, $T = \{$a, b$\}$, S es el símbolo inicial y P consiste en:

$$S \to aAS$$
$$S \to a$$
$$A \to SbA$$
$$A \to SS$$
$$A \to ba$$

En adelante se utilizan las siguientes reglas con respecto a las gramáticas.

1. Las letras mayúsculas tales como A, B, C, D, E y S representan variables; S es el símbolo inicial.

2. Las letras minúsculas como *a, b, c, d, e* y los dígitos serán los terminales.

3. Las letras minúsculas *x, u, v, w* denotan cadenas de terminales.

4. Las letras griegas minúsculas α, β, γ denotan cadenas de variables y terminales.

Además si $A \to \alpha_1$, $A \to \alpha_2$,..., $A \to \alpha_k$ son las producciones para la variable A de alguna gramática, entonces podemos expresarlas mediante la notación

$$A \rightarrow \alpha_1 \mid \alpha_2 \mid ... \mid \alpha_k$$

en donde la línea vertical se lee "o".

2.2.1 Derivaciones y Lenguajes

Antes de definir de manera formal el lenguaje generado por una gramática $G = (V, T, P, S)$, a continuación se mostrará la notación con la cual se representa una derivación. Primero se definen dos relaciones $\underset{G}{\Longrightarrow}$ y $\overset{*}{\underset{G}{\Longrightarrow}}$ entre cadenas de $(V \cup T)^*$. Si α y γ son cadenas cualesquiera de $(V \cup T)^*$ y $A \rightarrow \beta$ es una producción de P, entonces $\alpha A \gamma \underset{G}{\Longrightarrow} \alpha\beta\gamma$. Se dice que la producción $A \rightarrow \beta$ está aplicada a la cadena $\alpha A \gamma$ para obtener $\alpha\beta\gamma$ o que $\alpha A \gamma$ *deriva directamente* a $\alpha\beta\gamma$ en la gramática G.

Supóngase que $\alpha_1, \alpha_2, ..., \alpha_m$ son cadenas de $(V \cup T)^*$, $m \geq 1$ y

$$\alpha_1 \underset{G}{\Longrightarrow} \alpha_2, \alpha_2 \underset{G}{\Longrightarrow} \alpha_3, ..., \alpha_{m-1} \underset{G}{\Longrightarrow} \alpha_m.$$

Entonces decimos que $\alpha_1 \overset{*}{\underset{G}{\Longrightarrow}} \alpha_m$ o α_1 deriva a α_m en la gramática G. Por lo general si es claro qué la gramática G está involucrada, utilizamos $\Longrightarrow$ en lugar de $\underset{G}{\Longrightarrow}$ y $\overset{*}{\underset{G}{\Longrightarrow}}$. Si α deriva a β exactamente en i pasos, decimos que $\alpha \overset{i}{\Longrightarrow} \beta$.

Definición 3: El *lenguaje generado por G* denotado por L(G) es $\{w | w$ está en T^* y S $\overset{*}{\underset{G}{\Longrightarrow}} w\}$. Es decir, una cadena está en L(G) si:

1. La cadena consiste solamente en terminales.

2. La cadena puede derivarse a partir de S.

Definición 4: Llamaremos a L un *lenguaje libre de contexto* si éste es L(G) para alguna gramática libre de contexto G. Una cadena de terminales y variables α se conoce como *forma oracional* si $S \overset{*}{\underset{G}{\Longrightarrow}} \alpha$. Definimos las gramáticas G_1 y G_2 como *equivalentes* si $L(G_1) = L(G_2)$.

Ejemplo 2.6

Sea la gramática $G = (V, T, P, S)$, en donde $V = \{S\}$, $T = \{a,\ b\}$ y $P = \{S \to aSb, S \to ab\}$ [HU79]. Aquí S es la única variable; a y b son terminales. Si aplicamos la primera producción $n-1$ veces, seguida por una aplicación de la segunda producción, se obtiene

$$S \Longrightarrow aSb \Longrightarrow aaSbb \Longrightarrow a^3Sb^3 \Longrightarrow ... \Longrightarrow a^{n-1}Sb^{n-1} \Longrightarrow a^nb^n.$$

Las únicas cadenas de L(G) son a^nb^n para $n \geq 1$. Cada vez que $S \to aSb$ se utiliza, el número de Ss en la cadena permanece igual (aparece una sola vez en la cadena). Después de utilizar la producción $S \to ab$ se tiene que el número de Ss en la forma oracional disminuye en uno. Luego después de utilizar $S \to ab$ ninguna S permanece en la cadena resultante. Puesto que ambas producciones tienen una S a la izquierda, el único orden en el cual las producciones pueden aplicarse es $S \to aSb$, algún número de veces seguida por la aplicación de $S \to ab$. Así que L(G) = $\{a^nb^n \mid n \geq 1\}$

Ejemplo 2.7

Sea ahora la gramática G =(V, T, P, S), en la que $V = \{\text{S, A, B}\}$, $T = \{a,\ b\}$ y P consiste en las siguientes producciones:

$S \to aB$	$A \to bAA$
$S \to bA$	$B \to b$
$A \to a$	$B \to bS$
$A \to aS$	$B \to aBB$

El lenguaje L(G) es el conjunto de todas las palabras que pertenecen a T^+ y que consiste en cadenas con un número igual de as y bs [HU79]. Probemos esta proposición por inducción sobre la longitud de una palabra.

Hipótesis inductiva Para w en T^+,

1. $S \overset{*}{\Longrightarrow} w$ si y solo si w consiste en un número igual de as ybs.

2. A $\overset{*}{\Longrightarrow} w$ si y solo si en la cadena w el número de as es uno más que el número de bs.

3. B $\overset{*}{\Longrightarrow} w$ si y solo si en la cadena w el número de bs es uno más que el número de as.

La hipótesis inductiva es verdadera para $|w| = 1$, ya que $A \overset{*}{\Longrightarrow} a$, $B \overset{*}{\Longrightarrow} b$ y ninguna cadena terminal es derivable de S. También, porque todas las producciones, con excepción de $A \to a$ y $B \to b$, aumentan la longitud de una cadena, ninguna cadena de longitud distinta de a y b es derivable de A y B, respectivamente. También se tiene que ninguna cadena de longitud uno es derivable de S.

Supongase que la hipótesis inductiva es verdadera para toda w de longitud menor o igual a k-1. Mostremos que es verdadera para $|w| = k$.
Primero, si $S \overset{*}{\Longrightarrow} w$, entonces la derivación debe empezar con

- $S \to aB$ en este caso w es de la forma aw_1 en el que $|w_1| = k - 1$ y $B \overset{*}{\Longrightarrow} w_1$. Por la hipótesis de inducción, el número de bs en w_1 es uno más que el número de as, de modo que w consiste en un número igual de as y bs.

- $S \to bA$ en este caso w es de la forma bw_1 en el que $|w_1| = k - 1$ y $A \overset{*}{\Longrightarrow} w_1$. Por la hipótesis de inducción, el número de as en w_1 es uno más que el número de bs, de modo que w consiste en un número igual de as y bs.

Se probará ahora de la parte (1), si $|w| = k$ y w consiste en número igual de as y bs, entonces $S \Longrightarrow w$. El primer símbolo de w es a o es b.

Supóngase que $w = aw_1$. Ahora $|w_1| = k - 1$ y w_1 tiene una b mas que as. Por la hipótesis de inducción, $B \Longrightarrow w_1$. Entonces $S \Longrightarrow aB \overset{*}{\Longrightarrow} aw_1 = w$.

Supongase ahora que $w = bw_1$. Luego $|w_1| = k - 1$ y tiene una a mas que bs . Por la hipótesis de inducción, $A \Longrightarrow w_1$. Entonces $S \Longrightarrow bA \overset{*}{\Longrightarrow} bw_1 = w$.

Se completa la prueba demostrando la parte (2) y (3). Esto se puede hacer de una manera similar al método utilizado para la parte (1).

2.3 Árboles de derivación

Es de utilidad ver las derivaciones a través de un árbol, a este diagrama se le llama *árbol de derivación o análisis sintáctico*, y superpone una estructura sobre las palabras de un lenguaje, que es útil en aplicaciones como la compilación de los lenguajes de programación [VAPP03].

De manera formal un árbol es un *árbol de derivación* si, dada $G = (V, T, P, S)$ una gramática libre de contexto para G se tiene que:

1. El conjunto de vértices del árbol es $V \cup T \cup \{\lambda\}$.
2. La raíz es S.
3. Todo vértice interior (vértice diferente de la raíz y de los terminales) A es un elemento de V.
4. Si $A \rightarrow X_1, X_2, ..., X_k$ es una regla de producción entonces existe un vértice A cuyos sucesores corresponden a los vértices $X_1, X_2, ..., X_k$ respectivamente de izquierda a derecha.
5. Si vértice n tiene etiqueta λ, entonces n es un terminal y no tiene sucesores

Ejemplo 2.8

Considerese la gramática $G = (\{S, A\}, \{a, b\}, P, S)$, en donde P consiste en

$$S \to aAS \mid a$$
$$A \to SbA \mid SS \mid ba$$

Fig. 2.1 (a)

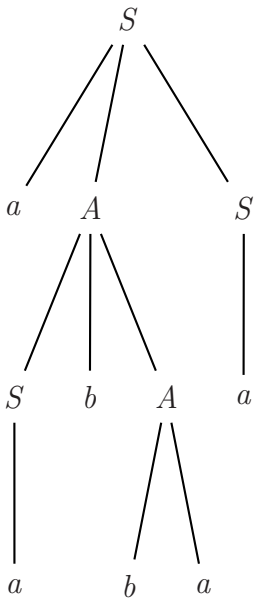
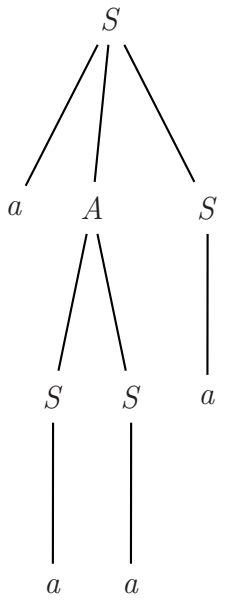


Fig. 2.1 (b)



Observación:

En la **Fig. 2.1** (a) Si se examinan las etiquetas de las hojas de izquierda a derecha, se tiene una forma oracional. Llamada esta cadena el *producto* del árbol de derivación, que en este caso es $aabbaa$. Note que en este caso todas las hojas tienen terminales como etiquetas, pero no existe una razón por la cual esto deba ser siempre así, algunas hojas podrían tener etiqueta λ de alguna variable. Además que $S \overset{*}{\underset{G}{\Longrightarrow}} aabbaa$ por la derivación. En la **Fig. 2.1** (b) se tiene otro ejemplo de producto del árbol de derivación, en este caso $aaaa$.

$$S \Longrightarrow aAs \Longrightarrow aSbAS \Longrightarrow aabAS \Longrightarrow aabbaS \Longrightarrow aabbaa.$$

$$S \Longrightarrow aAs \Longrightarrow aSSS \Longrightarrow aaaa.$$

2.3.1 Subárbol de derivación

Un subárbol de derivación es un vértice particular del árbol junto con todos sus descendientes, las aristas que los conectan y sus etiquetas. Tiene la apariencia de un árbol de derivación, excepto que la etiqueta de la raíz puede no ser el símbolo inicial de la gramática.

Ejemplo 2.9

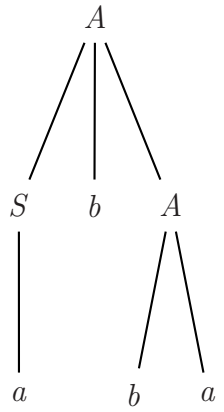


Fig. 2.2

Observación:

La **Fig. 2.2** muestra un subárbol del árbol de la figura 2.1. (a). El producto del subárbol es $abba$ la etiqueta de la raíz del subárbol es A y $A \stackrel{*}{\Longrightarrow} abba$. Una derivación, en este caso, es

$$A \Longrightarrow SbA \Longrightarrow abA \Longrightarrow abba.$$

2.3.2 La relación entre los árboles de derivación y las derivaciones

Teorema 2.1 Sea $G = (V, T, P, S)$ una gramática incontextual. Entonces $S \stackrel{*}{\Longrightarrow} \alpha$ si y sólo si existe un árbol de derivación de la gramática G con producto α [HU79].

Demostración Es más fácil probar algo superior al teorema. Lo que se demostrará es que para cualquier A en V, $A \stackrel{*}{\Longrightarrow} \alpha$ si y sólo si existe un árbol con raíz A para el cual α es el producto.

Primero, supóngase que α es el producto de un árbol con raíz A. Probamos, por inducción sobre el número de vértices interiores del árbol, que $A \stackrel{*}{\Longrightarrow} \alpha$. Si existe sólo un vértice interior, el árbol deberá verse como el que se presenta en la Fig. 2.3. En ese caso, $X_1X_2....X_n$ debe ser α, y $A \rightarrow X_i$ para $i = 1, ..., n$ debe ser una regla de producción de P, según la definición de árbol de derivación.

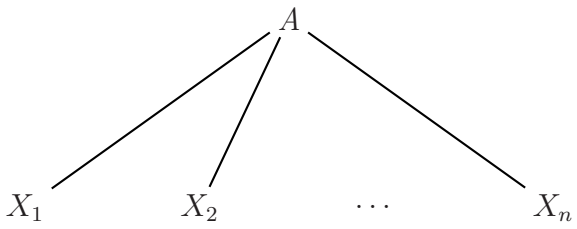


Fig. 2.3 Árbol con un vértice interior

Si existen k-1 vértices interiores. También supóngase que α es el producto de un árbol A con k vértices interiores para algún $k > 1$. Considérense los sucesores de la raíz. No todos pueden ser terminales (estaríamos en el caso anterior). Sean las etiquetas de los sucesores $X_1, X_2, ..., X_n$ de izquierda a derecha. Entonces, $A \rightarrow X_1X_2....X_n$ es una producción en P. Nótese que n puede ser cualquier entero mayor o igual que uno en el siguiente argumento.

Si el i-ésimo sucesor no es un terminal, sea $\alpha_i = X_i$. Si $j < i$, el vértice j y todos sus descendientes se encuentran a la izquierda del vértice i y de todos sus descendientes. Por consiguiente, $\alpha = \alpha_1\alpha_2...\alpha_n$. Un subárbol debe tener menos vértices interiores que los que tiene su árbol, a menos que el subárbol sea el árbol completo. Por la hipótesis de inducción, para cada vértice i que no sea un terminal, $X_i \stackrel{*}{\Longrightarrow} \alpha_i$, ya que el subárbol con raíz X_i no es el árbol completo. Si $X_i = \alpha_i$, entonces, $X_i \stackrel{*}{\Longrightarrow} \alpha_i$. Podemos juntar todas estas derivaciones parciales para ver que

$$A \Longrightarrow X_1X_2....X_n \stackrel{*}{\Longrightarrow} \alpha_1X_2....X_n \stackrel{*}{\Longrightarrow} \alpha_1\alpha_2X_3....X_n \stackrel{*}{\Longrightarrow} ... \stackrel{*}{\Longrightarrow} \alpha_1\alpha_2\alpha_3....\alpha_n = \alpha \quad (2.1)$$

Por consiguiente, $A \stackrel{*}{\Longrightarrow} \alpha$. Adviértase que (2.1) es sólo una de muchas derivaciones posibles que se pueden producir a partir de un árbol de análisis sintáctico dado.

Ahora supóngase que $A \stackrel{*}{\Longrightarrow} \alpha$. Se debe mostrar que existe un árbol A con producto α. Si $A \stackrel{*}{\Longrightarrow} \alpha$ mediante un solo paso, entonces $A \rightarrow \alpha$ es una producción de P, y existe un árbol con producto α, de la forma que se presenta en la fig. 2.3.

Supongase, ahora, que para cualquier variable A, si $A \stackrel{*}{\Longrightarrow} \alpha$ mediante una derivación con menos de k pasos, entonces existe un árbol A con producto α. Supongase que $A \stackrel{*}{\Longrightarrow} \alpha$ mediante una derivación de k pasos. Sea el primer paso $A \rightarrow X_1X_2...X_n$, cualquier símbolo de α debe ser uno de $X_1X_2...X_n$ o ser derivado de uno de estos. También, la porción de α derivada de X_1, debe estar a la izquierda de los símbolos derivados de X_j, si $i < j$. Por tanto, se puede escribir α como $\alpha_1\alpha_2\alpha_3....\alpha_n$, en donde, para i entre 1 y n,

1. $\alpha_i = X_i$ Si X_i es un terminal y
2. $X_i \stackrel{*}{\Longrightarrow} \alpha_i$ Si X_i es una Variable.

Si X_i es una variable, entonces la derivación de α_i de X_i debe tomar menos de k pasos, ya que la derivación completa $A \stackrel{*}{\Longrightarrow} \alpha$ toma k pasos y definitivamente el primero no es parte de la derivación $X_i \stackrel{*}{\Longrightarrow} \alpha_i$. Por tanto, por la hipótesis inductiva, para cada X_i que sea una variable, existe un árbol X_i con producto α_i. Sea este árbol T_i.

Se comienza construyendo un árbol A con n terminales etiquetados $X_1, X_2, ..., X_n$ y sin ningún otro vértice. Este árbol se muestra en la Fig. 2.4.(a). Cada vértice con etiqueta X_i , en donde X_i no es terminal, se reemplaza por el árbol T_i. Si X_i es terminal, no se hacen sustituciones. En la Fig. 2.4 (b) aparece un ejemplo. El producto de este árbol es α.

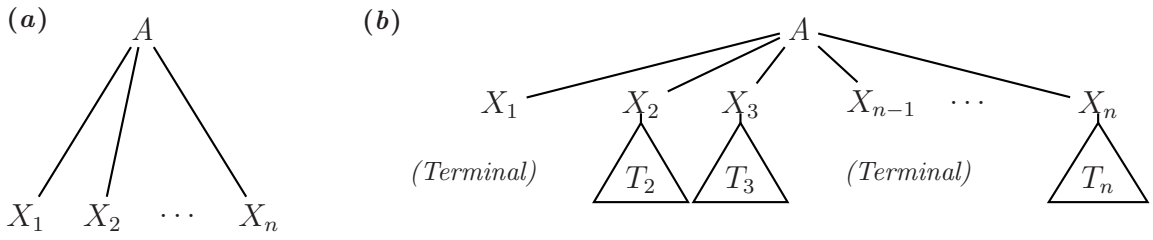


Fig. 2.4 Árboles de derivación

Ejemplo 2.10

Considerese la derivación $S \overset{*}{\Longrightarrow} aabbaa$ del ejemplo 2.8. El primer paso es $S \rightarrow aAS$. Si seguimos la derivación, se puede ver que A es sustituida por SbA, después por abA y finalmente por $abba$. La Fig. 2.2. representa un árbol de análisis sintáctico para esta derivación. El único símbolo derivado de S en aAS es a (Esta sustitución representa el último paso). La Fig. 2.5 (a) es un árbol para la última derivación.

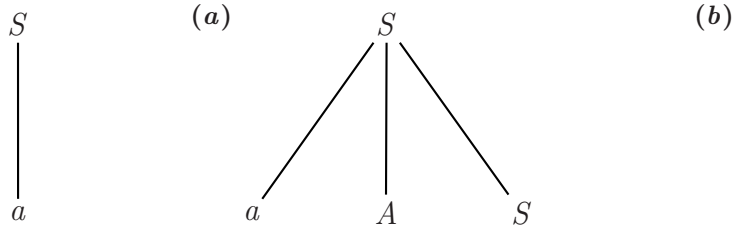


Fig. 2.5 Árboles de derivación

La Fig. 2.5 muestra el árbol de derivación para $S \rightarrow aAS$. Se sitúan los vértices con etiqueta A, de la Fig. 2.5 (b), por el árbol de la Fig. 2.2 y el vértice con etiqueta S, de la Fig. 2.5 (b), con el árbol de la Fig. 2.5 (a), se obtiene el árbol de la Fig. 2.1 (a), cuyo producto es $aabbaa$.

2.3.3 Derivación a la extrema izquierda y a la extrema derecha; ambigüedad

- Si a cada paso de una derivación se aplica una producción a la variable que se encuentra más a la izquierda, entonces la derivación es extrema izquierda.

- Si a cada paso de una derivación se aplica una producción a la variable que se encuentra más a la derecha, entonces la derivación es extrema derecha.

Si w está en $L(G)$ para la gramática incontextual G, entonces w tiene al menos un árbol de análisis sintáctico, y correspondiendo a un árbol de análisis sintáctico partícular, w tiene una derivación izquierda y una derecha únicas.

Ejemplo 2.11:
La derivación extrema izquierda correspondiente al árbol de la Fig. 2.1 es
$S \Longrightarrow aAS \Longrightarrow aSbAS \Longrightarrow aabAS \Longrightarrow aabbaS \Longrightarrow aabbaa$

La correspondiente derivación derecha es
$S \Longrightarrow aAS \Longrightarrow aAa \Longrightarrow aSbAa \Longrightarrow aSbbaa \Longrightarrow aabbaa.$

Una gramática incontextual G en la que alguna palabra tenga dos árboles de análisis sintáctico diferentes se dice que es ambigüa. Una definición equivalente de ambigüedad está dada por el hecho de que alguna palabra tenga más de una derivación extrema izquierda o más de una extrema derecha.

2.4 Simplificación de gramáticas incontextuales

Los teoremas que siguen a continuación permiten obtener la simplificación de las gramáticas incontextuales (GI) a una forma estándar denominada *Forma Normal de Chomsky* (FNC).

Definición 5: Si L es un lenguaje libre de contexto entonces puede ser generado por una gramática incontextual G que posee las propiedades siguientes.

1. Cada variable y cada terminal de G aparece en la derivación de alguna palabra de L.

2. No existen producciones de la forma $A \rightarrow B$ en las que A y B sean variables.

Si λ no está en L, es necesario que no haya producciones de la forma $A \rightarrow \lambda$. De hecho si λ no está en L, podemos necesitar que cada producción de G sea de una de las formas $A \rightarrow BC$, o, $A \rightarrow a$ en donde A, B y C son variables cualesquiera y a es un terminal cualquiera ***(Forma Normal de Chomsky)***[2].

2.4.1 Símbolos inútiles

Sea $G = (V, T, P, S)$ una gramática. Un símbolo $\boldsymbol{X}$ es ***útil*** si existe una derivación $S \stackrel{*}{\Longrightarrow} \alpha\, X\, \beta \stackrel{*}{\Longrightarrow} w$ para algunas α, β y w, en las que w está en T^*. De otra manera $\boldsymbol{X}$ es ***inútil***.

Lema 2.1: Dada una gramática incontextual $G = (V, T, P, S)$, con $L(G) \neq \emptyset$, podemos efectivamente encontrar una gramática incontextual $G' = (V', T, P', S)$ tal que para cada A en V' exista alguna w en T^* para la cual $A \stackrel{*}{\Longrightarrow} w$ [HU79].

Demostración Se construirá G' a partir de G a través del procedimiento que se detallará a continuación. Cada variable A con producción $A \rightarrow w$ en P pertenece a V'. Si $A \rightarrow X_1X_2...X_n$ es una producción, en la que cada X_i es un terminal o una variable que ya está situada en V', entonces una cadena terminal puede derivarse de A mediante una derivación que comience $A \Longrightarrow X_1X_2...X_n$ y, por consiguiente, A pertenece a V'. El conjunto V' puede calcularse mediante un sencillo algoritmo iterativo. P' es el conjunto de todas las producciones cuyos símbolos están en $V' \cup T$.

El algoritmo 2.1 encuentra todas las variables A que pertenecen a V'. Seguramente, si A se agrega a **NEWV** en la línea (2) o en la (5), entonces A deriva una cadena terminal. Para mostrar que **NEWV** no es demasiado pequeña, deberemos mostrar que si A deriva una cadena terminal w, entonces A se añade finalmente a **NEWV**. Hacemos esto por inducción sobre la longitud de la derivación $A \stackrel{*}{\Longrightarrow} w$.

Base Si la longitud es uno, entonces $A \rightarrow w$ es una producción y A se agrega a

[2]La definición formal de FNC se dará en la sección 2.6

NEWV en el paso (2).

Inducción Sea $A \stackrel{*}{\Longrightarrow} X_1X_2...X_n \stackrel{*}{\Longrightarrow} w$ mediante una derivación en k pasos. Entonces podemos escribir $w = w_1w_2...w_n$, en donde $X_i \stackrel{*}{\Longrightarrow} w_i$ para $1 \leq i \leq n$, mediante una derivación con menos de k pasos. Por la hipótesis de inducción,

Algoritmo 2.1 Cálculo de V' [HU79]

Entrada : $G = (V, T, S, P)$
Salida : V' **begin**
1. OLDV : $= \oslash$;
2. NEWV : $=\{A|A \rightarrow w$ para alguna w en $T^*\}$
3. **While OLDV** $\neq$ **NEWV do**
   begin
4. OLDV = **NEWV;**
5. **NEWV** : = OLDV $\cup \{A|A \rightarrow \alpha$ para alguna α en $(T \cup OLDV)^*\}$
   end;
6. $\boldsymbol{V'}$: = **NEWV**

end

aquellas X_i, que son variables, finalmente se agregan a **NEWV**. En la prueba del ciclo *while* en la línea (3), inmediatamente después que la última X_i se ha añadido a **NEWV**, no podemos tener **NEWV** = **OLDV** por que la última de estas X_i no está en **OLDV**. Por tanto el ciclo *while* se repite al menos una vez más, y A será agregada a **NEWV** en la línea (5).

Tómese V' como el conjunto calculado en la línea (6) y P' como todas las producciones cuyos símbolos están en $V' \cup T$. Seguramente $G' = (V', T, P', S)$ satisface la propiedad de que si A está en V', entonces $A \stackrel{*}{\Longrightarrow} w$ para alguna w. También, puesto que cada derivación de G' es una derivación de G, sabemos que $L(G') \subseteq L(G)$. Pero si existe

alguna w en $L(G)$ que no esté en $L(G^{'})$, entonces cualquier derivación de w en G debe involucrar una variable que esté en $V - V^{'}$ o una producción $P - P^{'}$ (lo que implica que existe una variable en $V - V^{'}$ que ha sido utilizada). Pero entonces existe una variable $V - V^{'}$ que deriva una cadena terminal, lo que significa una contradicción.

Considerando el ejemplo 2.7: Sea $G = (V, T, P, S)$, $V = \{\text{S, A, B}\}$, $T = \{a,\ b\}$ y P consiste en las producciones:

$$\begin{array}{ll} S \to aB & A \to bAA \\ S \to bA & B \to b \\ A \to a & B \to bS \\ A \to aS & B \to aBB \end{array}$$

Aplicando el Algoritmo tenemos que

begin
OLDV : = $\oslash$;
NEWV : =$\{B,\ A\ \}$;
While OLDV $\neq$ **NEWV do**
begin
OLDV = $\{B,\ A\ \}$
NEWV : = $\{B,\ A,\ S\}$;
While OLDV $\neq$ **NEWV do**
begin
OLDV = $\{B,\ A,\ S\}$;
NEWV : = $\{B,\ A,\ S\}$;

end;
$\boldsymbol{V^{'}}$: = **NEWV**
end

Lema 2.2: Dada una gramática incontextual $G = (V, T, P, S)$ se puede encontrar efectivamente otra $G' = (V', T', P', S)$ equivalente con G tal que para cada X en $V' \cup T'$ existan α y β en $(V' \cup T')^*$ para las que $S \stackrel{*}{\Longrightarrow} \alpha X \beta$ [HU79].

Demostración El conjunto $V' \cup T'$ de símbolos que aparecen en las formas oracionales de G se construye mediante un algoritmo iterativo. Colóquese S en V'. Si A está colocado en V' y $A \rightarrow \alpha_1 \mid \alpha_2 \mid ... \mid \alpha_n$, entonces agreguense todas las variables de $\alpha_1, \alpha_2, ..., \alpha_n$, al conjunto V'y todas las terminales de $\alpha_1, \alpha_2, ..., \alpha_n$ a T'. P' es el conjunto de producciones de P que contienen solo símbolos de $V' \cup T'$.

Aplicando, primero, el Lema 2.1 y, después, el Lema 2.2, se puede convertir una gramática a una equivalente que no contenga símbolos inútiles. Es interesante notar que aplicando el Lema 2.2, primero, y el Lema 2.1 después, talvez no se eliminen los símbolos inútiles.

Teorema 2.2: Cada lenguaje libre de contexto no vacío es generado por una gramática incontextual que no posee símbolos inútiles [HU79].

Ejemplo 2.12: Considérese una gramática con las siguientes producciones

$$\begin{aligned} &S \rightarrow AB \mid a \\ &A \rightarrow a \end{aligned} \qquad (2.2)$$

Aplicando Lema 2.1, se encuentra que ninguna cadena terminal es derivable de B. Por tanto, eliminamos B y la producción $S \rightarrow AB$. Después aplicando el Lema 2.2 a la gramática

$$\begin{aligned} &S \rightarrow a \\ &A \rightarrow a \end{aligned} \qquad (2.3)$$

Se encuentra que sólo S y a aparecen en las formas oracionales. Por consiguiente ($\{S\}$, {a}, $\{S \rightarrow a\}$, S) es una gramática equivalente que no tiene símbolos inútiles.

Supóngase que primero aplicamos el Lema 2.2 a (2.2). Podríamos encontrar que todos los símbolos aparecen en formas oracionales. Entonces aplicando el Lema 3.1 se quedaría con (3.3) que tiene un símbolo inútil, A.

2.4.2 Producciones λ

Una cadena puede tener infinitas derivaciones si dejamos las derivaciones λ, por lo tanto es aconsejable tratar de eliminarlas. Así toda cadena tendrá un número finito de derivaciones posibles.

- Si λ está en $L(G)$, no se pueden eliminar todas las producciones λ de G.
- Si λ no está en $L(G)$ se puede eliminar las producciones λ y para esto se determina para cada variable A, si $A \stackrel{*}{\Longrightarrow} \lambda$. Si esto sucede, se dice que A es *anulable*.
- Se puede sustituir cada producción $B \rightarrow X_1X_2X_3...X_n$ por todas las producciones formadas mediante la eliminación de algún subconjunto de aquellas X_i que son anulables, pero no se incluye $B \rightarrow \lambda$, aún si todas las X_i son anulables.

Teorema 2.3: Si $L = L(G)$ para alguna gramática incontextual $G = (V, T, P, S)$, entonces $L - \{\lambda\}$ es $L(G')$ para una gramática incontextual G' que no tiene símbolos inútiles o producciones λ [HU79].

Demostración Se pueden determinar los símbolos anulables de G a través del siguiente algoritmo iterativo. Para empezar, si $A \rightarrow \lambda$ es una producción, entonces A es anulable. En seguida, si $B \rightarrow \alpha$ es una producción y todos los símbolos de α se han encontrado anulables, entonces B es anulable. Se repite este proceso hasta que no se encuentren más símbolos anulables.

El conjunto de producciones de P' se construye de la manera siguiente. Si $A \rightarrow X_1X_2...X_n$ está en P, entonces agréguense todas las producciones $A \rightarrow \alpha_1\alpha_2...\alpha_n$ en P' donde

1. Si X_i no es anulable, entonces $\alpha_i = X_i$;

2. Si X_i es anulable, entonces α_i es X_i o λ;
3. no todas las α_i son λ.

Sea $G'' = (V, T, P', S)$. Se pide que para toda A en V y w en T^*, $A \underset{G'}{\overset{*}{\Longrightarrow}} w$ si y sólo si $w \neq \lambda$ y $A \underset{G''}{\overset{*}{\Longrightarrow}} w$.

Se demostrará primero que, *si para toda A en V y w en T^*, $A \underset{G'}{\overset{*}{\Longrightarrow}} w$ entonces $w \neq \lambda$ y $A \underset{G''}{\overset{*}{\Longrightarrow}} w$.*

Sea $G'' = (V, T, P', S)$. Se demostrará por inducción que $A \underset{G}{\overset{*}{\Longrightarrow}} w$. La *base*, $i = 1$, es inmediata, pues $A \to w$ debe ser una reducción de P como $w \neq \lambda$ también es una producción de P'. Según el paso inductivo, sea $i > 1$. Entonces $A \underset{G''}{\Longrightarrow} X_1X_2...X_n \underset{G''}{\overset{i-1}{\Longrightarrow}} w$. Escríbase $w = w_1w_2...w_n$ tal que para cada j, $X_j \underset{G'}{\overset{*}{\Longrightarrow}} w_j$ en menos de i pasos. Si $w_j \neq \lambda$ y X_j es una variable, entonces por la hipótesis de inducción tenemos que $X_j \underset{G}{\overset{*}{\Longrightarrow}} w_j$. Si $w_j = \lambda$, entonces X_j es anulable. Por tanto, $A \to \beta_1\beta_2...\beta_n$ es una producción de P', en donde $\beta_j = X_j$ si $w_j \neq \lambda$ y $\beta_j = \lambda$ si $w_j = \lambda$. Como $w \neq \lambda$, no todas las β_j son λ. De aquí que se tenga una derivación en G''.

$$A \Longrightarrow \beta_1\beta_2...\beta_n \overset{*}{\Longrightarrow} w_1\beta_2...\beta_n \overset{*}{\Longrightarrow} w_1w_2\beta_3...\beta_n \overset{*}{\Longrightarrow} ... \overset{*}{\Longrightarrow} w_1w_2...w_n = w$$

en G''.

Ahora, se demuestra que, *si $w \neq \lambda$ y $A \underset{G''}{\overset{*}{\Longrightarrow}} w$ entonces para toda A en V y w en T^*, $A \underset{G'}{\overset{*}{\Longrightarrow}} w$*

Supóngase que $A \underset{G''}{\overset{i}{\Longrightarrow}} w$. Seguramente $w \neq \lambda$, ya que G'' no tiene producciones λ. Mostramos por inducción en i que $A \underset{G}{\overset{*}{\Longrightarrow}} w$. Para la *base*, $i = 1$, obsérvese que $A \to w$ es una producción de P'. Debe existir una producción $A \to \alpha$ en P talque eliminando ciertos símbolos anulables de α se queda con w. Entonces existe una derivación $A \underset{G''}{\Longrightarrow} \alpha \underset{G''}{\overset{i}{\Longrightarrow}} w$, en la que la derivación $\alpha \overset{*}{\Longrightarrow} w$ implica la derivación de α a partir de los símbolos anulables de α que fueron eliminados para conseguir w.

Para el paso inductivo, hagase $i > 1$. Entonces $A \underset{G''}{\Longrightarrow} X_1X_2...X_n \overset{i-1}{\underset{G''}{\Longrightarrow}} w$. Debe existir alguna $A \to \beta$ en P tal que $X_1X_2...X_n$ se encuentra eliminando algunos símbolos anulables de β. Por consiguiente $A \overset{*}{\underset{G''}{\Longrightarrow}} X_1X_2...X_n$. Escríbase $w = w_1w_2...w_n$, tal que para toda j, $X_j \overset{*}{\underset{G}{\Longrightarrow}} w_j$ mediante menos de i pasos. Por la hipótesis de inducción, $X_j \overset{*}{\underset{G''}{\Longrightarrow}} w_j$ si X_j es una variable. Ciertamente, si X_j es un terminal entonces $X_j = w_j$ y $X_j \overset{*}{\underset{G}{\Longrightarrow}} w_j$ es trivialmente verdadera. En consecuencia $A \overset{*}{\underset{G}{\Longrightarrow}} w$.

El último paso de la demostración consiste en aplicar el Teorema 2.2 a G'' para producir G' sin símbolos inútiles. Ya que la construcción de los Lemas 2.1 y 2.2 no introducen ninguna producción, G' no tiene ni símbolos anulables ni inútiles. Aún más, $S \overset{*}{\underset{G''}{\Longrightarrow}} w$ si y sólo si $w \neq \lambda$ y $S \overset{*}{\underset{G''}{\Longrightarrow}} w$. Esto es, $L(G') = L(G) - \{\lambda\}$.

2.4.3 Producciones unitarias

Son producciones de la forma $A \to B$ cuyo lado derecho consiste en una sola variable. Todas las otras producciones, incluyendo las de la forma $A \to a$ y las producciones λ, son producciones ***no unitarias***.

Teorema 2.4 Cada Lenguaje libre de contexto sin λ puede definirse por una gramática que no tiene símbolos inútiles, producciones λ, o producciones unitarias [HU79].

Demostración Sea L un lenguaje libre de contexto sin λ y $L = L(G)$ para alguna $G = (V, T, P, S)$. Según el Teorema 2.3, supóngase que G no tiene producciones λ. Constrúyase un nuevo conjunto de producciones P' de P, mediante la inclusión, primero, de todas las producciones no unitarias de P. Entonces, supóngase que $A \overset{*}{\underset{G}{\Longrightarrow}} B$, para A y B de V. Agréguese a P' todas las producciones de la forma $A \to \alpha$, en donde $B \to \alpha$ es una producción no unitaria de P.

Observe que se puede probar de manera sencilla si $A \overset{*}{\underset{G}{\Longrightarrow}} B$, ya que G no tiene producciones λ, y si

$$A \underset{G}{\Longrightarrow} B_1 \underset{G}{\Longrightarrow} B_2 \underset{G}{\Longrightarrow} ... \underset{G}{\Longrightarrow} B_m \underset{G}{\Longrightarrow} B,$$

y algunas variables aparecen dos veces en la secuencia, se puede encontrar una secuencia más corta de producciones unitarias que traen como resultado que $A \overset{*}{\underset{G''}{\Longrightarrow}} B$. Por tanto es suficiente considerar aquellas secuencias de producciones unitarias que no repiten ninguna de las variables de G.

Ahora tenemos una gramática modificada, $G' = (V, T, P', S)$. Seguramente, si $A \to \alpha$ es una producción de P', entonces $A \overset{*}{\underset{G}{\Longrightarrow}} \alpha$. Por consiguiente, si existe una derivación de w en G', entonces existe una derivación extrema izquierda de w en G, digamos

$$S = \alpha_0 \underset{G}{\Longrightarrow} \alpha_1 \underset{G}{\Longrightarrow} ... \underset{G}{\Longrightarrow} \alpha_n = B.$$

Si, para $0 \leq i < n$, $\alpha_i \underset{G''}{\Longrightarrow} \alpha_{i+1}$ a través de una producción no unitaria, entonces $\alpha_i \underset{G''}{\Longrightarrow} \alpha_{i+1}$. Supóngase que $\alpha_i \underset{G''}{\Longrightarrow} \alpha_{i+1}$ a través de una producción unitaria, pero que $\alpha_{i-1} \underset{G''}{\Longrightarrow} \alpha_i$ mediante una producción no unitaria, o $i = 0$. También supóngase que $\alpha_{i+1} \underset{G''}{\Longrightarrow} \alpha_{i+2} \underset{G''}{\Longrightarrow} ... \underset{G''}{\Longrightarrow} \alpha_j$, todo mediante producciones unitarias, y $\alpha_j \underset{G''}{\Longrightarrow} \alpha_{j+1}$ mediante una producción no unitaria. Entonces $\alpha_i, \alpha_{i+1}, ..., \alpha_j$ son todas de la misma longitud, y como la derivación es extrema izquierda, el símbolo sustituido en cada una de éstas debe estar en la misma posición. Pero entonces $\alpha_i \underset{G''}{\Longrightarrow} \alpha_{j+1}$ mediante una de las producciones de $P' - P$. De aquí que $L(G') = L(G)$. Para completar la demostración, obsérvese que G' no tiene producciones unitarias o producciones λ. Si se utilizan los Lemas 2.1 y 2.2 para eliminar los símbolos inútiles, no se agrega ninguna producción, de modo que el resultado de aplicar la construcción de tales lemas a G' es una gramática que satisface el teorema.

2.5 Forma Normal de Chomsky

Teorema 2.5: (Forma Normal de Chomsky) Cualquier lenguaje libre de contexto sin λ, es generado por una gramática en la que todas las reglas de producción son de la forma $A \to BC$ o $A \to a$. Aquí, A, B y C son variables y a es una terminal [HU79].

Demostración: Sea $G_1 = (V, T, P, S)$ una gramática incontextual que genera un lenguaje que no contiene producciones λ. Ahora considérese la regla de producción

$p : A \to X_1X_2...X_m$, donde $m \geq 2$, $p \in P$. Si X_i es un terminal, supóngase que sea: a, se introduce una variable C_a y una producción $C_a \to a$; así que se sustituye X_i por C_a. Entonces tenemos una nueva variable (C_a) y una nueva regla de producción $(C_a \to a)$, así que se construye una nueva gramática con dos nuevos conjuntos de no terminales y de reglas de producción.

Sea G_2 esta nueva Gramática, $G_2 = \left(V', T, P', S\right)$. Si la derivación directa $\alpha \underset{G_1}{\Longrightarrow} \beta$, existe en la primera gramática, entonces en la segunda gramática también existe esta derivación con más derivaciones directas $\alpha \overset{*}{\underset{G_2}{\Longrightarrow}} \beta$. Por tanto $L(G_1) \subseteq L(G_2)$.

Ejemplo 2.13: Sea la gramática G= ($\{S, A, B\}, \{a, b\}, P, S$) y el conjunto de reglas P viene dado por: $S \to bA \mid aB$

$A \to bAA \mid aS \mid a$
$B \to aBB \mid bS \mid b$

Las únicas reglas de producciones que ya se encuentran en forma apropiada son $A \to a$ y $B \to b$. No existen producciones unitarias, así que se puede empezar sustituyendo las terminales de la derecha por variables, excepto en el caso de las producciones $A \to a$ y $B \to b$.

En $S \to bA$ se sustituye por $S \to C_bA$ donde C_b es un no terminal y se agrega la regla de producción $C_b \to b$.

De manera similar, en $A \to aS$ se sustituye por $A \to C_aS$ donde C_a es también un no terminal y se agrega la regla de producción $C_a \to a$.

Ahora $A \to bAA$ es sustituida por $A \to C_bD_1$ donde D_1 es un no terminal y se agrega la regla de producción $D_1 \to AA$. De manera similar, en $B \to aBB$ se sustituye por $B \to C_aD_2$ donde D_2 es un no terminal y se agrega la regla de producción $D_2 \to BB$.

Así se obtiene el nuevo conjunto de reglas de producción:

$$S \to C_b A \mid C_a B$$
$$A \to C_b D_1 \mid C_a S \mid a$$
$$B \to C_a D_2 \mid C_b S \mid b$$
$$D_1 \to AA$$
$$D_2 \to BB$$
$$C_a \to a$$
$$C_b \to b$$

Que representa una gramática en Forma Normal de Chomsky, es decir todas las reglas de producción parten de un símbolo no terminal a dos símbolos no terminales o un sólo símbolo terminal.

Capítulo 3

GRAMÁTICAS Y CADENAS

En esta capítulo se estudian dos algoritmos, El Algoritmo de Cocker-Younger-Kasami y el Algoritmo de Crespi - Reghizzi. El primero permite establecer si una cadena pertenece a una Gramática y el segundo permite determinar las reglas de producción para generar una cadena de la Gramática.

3.1 Pertenencia de una cadena a un lenguaje

Una cuestión esencial relacionada con las Gramáticas incontextuales (o Gramáticas libres de contexto) (GI) es cómo se pueden identificar cadenas que pertenecen al lenguaje generado por una GI G dada. El problema consiste en determinar si $x \in L(G)$, dada una cadena x y una GI G, o alternativamente probar que la cadena no se pueda generar por la gramática. La solución al problema consiste en encontrar una secuencia de derivaciones que permitan derivar x a partir del símbolo inicial de G utilizando reglas de la gramática.

Hay dos tipos principales de análisis: *bottom up* que empieza de la cadena x y se trata de construir un árbol que comienze en el símbolo inicial de G; y el *top down*, en la dirección opuesta. Se puede solucionar el problema con un coste lineal con la longitud de la cadena, pero restringiendo severamente la clase de gramáticas, y por tanto la capacidad expresiva de las mismas. Otra forma de solucionarlo se la trata en este trabajo con un coste cúbico con la longitud de la cadena utilizando gramáticas en FNC.

Una solución eficiente para abordar el análisis sintáctico consiste en utilizar algún

método tabular basado en Programación Dinámica. Estos métodos se basan en la construcción de una tabla de análisis tal que cada celda representa la solución a un determinado subproblema. Los métodos tabulares más conocidos son el algoritmo de Cocke-Younger-Kasami que opera con GI en FNC, y el algoritmo de Earley que permite trabajar con una GI cuyas reglas no es necesario que tengan ninguna particularidad. En esencia, ambos algoritmos son similares. En este trabajo sólo se hará uso del primero de ellos, por lo que se describirá éste únicamente.

El algoritmo de Cocke-Younger-Kasami se basa en la construcción de una tabla de análisis V de dimensión $|x| \times |x|$, tal que si $A \in V_{i,j}$, entonces $A \overset{*}{\Longrightarrow} x_i...x_j$. De esta forma $x \in L(G)$ si S pertenece a $V_{1,|x|}$ (ver el algoritmo en la figura 4.1).

El algoritmo opera analizando partes de la cadena cada vez de mayor longitud hasta incluir finalmente toda la cadena. Este tipo de análisis se conoce como análisis ascendente puesto que va considerando subárboles de análisis desde las hojas hacia la raíz. El coste computacional temporal del algoritmo de Cocke-Younger-Kasami es $O(|x|^3 |P|)$ mientras que el coste computacional espacial es $O(|x|^2 |N|)$.

1. **para** $i = 1$ hasta n **hacer**.
2. $\quad V_{i,1} = \{A \mid A \longrightarrow a \in P\}$.
3. **fin para**
4. **para** $j = 2$ hasta n **hacer**
5. **para** $i = 1$ hasta n-$j + 1$ **hacer**
6. $\quad V_{i,j} = \oslash$
7. $\quad$ **para** $k = 1$ hasta $j - 1$ **hacer**
8. $\quad\quad V_{i,j} = V_{i,j} \cup \{A \mid A \longrightarrow BC \in P,\ B \in V_{i,k}$ y $C \in V_{i+k,j-k}\}$.
9. $\quad$ **fin para**
10. **fin para**
11. **fin para**

figura 3.1. Algoritmo de Cocke-Younger-Kasami [BS00].

Ejemplo 3.1

Consideremos la siguiente gramática G en forma Normal de Chomsky:

G =(V, T, P, S), donde

T =$\{$a,b$\}$, $V = \{S, A, B, C\}$ y

P:

$$S \longrightarrow AB \mid BC$$
$$A \longrightarrow BA \mid a$$
$$B \longrightarrow CC \mid b$$
$$C \longrightarrow AB \mid a$$

La cadena de entrada *baaba*.

La tabla de análisis es mostrada en la figura 3.2, creada de la siguiente forma:

Los pasos 1 y 2 del algoritmo me permiten llenar las casillas $V_{1,i}$ con $i = 1, 2, 3, 4, 5$, de esta forma $B = V_{1,1} = V_{1,4}$, ya que B es la única variable que deriva en $b = x_1 = x_4$; y $A,C = V_{1,2} = V_{1,3} = V_{1,5}$, pues A y C son las variables que derivan en $a = x_2 = x_3 = x_5$.

Para $j \geq 2$ las líneas de la 4 a la 11 nos presentan dos ciclos **para**, anidados que permiten seguir llenando la tabla.

Una vez llena la tabla y teniendo en cuenta que:

Si $A \in V_{i,j}$, entonces $A \overset{*}{\Longrightarrow} x_i...x_j$

hacemos el análisis de ella.

Por ejemplo podemos afirmar que :

Como $C \in V_{2,4}$, entonces $C \overset{*}{\Longrightarrow} x_2x_3x_4$ o sea que $C \overset{*}{\Longrightarrow} aab$

Pero en especial como S pertenece a la última casilla de la tabla en este caso $V_{1,5}$, entonces $S \overset{*}{\Longrightarrow} x_1x_2x_3x_4x_5$, por tanto $S \overset{*}{\Longrightarrow} baaaba$

Así que la cadena $x = baaba$ si es posible que sea generada por la gramática G anteriormente mencionada.

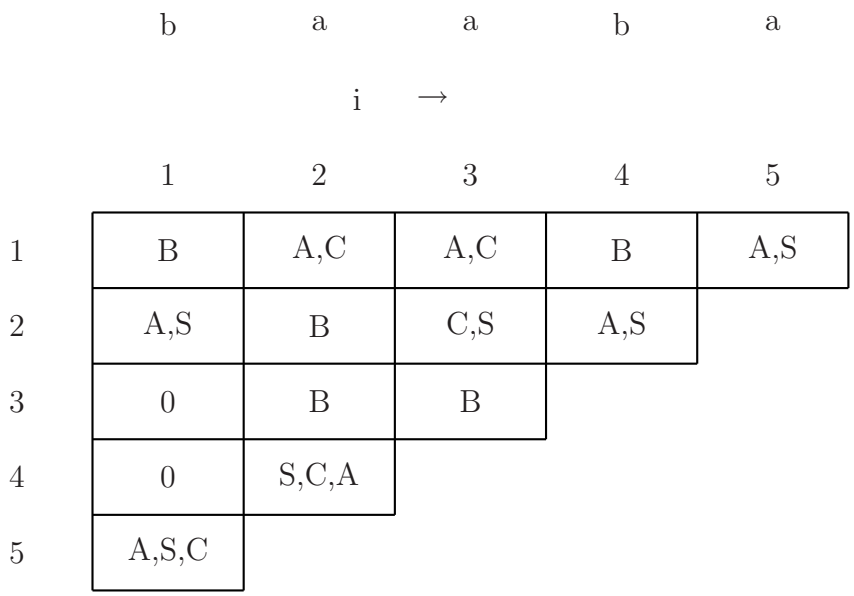


Figura 3.2 Tabla de Análisis

3.2 Construcción de gramáticas a partir de datos estructurados

Usando muestras estructuradas en el método, se asume información suplementaria concerniente a la muestra, concretamente el orden en que las letras que forman la cadena aparecen cuando está es generada por la gramática. Esto es equivalente a "estructurar" la muestra con la ayuda de un sistema de paréntesis de tal modo que el primer símbolo a ser analizado es el que esta encerrado por la cantidad más grande de paréntesis. Esta información que puede no ser necesaria en el caso de ciertos problemas reduce notablemente la escala de combinación del problema dando una indicación a priori de la estructura del árbol de derivación asociado con la cadena, pero las reglas de derivación correspondientes aún están por ser encontradas.

El algoritmo está basado en el concepto de perfil terminal de una cadena en $(V \cup T)^*$. Si α es una cadena y G una gramática $G = (V, T, P, S)$ entonces el perfil terminal de

α es definido como

$$P(\alpha) = ((L_t(\alpha), R_t(\alpha)))$$

donde

$$L_t(\alpha) = \left\{ a : \alpha \overset{*}{\Longrightarrow} Aa\gamma \right\}$$

$$R_t(\alpha) = \left\{ a : \alpha \overset{*}{\Longrightarrow} \gamma Aa \right\}$$

y $\quad a \in T \quad A \in V \cup \{\lambda\} \quad \gamma \in (V \cup T)^*$

Así que $L_t(\alpha)$ es el conjunto de elementos del alfabeto $\boldsymbol{T}$ que puede aparecer en posiciones $Aa\gamma$ o $a\gamma$ en las cadenas que pueden ser derivadas de α al aplicar las reglas de G. El algoritmo es el siguiente.

Algoritmo Crespi - Reghizzi

(a) Para cada cadena de la muestra estructurada se extrae una gramática libre de contexto canónico G_i, dada la información en la estructura.

(b) Para cada gramática G_i calcule el perfil terminal de la parte derecha de las reglas de G_i.

(c) Identifique los elementos de V asociados con la parte izquierda de las reglas de G_i, con las de la parte derecha que tengan el mismo perfil terminal.

(d) Combine todas las reglas obtenidas así del conjunto de cadenas y simplifique la gramática resultante.

El autor del algoritmo a mostrado que produce gramáticas algebraicas de forma especial, y no cubre todas las formas posibles [BS00].

Ejemplo 3.2:

```
(  (S
  (NP – SBJ
    (NP  (NNP Pierre)  (NNP Vinken)  )
    (ADJP
      (NP  (CD 61)  (NNS years)  )
      (JJ old)  ))
  (VP  (MD will)
    (VP  (VB join)
      (NP  (DT the)  (NN board))
      (PP – CLR(IN as)
    (NP  (DT a)  (JJ nonexecutive)  (NN director)  ))
      (NP – TMP  (NNP Nov.)  (CD 29)  )))))
```

Figura 3.3: La frase "Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29". Analizada y etiquetada en el proyecto Penn Treebank.

El primer paso a realizar es la construcción del árbol de derivación teniendo en cuenta la información que suministra la forma como está estructurada la muestra; es decir el sistema de paréntesis que presenta.

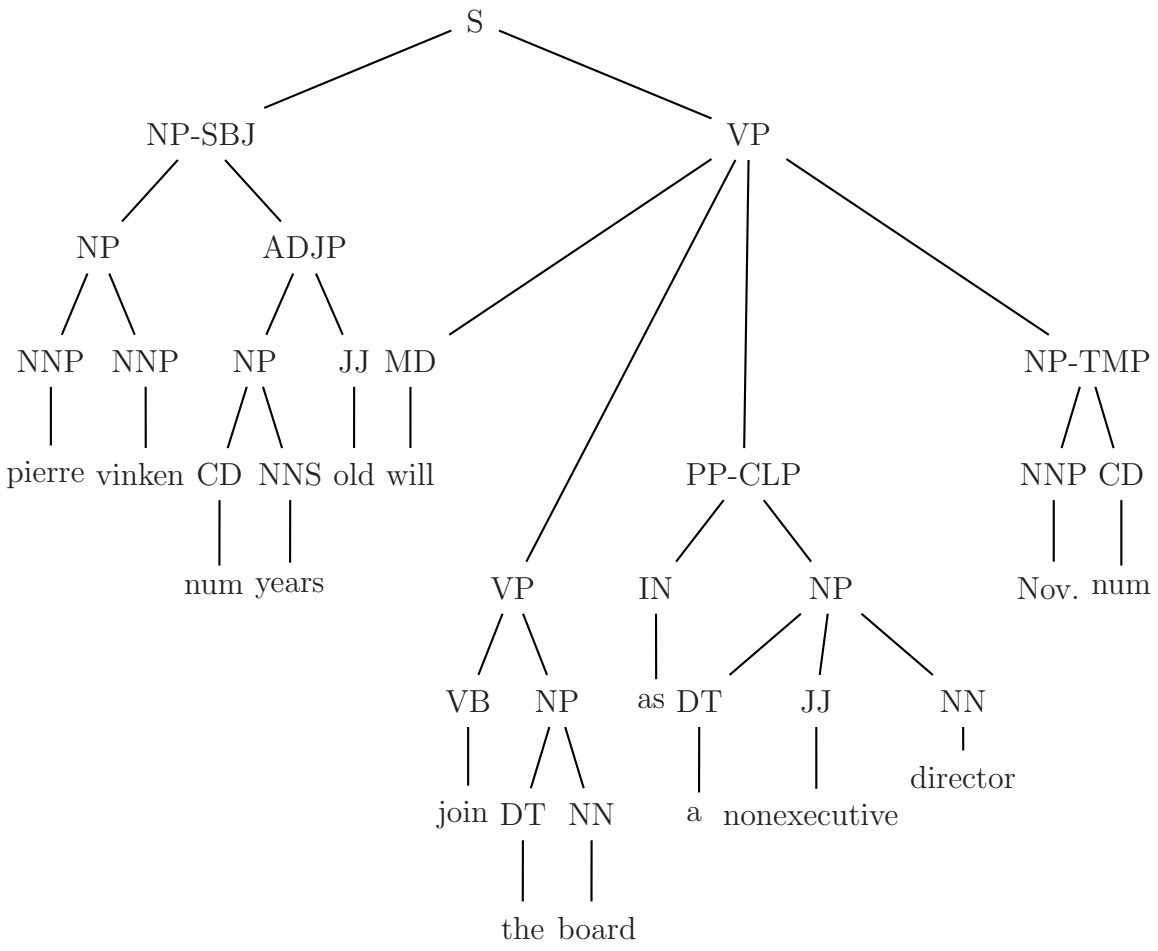


Fig. 3.4 Árbol de derivación

El segundo paso a seguir es la construcción de los perfiles terminales a través de la definición de perfil terminal que se dió anteriormente.

Perfiles Terminales

$\mathbf{P}((\texttt{NP})\,(\texttt{ADJP})) = (\{pierre, vikent\}, \{old, years\})$

$\mathbf{P}((\texttt{NNP})\,(\texttt{NNP})) = (\{pierre, vikent\}, \{pierre, vikent\})$

$\mathbf{P}((\texttt{CD})\,(\texttt{NNS})) = (\{num, years\}, \{num, years\})$

$\mathbf{P}((\texttt{DT})\,(\texttt{NN})) = (\{the, boar\}, \{the, boar\})$

$\mathbf{P}((\texttt{JJ})\,(\texttt{DT})\,(\texttt{NN})) = (\{nonexecutive, a\}, \{a, director\})$

$\mathbf{P}\,(pierre) = (\{pierre\}, \{pierre\})$

$\mathbf{P}\,(vinken) = (\{vinken\}, \{vinken\})$

$\mathbf{P}\,(num) = (\{num\}, \{num\})$

$\mathbf{P}\,(years) = (\{years\}, \{years\})$

$\mathbf{P}\,(old) = (\{old\}, \{old\})$

$\mathbf{P}((\texttt{NP})\,(\texttt{JJ})) = (\{num, years\}, \{old, years\})$

$\mathbf{P}\,(nonexecutive) = (\{nonexecutive\}, \{nonexecutive\})$

$\mathbf{P}((\texttt{MD})\,(\texttt{VP})\,(\texttt{PP}-\texttt{CLR})\,(\texttt{NP}-\texttt{TMP})) = (\{will\}, \{num, Nov\})$

$\mathbf{P}((\texttt{NP})\,(\texttt{VB})) = (\{the, board\}, \{join, board\})$

$\mathbf{P}\,(will) = (\{will\}, \{will\})$

$\mathbf{P}\,(the) = (\{the\}, \{the\})$

$\mathbf{P}\,(a) = (\{a\}, \{a\})$

$\mathbf{P}\,(board) = (\{board\}, \{board\})$

$\mathbf{P}\,(director) = (\{director\}, \{director\})$

$\mathbf{P}\,(join) = (\{join\}, \{join\})$

$\mathbf{P}((\texttt{IN})\,(\texttt{NP})) = (\{as, nonexecutive\}, \{director, a\})$

$\mathbf{P}\,(Nov.) = (\{Nov.\}, \{Nov.\})$

$\mathbf{P}((\texttt{NNP})\,(\texttt{CP})) = (\{Nov., num\}, \{Nov., num\})$

El tercer y último paso es la construcción de las reglas de producción que se infieren a partir de los perfiles terminales.

Reglas De Producción Inferidas

$\texttt{S} \rightarrow (\texttt{NP} - \texttt{SBJ})\,(\texttt{VP})$

$\texttt{NP} - \texttt{SBJ} \rightarrow (\texttt{NP})\,(\texttt{ADJP})$

$\texttt{NP} \rightarrow (\texttt{NNP})\,(\texttt{NNP}) \mid (\texttt{CD})\,(\texttt{NNS}) \mid (\texttt{DT})\,(\texttt{NN}) \mid (\texttt{JJ})\,(\texttt{DT})\,(\texttt{NN})$

$\texttt{NNP} \rightarrow (pierre) \mid (vinken)$

$\texttt{CD} \rightarrow (num)$

$\texttt{NNS} \rightarrow (years)$

$\texttt{JJ} \rightarrow (old) \mid (nonexecutive)$

$\texttt{ADJP} \rightarrow (\texttt{NP}) \mid (\texttt{JJ})$

$\texttt{VP} \rightarrow (\texttt{MD})\,(\texttt{VP})\,(\texttt{PP} - \texttt{CLR})\,(\texttt{NP} - \texttt{TMP}) \mid (\texttt{NP})\,(\texttt{VB})$

$\texttt{MD} \rightarrow (will)$

$\texttt{DT} \rightarrow (the) \mid (a)$

$\texttt{NN} \rightarrow (board) \mid (director)$

$\texttt{VB} \rightarrow (join)$

$\texttt{IN} \rightarrow (as)$

$\texttt{PP} - \texttt{CLR} \rightarrow (\texttt{IN})\,(\texttt{NP})$

$\texttt{NNP} \rightarrow (Nov)$

$\texttt{NP} - \texttt{TMP} \rightarrow (\texttt{NNP})\,(\texttt{CD})$

Capítulo 4

ESTIMACIÓN DE LAS GIP

En este capítulo se presentan algunas definiciones sobre lenguajes y gramáticas probabilísticas en el marco de Teoría de Lenguajes Probabilísticos y la introducción de nociones para realizar el análisis sintáctico probabilístico de una cadena. Además se presentan dos de los algoritmos clásicos de estimación de las Gramáticas Incontextuales Probabilísticas.

4.1 Lenguajes y gramáticas formales probabilísticas

Definición 1 : Un lenguaje probabilístico sobre un alfabeto $\sum$ es un par (L, Φ) , donde L es un lenguaje formal y Φ es una medida de probabilidad sobre $\sum^*$, talque $\Phi(x) > 0$ si $x \in L$

Definición 2 : Una Gramática Incontextual Probabilística (GIP) G_p es un par (G, p) tal que G es una Gramática Libre de Contexto, denominada en este caso gramática característica, y p una función definida sobre el conjunto de producciones con rango $[0, 1]$ y con la propiedad:
$\forall A \in V, \sum_{(A \to \alpha) \in \Gamma_A} p(A \to \alpha) = 1$ Donde Γ_A representa el conjunto de reglas de la gramática cuyo antecedente es A.

Definición 3 : Dada una GIP G_p. La probabilidad de una derivación de la cadena

$x \in \sum^*$ se define como:

$$\Pr(x, d_x \mid G_p) = \prod_{(A\to\alpha)\in P} p(A \to \alpha)^{N(A\to\alpha, d_x)}$$

Donde $N(A \to \alpha, d_x)$ representa el número de veces que la regla $A \to \alpha$ ha aparecido en la derivación d_x.

Definición 4 : Dada una GIP G_p. La probabilidad de la cadena $x \in \sum^*$ se define como:

$$\Pr(x \mid G_p) = \sum_{d_x \in D_x} \Pr(x, d_x \mid G_p)$$

Donde D_x denota el conjunto de todas las derivaciones diferentes de la cadena x.

Dada una Gramática Libre de Contexto $G_p = (G, p)$ se puede definir una Gramática Libre de Contexto Probabilística $G_p^{'}$ cuya gramática característica está en Forma Normal de Chomsky y además $\forall x \in L(G)$ se cumple que $\Pr(x \mid G_p) = \Pr\left(x \mid G_p^{'}\right)$.

Definición 5 : Dada una GIP G_p. Se llamará probabilidad de la mejor derivación de la cadena x a:

$$\hat{\Pr}(x \mid G_p) = \max_{d_x \in D_x} \Pr(x, d_x \mid G_p)$$

Definición 6 : Dada una GIP G_p. Se llamará derivación más probable o mejor derivación a:

$$\hat{d_x} = arg \max_{d_x \in D_x} \Pr(x, d_x \mid G_p)$$

Definición 7 : El lenguaje generado por una Gramática Libre de Contexto Probabilística G_p es: $L(G_p) = \{x \in L(G) \,|Pr(x|G_p) > 0\}$

Dado un lenguaje probabilístico cualquiera, (L, Φ) donde L es un lenguaje libre de contexto, puede pensarse que siempre es posible encontrar una Gramática Libre de Contexto Probabilística tal que $G_p = (G, p)$ y $L = L(G)$, Φ se calcula en términos de la definición 3. Sin embargo, no todo lenguaje probabilístico en el cual L es un lenguaje

Libre de Contexto puede ser representado por una Gramática Libre de Contexto Probabilística como se muestra en el siguiente teorema:

Teorema 4.1 : Dado el lenguaje libre de contexto $L = \{a^n b^n \mid n \geq 0\}$, y la función $\Phi(a^n b^n) = \frac{1}{en!}$; no existe ninguna Gramática Libre de Contexto $G_p = (G, p)$ que represente el lenguaje probabilístico (L, Φ) [SJA99].

Demostración: Sea el lenguaje probabilístico (L, Φ). Además, $\Phi(a^n b^n) = \frac{1}{en!}$ crece de forma inversa a una función exponencial y la función $\Pr(x \mid G_p)$ con $x \in L$ crece de forma inversa a una función polinómica.

Esto es $\Pr(x \mid G_p) = \sum_{d_x \in D_x} \Pr(x, d_x \mid G_p)$ donde cada regla de producción puede verse como una variable; $\Pr(x \mid G_p)$ se ve como un polinomio cuya variable es cada una de las reglas de producción.

Ya que ningún polinomio se puede aproximar a una función que crece exponencialmente, entonces sus funciones inversas respectivas tampoco se pueden aproximar, por tanto Φ no puede ser estimada por G_p.

De forma inversa podría pensarse que cualquier Gramática Libre de Contexto Probabilística $G_p = (G, p)$ es capaz de generar un lenguaje probabilístico (L, Φ) donde $L = L(G)$ y $\Phi(x) = \Pr(x \mid G_p)$. Sin embargo esto tampoco es necesariamente cierto ya que $\sum_{x \in L(G)} \Phi(x) = 1$.

Definición 8 : Una gramática incontextual $G_p = (G, p)$ es consistente si y solo si:

$$\sum_{x \in L(G)} \Pr(x \mid G_p) = 1$$

En cualquier otro caso la gramática no es consistente.

Definición 9 : Dada una gramática incontextual $G_p = (G, p)$ consistente, el par $(L(G), p)$ es un lenguaje libre de contexto probabilístico, donde p es una función de

probabilidad computada en términos de la expresión de la definición 4.

4.2 Análisis sintáctico probabilístico de una cadena

El análisis sintáctico probabilístico trata el problema de determinar si $\Pr(x, d_x \mid G_p) > 0$, para lo cual hay que encontrar al menos una derivación de la cadena cuya probabilidad sea mayor que cero [SJA99].

Los algoritmos más utilizados para el análisis sintáctico de una cadena son tres algoritmos que resuelven por Programación Dinámica este problema en un tiempo polinómico. Para ello consideraremos que la GIP está en Forma Normal de Chomsky.

4.2.1 Algoritmo Inside

El **algoritmo Inside** permite determinar si $\Pr(x, d_x \mid G_p) > 0$ calculando la probabilidad de la cadena a partir de todas las posibles derivaciones. El algoritmo Inside es un algoritmo basado en un esquema de Programación Dinámica análogo al algoritmo de Cocke-Younger-Kasami.

Se basa en la definición de la probabilidad de que la sub-cadena $x_{i,}...,x_j$ sea generada a partir de A:
$e(A\langle i,j\rangle) = Pr\left(A \overset{*}{\Rightarrow} x_i...x_j \mid G_p\right)$,
Esta probabilidad puede ser evaluada eficientemente mediante el esquema recursivo, $\forall A \in N$, como:

$$e(A\langle i,i\rangle) = p(A \to x_i) \qquad 1 \leq i \leq |x|$$

$$e(A\langle i,j\rangle) = \sum_{B,C \in N} p(A \to BC) \sum_{k=i} j - 1 e(B\langle i,k\rangle)\, e(C\langle k+1,j\rangle)$$
$$1 \leq i \leq j \leq |x|$$

De esta forma, $Pr(x|G_p) = e(S\langle 1,|x|\rangle)$

4.2.2 Algoritmo Outside

De manera análoga al algoritmo Inside y el algoritmo Outside permiten determinar si una cadena x es generada por una GIP calculando la probabilidad de la misma a partir de todas las posibles derivaciones.

En el algoritmo Outside se define la probabilidad de que a partir del axioma inicial se genere la subcadena $x_{1,}...,x_{i-1}$ a continuación el no terminal A y a continuación la subcadena $x_{j,}...,x_{|x|}$, $\forall A \in N$
$f(A < i,j >) =$Pr $(S \Rightarrow x_1...x_{i-1}Ax_{j+1}...x_x \mid G_p)$

$$f(A < i, \mid x \mid >) = \begin{cases} 1, & si \quad A = S \\ 0, & si \quad A \neq S \end{cases}$$

$$f(A < i,j >) = \sum_{B,C \in N} (p(B \longrightarrow CA) \sum_{k=1}^{i-1} f(B < k,j >)e(C < k,i-1 >)$$
$$+p(B \longrightarrow AC) \sum_{k=j+1}^{|x|} f(B < i,k >)e(C < j+1,k >))$$
$$1 \leq i \leq j \leq |x|$$

De esta forma, $Pr\,(x|G_p) = \sum_{A \in N} f(A < i,i >)p(A \longrightarrow x_i)$, $1 \leq i \leq \mid x \mid$

4.2.3 Algoritmo de Viterbi

Otra posibilidad para determinar si $\Pr\,(x, d_x \mid G_p) > 0$ consiste en encontrar al menos una derivación cuya probabilidad sea mayor que cero [SJA99]. El siguiente algoritmo permite calcular la derivación de la cadena cuya probabilidad es máxima.
El cálculo de esta probabilidad está basado en la definición de la probabilidad de la mejor derivación que genera la subcadena $x_{i,}...,x_j$ a partir de A:
$\hat{e}\,(A\,\langle i,j\rangle) = \hat{Pr}\left(A \stackrel{*}{\Rightarrow} x_i...x_j|G_p\right)$

Esta expresión puede ser calculada eficientemente mediante la aplicación de la siguiente función recursiva: $\forall A \in N$
$\hat{e}\,(A\,\langle i,i\rangle) = p\,(A \rightarrow x_i)$

$\hat{e}\left(A\left\langle i,j\right\rangle\right) = \underset{B,C\in N}{\text{máx}}\, p\left(A \to BC\right) \underset{k=1,\ldots j-1}{\text{máx}}\, \hat{e}\left(B\left\langle i,k\right\rangle\right)\hat{e}\left(C\left\langle k+1,j\right\rangle\right)$
$1 \leq i \leq j \leq |x|$
Por tanto, $\hat{Pr}\left(x|G_p\right) = \hat{e}\left(S\left\langle 1,|x|\right\rangle\right)$

Este algoritmo es similar al algoritmo de Cocke-Younger-Kasami.

Ejemplo 4.1:

A continuación se presenta un ejemplo para comprender mejor la implementación de estos algoritmos.

Si tenemos $G = (V, T, P, S)$ una gramática incontextual en Forma Normal de Chomsky, donde $V = \{S, A, B, C\}$, $T = \{a, b\}$, y el conjunto de reglas P, con sus respectivas probabilidades entre paréntesis, viene dado por:

$S \to AB$	(0,25)
$S \to BC$	(0,75)
$A \to BA$	(0,5)
$A \to a$	(0,5)
$B \to CC$	(0,1)
$S \to b$	(0,9)
$C \to AB$	(0,2)
$C \to a$	(0,8)

Donde la frase, $s = bab$
se deriva mediante, $S \to AB \to BAB \to baB \to bab$
Tiene los siguientes cálculos en el algoritmo de análisis.

Algoritmo Inside:

$e\left(S\left\langle 1,3\right\rangle\right) = \sum_{A,B} p\left(S \to AB\right) \cdot \sum 2k = 1e\left(A\left\langle 1,k\right\rangle\right) e\left(B\left\langle k+1,3\right\rangle\right)$
Donde en la última sumatoria si $k = 1$ tenemos:

$e\left(A\left\langle 1,1\right\rangle\right)e\left(B\left\langle 2,3\right\rangle\right)$ debemos encontrar este valor,

$e\left(B\left\langle 2,3\right\rangle\right)=p\left(B\rightarrow CC\right)\cdot e\left(C\left\langle 2,2\right\rangle\right)\cdot e\left(C\left\langle 3,3\right\rangle\right)$

pero $e\left(A\left\langle 1,1\right\rangle\right)=p\left(A\rightarrow b\right)=0$

$e\left(C\left\langle 2,2\right\rangle\right)=p\left(C\rightarrow a\right)=0{,}8$

$e\left(C\left\langle 3,3\right\rangle\right)=p\left(C\rightarrow b\right)=0$

y ahora si $k=2$ tenemos:

$e\left(A\left\langle 1,2\right\rangle\right)e\left(B\left\langle 3,3\right\rangle\right)$ debemos encontrar este último valor,

$e\left(A\left\langle 1,2\right\rangle\right)=p\left(A\rightarrow BA\right).e\left(B\left\langle 1,1\right\rangle\right)\cdot e\left(A\left\langle 2,2\right\rangle\right)$

pero,$e\left(B\left\langle 3,3\right\rangle\right)=p\left(B\rightarrow b\right)=0{,}9$

$e\left(B\left\langle 1,1\right\rangle\right)=p\left(B\rightarrow b\right)=0{,}9$

$e\left(A\left\langle 2,2\right\rangle\right)=p\left(A\rightarrow a\right)=0{,}5$

y $p\left(A\rightarrow BA\right)=0{,}5$

Entonces, $e\left(A\left\langle 1,1\right\rangle\right)e\left(B\left\langle 2,3\right\rangle\right)=0$

y $e\left(A\left\langle 1,2\right\rangle\right)e\left(B\left\langle 3,3\right\rangle\right)=p\left(A\rightarrow BA\right).p\left(B\rightarrow b\right).p\left(B\rightarrow b\right).p\left(B\rightarrow b\right)$

$0{,}9\cdot 0{,}5\cdot 0{,}9\cdot 0{,}9=0{,}3645$

este último valor multiplicado por $p\left(S\rightarrow AB\right)=0{,}25$ nos da como resultado:

$$0{,}091125 \text{ así que } e\left(S\left\langle 1,3\right\rangle\right)=0{,}091125>0$$

4.3 Estimación de las GIP

A continuación se presentan dos de los algoritmos principales de estimación de las GIP. Uno de ellos optimiza la función de verosimilitud de la muestra, y el otro optimiza la verosimilitud de la mejor derivación de la muestra.

4.3.1 Estimación probabilística de las GIP

El problema de estimación de las GIP puede enunciarse en los siguientes términos: Sea (L,ϕ) un lenguaje probabilístico sobre un alfabeto $\sum$, tal que $L \subseteq L(G)$ para una GIP G dada, y sea ϕ una función de probabilidad, en general desconocida. Dada una muestra Ω del lenguaje, que se asume refleja la distribución (desconocida) de ϕ, el problema consiste en determinar las probabilidades de las reglas de la GIP a partir de dicha muestra para representar dicha función. Para abordar este problema es necesario asumir que la función ϕ puede representarse por medio de una GIP [SJA99]. Por tanto, dada $G_p = (G, p)$ y Ω, se desea encontrar el conjunto de parámetros de la GIP que hacen que la distribución de probabilidad que ésta define sobre L se ajuste lo máximo posible a la distribución desconocida ϕ y representada por Ω, se pretende obtener el conjunto de parámetros tal que:

$$\text{p'} = \arg \max_{p} f_p(\Omega) \qquad (4.1)$$

Donde p es el conjunto de probabilidades de la GIP, y f_p es la función criterio a optimizar dependiente de la muestra, y definida en términos del conjunto de probabilidades. Así pues para abordar el problema de estimación debemos definir dicha función criterio, y algún método de optimización que nos permita obtener p'.

Respecto al método de optimización, en este trabajo se van ha considerar algoritmos de maximización en el marco de las transformaciones crecientes .

El siguiente teorema proporciona un método para obtener un máximo local de un polinomio de varias variables, se denomina teorema de transformación creciente, porque a cada punto de un dominio determinado le aplica la transformación Q indicada en la ecuación (4.2) de tal forma que el polinomio calculado en el punto así obtenido toma mayor valor que en el punto anterior. en términos formales:

Teorema 4.2: Sea $P(\Theta)$ un polinomio homogéneo con coeficientes no negativos de grado d en sus variables $\Theta = \{\Theta_{ij}\}$. Sea $\theta = \{\ \theta_{ij}\ \}$ un punto del dominio $D = \{\ \theta_{ij} \mid \theta_{ij} \geq 0,\ \sum_{j=1}^{q_i} \theta_{ij} = 1,\ i = 1,..,p \text{ y } j = 1,..,q_i\ \}$, y sea $Q(\Theta)$ un punto de D definido como:

$$Q(\Theta)_{ij} = \frac{\theta_{ij}(\partial P/\partial\Theta_{ij})_\theta}{\sum_{k=1}^{q_i} \theta_{ik}(\partial P/\partial\Theta_{ik})_\theta} \qquad (4.2)$$

tal que $\forall\ i \sum_{k=1}^{q_i} \theta_{ik}\ (\partial P/\partial\Theta_{ik}\ ) \neq 0$. Entonces, $P\ (Q(\theta)) \geq P\ (\theta)$ excepto si $Q\ (\theta)$=θ.[SJA99]

La aplicación del teorema anterior permite obtener un máximo local del polinomio P en el espacio de búsqueda definido por D haciendo uso del algoritmo de descenso por gradiente (Algoritmo 4.3).

1. **Entrada** : $P(\Theta)$
2. **repetir** : calcular $Q(\Theta)$ utilizando $P(\Theta)$
3. $\theta = Q(\theta)$
4. **hasta** converger
5. **salida** : θ

Algoritmo 4.3: Esquema general de los algoritmos de estimación [SJA99].

Dado un polinomio que cumple las condiciones del teorema anterior y un punto inicial ϑ del dominio definido en el teorema, el problema de la estimación se plantea como la aplicación repetida de la transformación hasta alcanzar un máximo local del polinomio definido.

Un punto importante en este algoritmo es la elección del valor inicial del parámetro ϑ, ya que condiciona el máximo alcanzado.

Nótese como las probabilidades de una GIP son un punto del dominio definido en el teorema 4.2. Para una GIP G_p:

$p(A_i \longrightarrow \alpha_j) = \vartheta_{ij}$, i = 1,...,$|V|$; j = 1,...,$|\Gamma_{A_i}|$,

Donde Γ_{A_i} representa el conjunto de reglas de la gramática cuyo antecedente es A_i.

Por tanto, una función criterio definida en términos de estas probabilidades (el polinomio P() en el teorema 4.2) puede ser convenientemente maximizada haciendo uso del teorema 4.2 y el algoritmo 4.3

En cuanto a la función criterio $f_p()$ (ver expresión 4.1) a optimizar será la función de verosimilitud de la muestra:

ln $\prod_{x\in\Omega}$ Pr $(x \mid G_p)$, es la función $f_p()$ de la expresión 4.1.

Ahora, nuestro problema consiste en obtener un parámetro p' tal que:

$$\text{p'} = \arg \max_{p} \ln \prod_{x\in\Omega} \Pr\ (x \mid G_p)$$

El Teorema 4.2 junto con el algoritmo 4.3 nos proporciona el mecanismo apropiado para optimizar la función elegida, y de esta forma obtener p'. De esta forma ya tenemos todos los mecanismos necesarios para el proceso de estimación.

Un método para obtener p' consiste en utilizar el algoritmo Inside-Outside (IO). Este algoritmo procede de forma iterativa, maximizando en cada etapa la verosimilitud de la muestra hasta alcanzar un máximo local.

La función de verosimilitud de la muestra es una función definida en términos del teorema 4.2 por lo que puede definirse una transformación creciente para optimizarla en la que se basa el algoritmo IO. La convergencia del algoritmo es garantizada por el teorema.

4.3.2 El algoritmo IO

Sea $G_p = (G\ ,\ p)$ una GIP, y sea Ω una muestra de $L(G)$, es decir, un conjunto de cadenas de $L(G)$ en el cual puede haber cadenas repetidas. La función de verosimilitud de la muestra Ω dada por la GIP G_p se define como:

$$\Pr\ (\Omega \mid G_p) = \prod_{x \in \Omega} \Pr\ (x \mid G_p). \qquad (4.3)$$

Puesto que esta función es un polinomio que cumple con las condiciones del Teorema 4.2 se puede definir una transformación $\forall (A \longrightarrow \alpha) \in P$ como:

$$\overline{p}(A \longrightarrow \alpha) = \frac{p\ (A \longrightarrow \alpha)(\dfrac{\partial \ln \Pr(\Omega \mid G_p)}{\partial p\ (A \longrightarrow \alpha)})}{\displaystyle\sum_{(a \longrightarrow \alpha) \in \Gamma_A} p\ (A \longrightarrow \alpha)(\frac{\partial \ln \Pr(\Omega \mid G_p)}{\partial p\ (A \longrightarrow \alpha)})}.$$

Por el Teorema 4.2 esta transformación permite obtener una GIP $G_{\overline{p}} = (G, \overline{p})$ tal que

$\Pr\ (\Omega \mid G_{\overline{p}}) > \Pr\ (\Omega \mid G_p)$ a no ser de que $\overline{p} = p$

Resolviendo parcialmente la expresión anterior la transformación que obtenemos $\forall (A \longrightarrow \alpha) \in P$ es:

$$\overline{p}(A \longrightarrow \alpha) = \frac{\displaystyle\sum_{x \in \Omega} \frac{1}{\Pr\ (x \mid G_p)} \quad p\ (A \longrightarrow \alpha)(\frac{\partial \Pr(x \mid G_p)}{\partial p\ (A \longrightarrow \alpha)})}{\displaystyle\sum_{x \in \Omega} \frac{1}{\Pr(x \mid G_p)} \sum_{(a \longrightarrow \alpha) \in \Gamma_A} p\ (A \longrightarrow \alpha)(\frac{\partial \Pr(x \mid G_p)}{\partial p\ (A \longrightarrow \alpha)})} \qquad (4.4)$$

Para resolver la expresión anterior vamos a resolver parte de su numerador haciendo

uso de las definiciones establecidas en capítulos anteriores.

$$p\ (A \longrightarrow \alpha)(\frac{\partial \ln \Pr(x \mid G_p)}{\partial p\ (A \longrightarrow \alpha)}) = p\ (A \longrightarrow \alpha) \sum_{d_x \in D_x} (\frac{\partial \Pr(x, d_x \mid G_p)}{\partial p\ (A \longrightarrow \alpha)})$$

$$= \sum_{d_x \in D_x} \mathrm{N}\ (A \longrightarrow \alpha, d_x) \Pr(x, d_x \mid G_p).$$

Resolvemos a continuación parte del denominador de (4.4) haciendo uso de la expresión anterior y teniendo en cuenta que el número de veces que el no terminal A ha sido derivado en d_x es $\mathrm{N}\ (A, d_x) = \sum_{(A \longrightarrow \alpha) \in \Gamma_A} \mathrm{N}\ (A \longrightarrow \alpha, d_x)$:

$$\sum_{(A \longrightarrow \alpha) \in \Gamma_A} p\ (A \longrightarrow \alpha)\ (\frac{\partial \Pr(x \mid G_p)}{\partial p\ (A \longrightarrow \alpha)}) = \sum_{(A \longrightarrow \alpha) \in \Gamma_A} \sum_{d_x \in D_x} \mathrm{N}\ (A \longrightarrow \alpha, d_x) \Pr(x, d_x \mid G_p)$$

$$= \sum_{d_x \in D_x} \mathrm{N}(A, d_x) \Pr(x, d_x \mid G_p)$$

Finalmente, la expresión (4.4) queda $\forall (A \longrightarrow \alpha) \in P$ como:

$$\bar{p}(A \longrightarrow \alpha) = \frac{\sum_{x \in \Omega} \frac{1}{\Pr(x \mid G_p)} \sum_{d_x \in D_x} N(A \longrightarrow \alpha, d_x) \Pr(x, d_x \mid G_p)}{\sum_{x \in \Omega} \frac{1}{\Pr(x \mid G_p)} \sum_{d_x \in D_x} N(A, d_x) \Pr(x, d_x \mid G_p)} \qquad (4.5).$$

Cuando la GIP esta en Forma Normal Chomsky tal transformación puede ser reescrita adecuadamente en términos de las probabilidades:

Inside: $e(A<i,j>)$ y *Outside*: $f(A<i,j>)$

$$e(A < i, i >) = p\,(A \to x_i) \qquad 1 \leq i \leq |x|$$

$$f(A < i, j >) = \Pr\ (S \Rightarrow x_1 ... x_{i-1} A x_{j+1} ... x_x \mid G_p)$$

Para toda $A \to \alpha$, con $A \in V$ y $\alpha \in \sum$ y la gramática en FNC , la expresión (4.5) será:

$$\overline{p}(A \to BC) = \frac{\sum_{x \in \Omega} \frac{1}{\Pr(x \mid G_p)} \sum_{i=1, a=x_i}^{|x|} f(< i, i >) p(A \to x_i)}{\sum_{x \in \Omega} \frac{1}{\Pr(x \mid G_p)} \sum_{i=1}^{|x|} \sum_{j=1}^{|x|} f(A < i, j >) e(A < i, j >)}$$

y para toda $(A \to a) \in P$:

$$\overline{p}(A \to a) = \frac{\sum\limits_{x \in \Omega} \frac{1}{Pr(x|G_p)} \sum\limits_{\substack{i=1 \\ a=x_i}}^{|x|} f(A < i, i >) p(A \to x_i)}{\sum\limits_{x \in \Omega} \frac{1}{Pr(x|G_p)} \sum\limits_{i=1}^{|x|} \sum\limits_{j=1}^{|x|} f(A < i, i >) e(A < i, j >)}$$

$$\bar{p}(A \to BC) = \frac{\sum_{x \in \Omega} \frac{1}{\Pr(x \mid G_p)} \sum_{i=1, a=x_i}^{|x|} f(< i, i >) p(A \to x_i)}{\sum_{x \in \Omega} \frac{1}{\Pr(x \mid G_p)} \sum_{i=1}^{|x|} \sum_{j=1}^{|x|} f(A < i, j >) e(A < i, j >)}$$

Observamos como en estas expresiones aparecen los valores que se calculan con los algoritmos Inside y Outside. El algoritmo IO consiste en la aplicación iterativa de esta transformación siguiendo el esquema del algoritmo 4.3 sobre las probabilidades de las reglas en la gramática G_p, a partir de unos valores iniciales de la GIP y obteniendo en cada paso una G_p hasta maximizar localmente la función (4.3).

4.3.3 El algoritmo VS

La probabilidad de la mejor derivación de una muestra es también un polinomio definido en términos del Teorema 4.2 El algoritmo VS se basa en la definición de una transformación creciente para optimizar esta función. Este algoritmo puede utilizarse para

aproximar la verosimilitud de la muestra considerando únicamente la información contenida en la mejor derivación. La función que define en este caso es:

$$\ln \hat{\Pr}\ (\Omega \mid G_p) = \ln \prod_{x \in \Omega} \Pr\ (x, \hat{d}_x \mid G_p).$$

Puesto que esta función es un polinomio que cumple las condiciones del teorema 4.2 se puede definir una transformación para maximizar dicha función. Entonces $\forall (A \longrightarrow \alpha) \in P$.

$$\hat{p}\ (A \longrightarrow \alpha) = \frac{\sum\limits_{x \in \Omega} N(A \longrightarrow \alpha, \hat{d}_x)}{\sum\limits_{x \in \Omega} N(A, \hat{d}_x)}$$

Como vimos en la sección 4.2.3 la mejor derivación de una cadena puede ser calculada utilizando el algoritmo de Viterbi (VS). Al igual que en el algoritmo IO, el cálculo de la mejor derivación esta basado en la definición:

$$\hat{e}(A < i, j >) = \hat{\Pr}(A \stackrel{*}{\Rightarrow} w_i ... w_j \mid G_p)$$

que es la mejor derivación que genera la subcadena $w_i....w_j$ dada la gramática G_p

El algoritmo VS, en cada iteración calcula la mejor derivación de la cadena, para finalmente aplicar la transformación.

Capítulo 5

MODELOS DE N-GRAMAS DE LENGUAJE BASADOS EN CLASES

En aplicaciones tales como: Traducción automática, Reconocimiento óptico de caracteres, corrección ortográfica automática, etc., en la que se procesa el lenguaje natural, se enfrenta el problema de recuperar una cadena de palabras después de haber sido degradadas al pasar por un canal ruidoso. Para enfrentar este problema en este capítulo se estudia un método para estimar la probabilidad con la que cualquier cadena particular de palabras será presentada como una entrada al canal de ruido. También se estudia lo relacionado con la asignación de palabras a clases de acuerdo al comportamiento estadístico sobre un gran corpus.

En la siguiente sección se revisará el concepto de modelo de lenguaje y se dará una definición de modelos de n-gramas. Luego se mirará en el subconjunto de los modelos de n-gramas en los que las palabras están divididas en clases. Se mostrará que para n = 2 la asignación de máxima verosimilitud de palabras a clases es equivalente a la asignación para la cual la información mutua promedio de clases adyacentes es la máxima. Por último se presentan dos algoritmos para dividir el vocabulario en clases, los cuales parten, uno del criterio de máxima verosimilitud y el otro del criterio de perplejidad.

5.2 Modelos de lenguaje

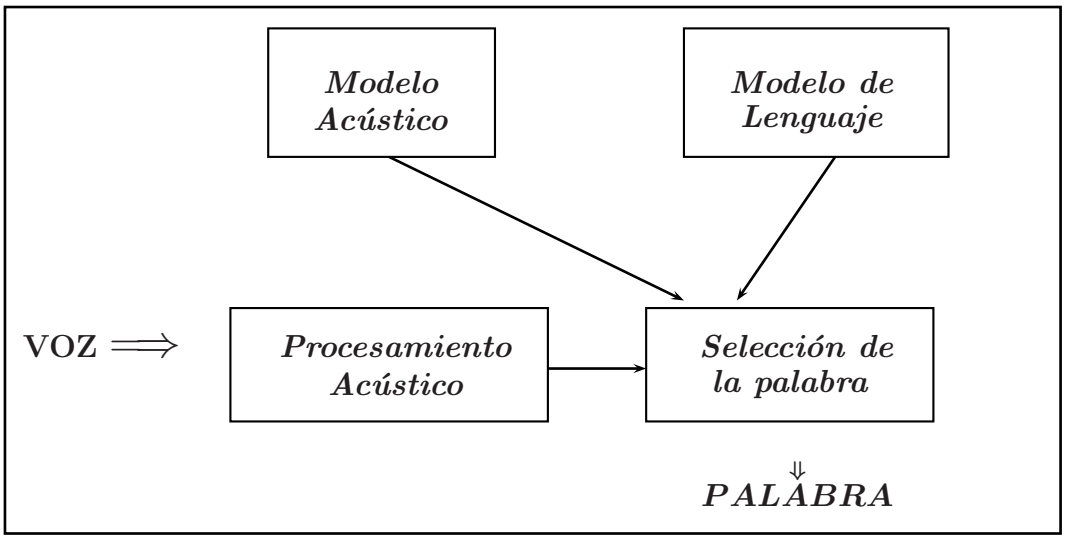


Fig. 5.1 Componentes básicos del proceso de reconocimiento del habla [AJ01]

El reconocimiento del habla es un proceso de clasificación de patrones, cuyo objetivo es clasificar la señal de entrada (onda acústica) en una secuencia de patrones previamente aprendidos y almacenados en unos diccionarios de modelos acústicos y de lenguaje, que permitirá obtener automáticamente una secuencia de caracteres que representan la señal de entrada[3].

Este proceso de clasificación supone; en primer lugar, que la señal de voz puede ser analizada en segmentos de corta duración y representar cada uno de los segmentos mediante su contenido frecuencial, de forma análoga al funcionamiento del oído; en segundo lugar, que mediante un proceso de clasificación se puede asignar a cada segmento o conjuntos consecutivos de segmentos una unidad con significado lingüístico y finalmente; en tercer lugar, que mediante un procesador lingüístico se puede dar significado a las secuencias de unidades. Este último paso del sistema supone incorporar al sistema de Reconocimiento Automático del Habla (RAH) conocimiento acerca de la

[3]Similar planteamiento se hace para otras aplicaciones como: Traducción automática, reconocimiento óptico de caracteres, etc.

estructura sintáctica, semántica y pragmática del lenguaje.

Sin embargo, los sistemas actuales de RAH solo incorporan estas fuentes de conocimiento sobre tareas muy restringidas y controladas, estando la mayoría de ellos en experimentación en condiciones de laboratorio. Matemáticamente, el problema del reconocimiento automático del habla se puede formular desde un punto de vista estadístico. Para ello se supone que:

O : Representa una secuencia de T medidas de la señal de voz,
O = $O_1O_2...O_T$, (datos acústicos).

W : Es una secuencia de **N** palabras que pertenecen a un vocabulario conocido, W = $W_1W_2...W_N$.

P (W | O) : Es la probabilidad de que la secuencia de palabras **W** se haya pronunciado dada la observación de los datos acústicos **O**.

El sistema de reconocimiento debe decidir la secuencia de palabras **W** que maximice la probabilidad **P (W | O)**

$$\boldsymbol{W'} = \arg\max_{\boldsymbol{W} \in \boldsymbol{V}^+} \boldsymbol{P(W \mid O)}$$

Utilizando la fórmula de Bayes se obtiene:

$$\boldsymbol{P(W \mid O)} = \frac{\boldsymbol{P(O \mid W)\, P(W)}}{\boldsymbol{P(O)}} \qquad \textbf{(5.1)}$$

Donde:

P(W): Es la probabilidad de la secuencia de palabras **W**

P (O | W) : Es la probabilidad de observar la secuencia de datos acústicos **O** cuando se pronuncia la secuencia de palabras **W**.

P(O) : Es la probabilidad de la secuencia de datos acústicos **O**

Sin embargo, como la probabilidad de la secuencia de datos acústicos $\mathbf{P(O)}$ es la misma independientemente de la secuencia de palabras pronunciada, en el proceso de maximización, esta probabilidad puede ser eliminada, Se obtiene la ***Fórmula fundamental del reconocimiento automático del habla***

$$\boldsymbol{W}' = \arg\max_{\boldsymbol{W}\in\boldsymbol{V}^+} \boldsymbol{P(O \mid W)P(W)} \qquad \mathbf{(5.2)}$$

Es decir, la secuencia de palabras reconocida es aquella que maximiza el producto de dos probabilidades, una $\boldsymbol{P(O \mid W)}$ que relaciona los datos acústicos con la secuencia de palabras y que se denominará modelo acústico y $\boldsymbol{P(W)}$ que únicamente depende de la secuencia de palabras y que se denominará modelo de lenguaje.

A partir de la fórmula (5.2) se desprende que se necesita poder estimar la probabilidad a priori $\boldsymbol{P(W)}$ de todas las palabras que el usuario desea pronunciar.

Utilizando La Ley de Bayes se obtiene la siguiente descomposición:

$$\boldsymbol{P(W)} = \prod_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{P(W_i \mid \underbrace{W1, W2, ..., W_{i-1}}_{hi})} \qquad \mathbf{(5.3)}$$

Es absurdo pensar que la pronunciación de la i-ésima palabra depende de toda la historia pronunciada anteriormente, además la cantidad de eventos a estimar en (5.3) es demasiado elevada por lo tanto, se suele definir una relación de equivalencia sobre las historias, digamos $\phi(h_i)$ y se considera en el modelo solamente el conjunto de clases de equivalencia $\phi(h_i)$.

$$\boldsymbol{P(W)} = \prod_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{P(W_i \mid \phi(h_i)\,)} \qquad \mathbf{(5.4)}$$

El arte del modelado de lenguaje consiste en determinar apropiadamente las clases de equivalencia ϕ y el método de estimación de las probabilidades

$$\boldsymbol{P(W_i \mid \phi(W1, W2, ..., W_{i-1}))} \qquad \mathbf{(5.5)}$$

Se debe escoger frecuentemente cual de dos diferentes modelos de lenguaje es el mejor.

El desempeño de un modelo de lenguaje en un sistema completo depende de una delicada interrelación entre el modelo lenguaje y otros componentes del sistema. Un modelo de lenguaje puede sobrepasar a otro como parte del sistema de reconocimiento del habla pero desempeñarse no muy bien en un sistema de traducción. Sin embargo, por que es costoso evaluar un modelo de lenguaje en el contexto de un sistema completo, se busca una medida específica de la calidad de un modelo de lenguaje. Por ejemplo, usar cada modelo de lenguaje para computar la probabilidad conjunta de algún grupo de cadenas y juzgar como mejor el modelo de lenguaje que produzca la mayor probabilidad.

La ***perplejidad*** de un modelo de lenguaje con respecto a una muestra, S, es el recíproco del promedio geométrico de las probabilidades de las predicciones en S [4]. Si S tiene $|S|$ palabras, entonces la perplejidad es $\mathbf{Pr}(\boldsymbol{S})^{-\frac{\mathbf{1}}{|S|}}$. De modo que, el modelo de lenguaje con la perplejidad mas pequeña será el que asigne la mayor probabilidad para S. Puesto que la perplejidad no solo depende del modelo de lenguaje sino también del texto con respecto al que es medida, es importante que el texto sea representativo de aquel para el cual el modelo de lenguaje es creado. Puesto que la perplejidad esta sujeta al error de muestreo, hacer buenas distinciones entre los modelos del lenguaje requiere que la perplejidad sea medida con respecto a una muestra grande.

5.3 Modelos de n-gramas

En el modelo de n-gramas se define la relación de equivalencia $\phi(h_i)$ [AJ01] como:

$$h_w = w_1 w_2 ... w_n ... w_m$$

$$h_v = v_1 v_2 ... v_n ... v_k$$

$$h_w \approx h_v \quad \text{si y solo si coinciden en las últimas n-1 palabras}$$

En cuanto al cálculo de la probabilidad, se realiza por medio de la frecuencias relativas obtenidas del corpus de entrenamiento. Si Ω es un gran corpus de datos, $C(w_{i-n+1}, ...w_{i-1}, w_i)$

[4] $Pr(S) = \left[\prod_{s \in S} P(s)\right]^{-\frac{1}{|S|}}$

el número de veces que la secuencia $w_{i-n+1}, ...w_{i-1}, w_i$ ocurre en Ω y $C(\phi(h_i))$ el número de veces que la clase de la historia de w_i ha ocurrido en Ω, entonces

$$\boldsymbol{p(w_i \mid \phi(h_i)) \approx \frac{C(w_{i-n+1}, ...w_{i-1}, w_i)}{C(\phi(h_i))}} \qquad \textbf{(5.6)}$$

que es la ***estimación de máxima verosimilitud***

En el caso de n =3, es decir el modelo de trigramas, la ecuación anterior toma la forma

$$\boldsymbol{p(w_i \mid w_{i-2}, w_{i-1}) \approx \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}} \qquad \textbf{(5.7)}$$

5.3.1 Ventajas y Desventajas del Modelo de N-Gramas

La potencia del modelo de n-gramas se debe principalmente a que está firmemente basado en datos.

- Recoge muy bien las relaciones locales en la frase
- Las probabilidades de sus eventos elementales se estiman con facilidad
- Es fácil de implementar y su formulación es muy sencilla

Estas características hacen que el modelo de n-gramas sea el más utilizado en los sistemas de reconocimiento del habla como también en otros campos en los que se procesa información lingüística mediante sistemas probabilísticos.

Por otra parte, las principales limitaciones del modelo de *n-gramas* son:

1. Por razones computacionales el valor de n se debe mantener bajo ($n \leq 3$), de manera que solamente captura información inmediata.
2. Los n-gramas no se acomodan a los cambios dinámicos en el discurso, las frecuencias relativas de los n-gramas reflejan promedios sobre el corpus de entrenamiento, sin embargo cuando hay cambio de tópico en el discurso, es posible que para algunas palabras la probabilidad de ocurrencia se vea modificada.

3. El modelo de n-gramas tiene problemas de dispersión: hay una gran cantidad de eventos, en el espacio de eventos del modelo, que no son vistos durante el entrenamiento y por tanto la frecuencia relativa es cero lo cual implicará que se les asigne probabilidad 0.

5.4 Suavisado

La principal técnica utilizada para corregir el problema de la dispersión en el modelo de n-gramas es el suavisado [CG98]. El suavisado es la técnica mediante la cual se ajustan las probabilidades estimadas en el modelo de tal manera que a los eventos no vistos les sea asignada una probabilidad diferente de cero. El uso del suavisado produce distribuciones más uniformes incrementando las probabilidades bajas y reduciendo las probabilidades altas, aumentando así la capacidad predictiva de los modelos.

La mayoría de los algoritmos de suavisado se enmarcan dentro del esquema de ***backoff***[5] es decir, la probabilidad de un evento en el modelo suavisado se calcula como:

$$p_{suav}\left(w_i \mid w_{i-n+1}^{i-1}\right) = \begin{cases} \alpha(w_i \mid w_{i-n+1}^{i-1}) \quad Si \quad c\left(w_{i-n+1}^{i}\right) > 0 \\ \gamma(w_{i-n+1}^{i-1})\, p_{suav}\left(w_{i-n+2}^{i-1}\right) \quad Si \quad c\left(w_{i-n+1}^{i}\right) = 0 \end{cases}$$

es decir, si un *n-grama* tiene frecuencia diferente de cero, se utiliza la distribución $\alpha(w_i \mid w_{i-n+1}^{i-1})$ para calcular su probabilidad, de lo contrario se desciende a la distribución de *n-gramas* de menor orden. $\alpha(w_i \mid w_{i-n+1}^{i-1})$ es una distribución dada y el factor $\gamma(w_{i-n+1}^{i-1})$ es un factor de normalización.

Otros algoritmos de suavisado son expresados mediante interpolación lineal entre modelos de *n-gramas* de diferente orden:

$$p_{suav}\left(w_i \mid w_{i-n+1}^{i-1}\right) = \lambda\, p\left(w_i \mid w_{i-n+1}^{i-1}\right) + (1-\lambda)\, p_{suav}\left(w_i \mid w_{i-n+2}^{i-1}\right)$$

puede mostrarse, que los algoritmos de suavisado mediante interpolación lineal se

[5]Un estudio de las más importantes técnicas de suavisado ha sido llevado a cabo por Chen y Goodman

pueden expresar en términos del marco definido por el ***backoff***.

Para un vocabulario de tamaño V, un modelo de 1-grama tiene $V-1$ parámetros independientes, uno para cada palabra menos uno para la restricción de que todas las probabilidades suman uno. Un modelo 2-gramas tiene $V(V-1)$ parámetros independientes de la forma Pr $(w_2 \mid w_1)$ y $V-1$ de la forma Pr (w) para un total de V^2-1 parámetros independientes. En general, un modelo n-gramas tiene V^n-1 parámetros independientes: $V^{n-1}(V-1)$ de la forma Pr $(w_k \mid w_1^{k-1})$, llamados los parámetros de orden n, mas los parámetros $V^{n-1}-1$ de un modelo de (n-1)-gramas.

Se estiman los parámetros de un modelo n-gramas examinando una muestra del texto, $\boldsymbol{t_1^T}$, al que se le llama ***texto de entrenamiento***, en un proceso llamado entrenamiento. Si $\boldsymbol{C(w)}$ es la cantidad de veces en que la palabra w se da en la cadena $\boldsymbol{t_1^T}$, entonces para un modelo de lenguaje de 1-grama la estimación de máxima verosimilitud para el parámetro Pr(w) es C$(w)/T$ (T es la cantidad de palabras que tiene el corpus).

El vocabulario es muy amplio y así aún para valores pequeños de n, la cantidad de parámetros en un modelo de n-gramas es enorme. El sistema de reconocimiento del habla IBM Tangora tiene un vocabulario de aproximadamente veinte mil palabras y emplea un modelo de lenguaje 3-gramas con cerca de 8 trillones de parámetros. Se pueden ilustrar los problemas relacionados con el cálculo de los parámetros para un modelo de 3-gramas, con los datos del cuadro 5.1. Ahí, se muestra el número de 1-, 2- y 3- gramas que aparecen con varias frecuencias en una muestra de 365.893.263 palabras de texto inglés de una variedad de fuentes. El vocabulario consiste de las 260.740 palabras diferentes mas una "palabra desconocida" especial dentro de la cual todas las otras palabras son enviadas. De los 2-gramas que podrían haber ocurrido en el dato solo 14.494.217 ocurrieron y de esos, 8.045.024 ocurrieron solo una vez. Igualmente, de los 3-gramas que pudieron haber ocurrido, solo 75.349.888 ocurrieron y de estos 53.737.350 ocurrieron solo una vez. De estos datos se puede esperar que las estimaciones de máxima verosimilitud serán 0 para el 14,7 % de los 3-gramas y 2.2 % de los 2-gramas en

una nueva muestra de texto ingles. Se puede asegurar que cualquier 3-grama que no aparece en la muestra es de hecho raro, pero hay tantos de ellos, que su probabilidad total es substancial.

conteo	1-gramas	2-gramas	3-gramas
1	36,789	8,045,024	53,7737,350
2	20,269	2,065,469	9,229,958
3	13,123	970,434	3,653,791
> 3	135,335	3,413,290	8,728,789
> 0	205,516	14,494,217	75,349,888
≥ 0	260,741	6.799×10^{10}	1.773×10^{16}

Cuadro 1 [BDP90]: Cantidad de n-gramas con varias frecuencias en 365,893,263 palabras de texto corrido

5.5 Clases de Palabras

Algunas palabras son claramente similares a otras en su significado y función sintáctica. Si se puede asignar palabras exitosamente a clases, puede ser posible hacer predicciones razonables para las historias que no hemos visto anteriormente asumiendo que ellas son similares a otras historias que han sido vistas.

Supongase que se particiona un vocabulario de V palabras en C clases usando una función,π, que asigna una palabra, w_i en su clase, c_i. Se dice que un modelo de lenguaje es un *modelo de n-grama de clases* si es un modelo de n-gramas y si además para $1 \leq k \leq n$, $\Pr(w_k \mid w_1^{k-1}) = \Pr(w_k \mid c_k)\ \Pr\ (c_k \mid c_1^{k-1})$. Un modelo de n-grama basado en clases tiene $C^n - 1 + V - C$ parámetros independientes: $V - C$ de la forma $\Pr(w_k \mid c_i)$, mas $C^n - 1$ parámetros independientes de un modelo de lenguaje n-gramas para un vocabulario de tamaño C. Por lo tanto, excepto en los casos triviales en los que $C = V$ o n = 1, un modelo de n-grama de clases siempre tiene menos parámetros

independientes que un modelo de n-grama general.

Dado el texto de entrenamiento, t_1^T, las estimaciones de máxima verosimilitud de los parámetros de un modelo de clase 1-grama son

$$Pr(\boldsymbol{w} \mid \boldsymbol{c}) = \frac{\boldsymbol{C(w)}}{\boldsymbol{C(c)}} \qquad \textbf{(5.9)}$$

y

$$Pr(\boldsymbol{c}) = \frac{\boldsymbol{C(c)}}{\boldsymbol{T}} \qquad \textbf{(5.10)}$$

Donde, por ***C(c)*** nos referimos al número de palabras en t_1^T para las que la clase es $\boldsymbol{c}$. De estas ecuaciones, se puede observar que, $c = \pi(w)$, $\Pr(w) = \Pr(w \mid c)\Pr(c) = C(w)/T$. Para un modelo 1-gramas de clases, la selección de la aplicación π no tiene efecto. Para un modelo 2-gramas de clases, la estimación de máxima verosimilitud de los parámetros de orden 2 y están dadas por

$$Pr(\boldsymbol{c_2} \mid \boldsymbol{c_1}) = \frac{\boldsymbol{C(c_1 c_2)}}{\sum_{\boldsymbol{c}} \boldsymbol{C(c_1 c)}} \qquad \textbf{(5.11)}$$

Por definición, $\Pr(c_1 c_2) = \Pr(c_1)\ \Pr(c_2 \mid c_1)$, y así, para la estimación de máxima verosimilitud

$$Pr(\boldsymbol{c_1 c_2}) = \frac{\boldsymbol{C(c_1 c_2)}}{\boldsymbol{T}} \frac{\boldsymbol{C(c_1)}}{\sum_{\boldsymbol{c}} \boldsymbol{C(c_1 c)}} = \frac{C(c_1 c_2)}{T} \qquad \textbf{(5.12)}$$

Puesto que $C(c_1)$ y $\sum_c C(c_1 c)$ son los números de palabras por las que la clase es c_1 en las cadenas t_1^T y t_1^{T-1} respectivamente, el término final en esta ecuación tiende a 1 cuando T tiende a infinito. Así que, $\Pr(c_1 c_2)$ tiende a la frecuencia relativa de $c_1 c_2$ como clases consecutivas en el texto de entrenamiento.

Para obtener la mejor clasificación se busca aquella, que minimice la perplejidad; como se verá adelante, esta es la misma clasificación que maximiza la información mutua promedio.

Sea

$$PP = Pr(T)^{-\frac{1}{|T|}} \quad \textbf{(Perplejidad)}$$

$$LogPP = -\frac{1}{|T|} LogPr(T)$$

$$L(\pi) = -LogPP = \frac{1}{|T|} LogPr(T)$$

$$y \qquad T = t_1^{t-1}$$

$$L(\pi) = \frac{1}{T-1} LogPr(t_2 \mid t_1) Pr(t_3 \mid t_2) ... Pr(t_T \mid t_{T-1})$$

$$L(\pi) = \frac{1}{T-1} \sum_{i=2}^{T} LogPr(t_i \mid t_{i-1})$$

Generalizando a toda pareja $w_1 w_2$:

$$L(\pi) = \frac{1}{T-1} \sum_{w_1 w_2} c(w_1 w_2) LogPr(w_i \mid w_{i-1})$$

Pero

$$Pr(w_i \mid w_{i-1}) = Pr(w_i \mid c_i) Pr(c_i \mid c_{i-1})$$

Luego

$$\begin{aligned}
L(\pi) &= \sum_{w_1 w_2} \frac{c(w_1 w_2)}{T-1} LogPr(c_2 \mid c_1) Pr(w_2 \mid c_2) \\
&= \sum_{c_1 c_2} \frac{c(c_1 c_2)}{T-1} LogPr(c_2 \mid c_1) \frac{Pr(c_2)}{Pr(c_2)} Pr(w_2 \mid c_2) \\
&= \sum_{c_1 c_2} \frac{c(c_1 c_2)}{T-1} Log \left[\frac{Pr(c_2 \mid c_1)}{Pr(c_2)} * Pr(w_2 \mid c_2) Pr(c_2) \right] \\
&= \sum_{c_1 c_2} \frac{c(c_1 c_2)}{T-1} Log \frac{Pr(c_2 \mid c_1)}{Pr(c_2)} \quad + \quad \sum_{c_1 c_2} \frac{c(c_1 c_2)}{T-1} Log \underbrace{Pr(w_2 \mid c_2) Pr(c_2)}_{Pr(w_2)} \\
&= \sum_{c_1 c_2} \frac{c(c_1 c_2)}{T-1} Log \frac{Pr(c_2 \mid c_1)}{Pr(c_2)} \quad + \quad \sum_{w_1 w_2} \frac{c(w_1 w_2)}{T-1} LogPr(w_2) \\
&= \sum_{c_1 c_2} \frac{c(c_1 c_2)}{T-1} Log \frac{Pr(c_2 \mid c_1)}{Pr(c_2)} \quad + \quad \sum_{w_1} \frac{\sum_{w_1} c(w_1 w_2)}{T-1} LogPr(w_2) \qquad \textbf{(5.13)}
\end{aligned}$$

Pero

$\sum_{w} \frac{c(w_1 w_2)}{T-1}$, tiende a la frecuencia relativa de w_2 por lo tanto es aproximadamente $Pr(w_2)$

$$L(\pi) = \sum_{c_1 c_2} \frac{c(c_1 c_2)}{T-1} Log \frac{Pr(c_2 \mid c_1)}{Pr(c_2)} \quad + \quad \sum_{w} Pr(w) Log Pr(w)$$

$$L(\pi) = \underbrace{\sum_{c_1 c_2} Pr(c_1 c_2) Log \frac{Pr(c_2 \mid c_1)}{Pr(c_2)}}_{I(c_1, c_2)} \quad + \quad \underbrace{\sum_{w} Pr(w) Log Pr(w)}_{-H(w)}$$

$$\boldsymbol{L(\pi) = -H(w) + I(c_1, c_2)} \qquad \textbf{(5.14)}$$

Donde $\boldsymbol{H(w)}$ es la *entropía* de la distribución de palabra 1-grama y $\boldsymbol{I(c_1, c_2)}$ es la *información mutua promedio de las clases adyacentes*. Puesto que $L(\pi)$ depende de π solo a través de información mutua promedio, la partición que maximiza $L(\pi)$ es, en el limite, también la partición que maximiza la información mutua promedio de las clases adyacentes.

Inicialmente se asigna cada palabra a una clase distinta y se computa la información mutua promedio entre clases adyacentes. Luego se fusiona el par de clases con menor pérdida de información mutua promedio, después de $V - C$ fusiones, quedan C clases. Frecuentemente se puede encontrar que para las clases obtenidas de esta manera la información mutua promedio puede ser más grande moviendo algunas palabras de una clase a otra. Por eso, después de haber derivado un grupo de clases a partir de fusiones sucesivas, se pasa por el vocabulario moviendo cada palabra para la clase para la que la partición resultante tiene la mayor información mutua promedio. Eventualmente ninguna reasignación potencial de una palabra conduce a una partición con la información mutua promedio más grande. En este punto se detiene.

Para hacer práctico este algoritmo subóptimo uno debe tomar con sumo cuidado la implementación. Hay aproximadamente $(V - i)^2/2$ fusiones que deben ser investigadas para llevar a cabo el i-ésimo paso. La información mutua promedio restante después de

uno de ellos es la suma de $(V-i)^2$ términos cada uno de los cuales involucra un logaritmo. Puesto que en total debemos hacer $V-C$ fusiones, esta sencilla aproximación para la computación es de orden V^5. No podemos contemplar seriamente tal cálculo excepto para valores muy pequeños de V.

Suponga que ya tenemos hecho $V-k$ fusiones, dando como resultado las clases $C_k(1), C_k(2), ..., C_k(V-k)$ y que ahora se quiere investigar la fusión de $C_k(i)$ con $C_k(j)$ para $1 \leq i \leq j \leq k$. Sea $\boldsymbol{p_k(l,m) = \mathbf{Pr}\ (C_k(l), C_k(m))}$, la probabilidad de que una palabra en la clase $C_k(m)$ sigue a una palabra en la clase $C_k(l)$.

Sea

$$\boldsymbol{pl_k(l) = \sum_m p_k(l,m)} \tag{5.16}$$

$$\boldsymbol{pr_k(m) = \sum_l p_k(l,m)} \tag{5.17}$$

$$\boldsymbol{q_k(l,m) = p_k(l,m)\ log\frac{p_k(l,m)}{pl_k(l)pr_k(m)}} \tag{5.18}$$

La información mutua promedio restante después de $V-k$ fusiones es

$$\boldsymbol{I_k = \sum_{l,m} q_k(l,m)} \tag{5.19}$$

Se utiliza la notación $i+j$ para representar la agrupación obtenida por la fusión de $C_k(i)$ y $C_k(j)$. Así, por ejemplo, $p_k(i+j,m) = p_k(i,m) + p_k(j,m)$ y

$$\boldsymbol{q_k(i+j,m) = p_k(i+j,m)\ log\frac{p_k(i+j,m)}{pl_k(i+j)pr_k(m)}} \tag{5.20}$$

La información mutua promedio restante después de que se une $C_k(i)$ y $C_k(j)$ es entonces

$$\begin{aligned} I_k(i,j) = I_k - S_k(i) - S_k(j) + q_k(i,j) + q_k(j,i) + q_k(i+j,i+j) \\ + \sum_{l \neq i,j} q_k(l,i+j) + \sum_{m \neq i,j} q_k(i+j,m) \end{aligned} \tag{5.21}$$

Donde

$$\boldsymbol{S_k(i) = \sum_l q_k(l,i) + \sum_m q_k(i,m) - q_k(i,j)} \qquad \textbf{(5.22)}$$

Si se conoce $I_k(i)$, $S_k(i)$ y $S_k(j)$, entonces la mayoría del tiempo empleado en la computación $I_k(i,j)$ es dedicada a computar las sumas en la segunda línea de la ecuación (5.14). Cada una de estas sumas tiene aproximadamente $V - k$ términos y entonces se puede reducir el problema de evaluar $I_k(i,j)$ desde uno de orden V^2 a uno de orden V. Podemos mejorar esto más adelante conservando aquellos pares l, m para los que $p_k(l,m)$ es diferente de cero. Del cuadro 1, por ejemplo, los $6{,}799 \times 10^{10}$ 2-gramas que podría haber ocurrido en la muestra de entrenamiento solo 14,494,217 ocurrieron realmente. Así que, en este caso, las sumas requeridas en la ecuación (5.14) tienen, un promedio, de solo cerca de 56 términos no nulos en lugar de 260,741 como se podría esperar del tamaño del vocabulario.

Examinando todos los pares, se puede encontrar el par, $i < j$, para el que la pérdida de información mutua promedio, $\boldsymbol{L_k(i,j) \equiv I_k - I_k(i,j)}$, sea menor. Se Completa el paso por medio de la fusión de $C_k(i)$ y $(C_k(j)$ para formar un nuevo grupo $C_{k-1}(i)$. Si $j \neq k$, se renombra $C_k(k)$ como $C_{k-1}(j)$ y para $l \neq i, j$, se redefine $C_{k-1}(l)$ como $C_k(l)$. Obviamente $I_{k-1} = I_k(i,j)$.

Algoritmo de agrupamiento Della Pietra

Inicio

Empezar con una asignación inicial $w \rightarrow C_k$

Hacer

Para cada clase k hacer

Calcular conteos de bigramas

Simular fusiones (C_k, C_{k+1})

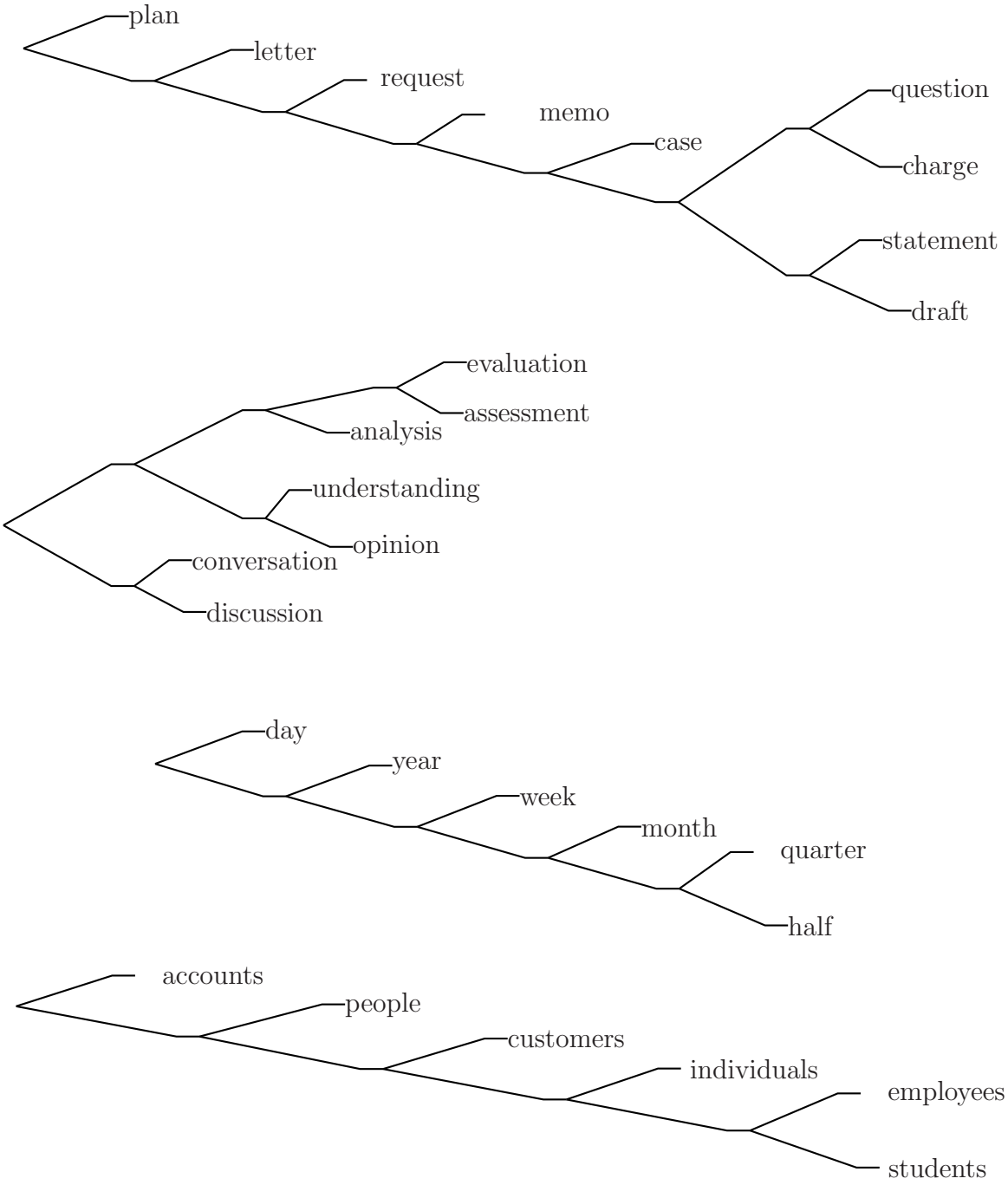
Encontrar la pérdida de información mutua promedio (L_k)

Realizar la fusión para el menor valor de L_k

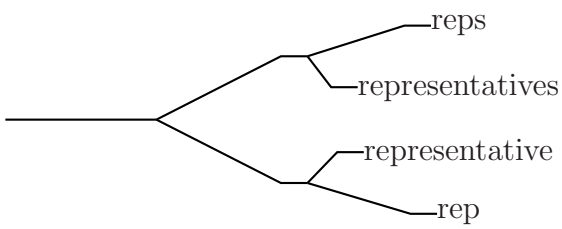
Hasta llegar al número de clases definidas

Fin

Ejemplo de subárboles de un árbol de información mutua de 1000-palabras [BDP90][6]



[6]La mayoría de los corpus son diseñados en inglés, por lo cual los experimentos estan elaborados en inglés.



Aunque se ha descrito este algoritmo como un algoritmo para encontrar grupos, en realidad se determina mucho más. Si se continúa el algoritmo para las $V-1$ fusiones, entonces tendremos un único grupo, que por supuesto será el vocabulario completo. El orden en el que los grupos son fusionados, determina un árbol binario, la raíz del cual corresponde a este único grupo y las hojas corresponden a las palabras en el vocabulario. Los nodos intermedios del árbol corresponden a los agrupamientos intermedios entre palabras únicas y todo el vocabulario. Las palabras que son estadísticamente similares con respecto a sus vecinas inmediatas en el texto de entrenamiento estarán cerca en el árbol. Hemos aplicado este algoritmo de árbol para vocabularios que están por encima de las 5000 palabras. La figura 2 muestra algunas de las subestructuras en un árbol construido de esta manera para las 1000 palabras mas frecuentes en una muestra de correspondencia de oficina. Mas allá de las 5000 palabras este algoritmo también pierde practicidad.

A continuación se mostrará un ejemplo del algoritmo con un texto de entrenamiento pequeño para ver su funcionamiento

Ejemplo

Supongase que se tiene el siguiente vocabulario

$$\boldsymbol{V} = \{a, b, c, d, e, f, g\}$$

TEXTO DE ENTRENAMIENTO (T = 23)

$$a \quad b \quad c \quad a \quad d \quad a \quad f \quad a \quad b \quad b \quad c \quad a \quad b \quad f \quad a \quad c \quad e \quad a \quad b \quad b \quad f \quad a \quad g$$

El primer paso a seguir es sacar los unigramas, partiendo del vocabulario encontrado en el texto de entrenamiento.

Tabla. 5.1 Grupos de 1-gramas

w	Clase	Conteo
a	C_1	8
b	C_2	6
c	C_3	3
d	C_4	1
e	C_5	1
f	C_6	3
g	C_7	1

Segundo paso a seguir es sacar los bigramas, que se obtienen del texto de entrenamiento, es decir siguiendo el orden en que aparecen.

Tabla 5.2 Grupos de 2-gramas

w_1w_2	Clase	Conteo
$a\ b$	C_1C_2	4
$b\ c$	C_2C_3	2
$c\ a$	C_2C_3	2
$a\ d$	C_3C_1	1
$d\ a$	C_1C_4	1
$a\ f$	C_4C_1	1
$f\ a$	C_6C_1	3
$b\ b$	C_2C_2	2
$b\ f$	C_2C_6	2
$a\ c$	C_1C_3	1
$c\ e$	C_3C_5	1
$e\ a$	C_5C_1	1
$a\ g$	C_1C_7	1

El tercer paso a seguir consiste en hacer una tabla que relacione los posibles bigramas que se pueden encontrar en el texto de entrenamiento, con la probabilidad correspondiente a la frecuencia con que aparece cada bigrama.

Tabla 5.3 CLASES DEL VOCABULARIO

$m\|l$	C_1	C_2	C_3	C_4	C_5	C_6	C_7	$Pr_k(m)$
C_1	0	0	2/22	1/22	1/22	3/22	0	7/22
C_2	4/22	2/22	0	0	0	0	0	6/22
C_3	1/22	2/22	0	0	0	0	0	3/22
C_4	1/22	0	0	0	0	0	0	1/22
C_5	0	0	1/22	0	0	0	0	1/22
C_6	1/22	2/22	0	0	0	0	0	3/22
C_7	1/22	0	0	0	0	0	0	1/22
$Pl_k(l)$	8/22	6/22	3/22	1/22	1/22	3/22	0	

El cuarto paso es calcular la información mutua promedio a traves de la fórmula (5.20)

Tabla 5.4 INFORMACION MUTUA PROMEDIO

$m\|l$	C_1	C_2	C_3	C_4	C_5	C_6	C_7	$\sum_l q_k(l,m)$
C_1	0	0	0,0292	0,0226	0,0226	0,0678	0	0,1422
C_2	0,04786	0,00792	0	0	0	0	0	0,05578
C_3	-0,001717	0,03528	0	0	0	0	0	0,033563
C_4	0,019969	0	0	0	0	0	0	0,019969
C_5	0	0	0,03933	0	0	0	0	0,03933
C_6	-0,001717	0,03528	0	0	0	0	0	0,033563
C_7	0,019969	0	0	0	0	0	0	0,019969
$\sum_m q_k(l,m)$	0,084364	0,07848	0,06853	0,0226	0,0226	0,0678	0	

Cómo se realizan las posibles fusiones a través de la Tabla 5.4

- Se calcula la información mutua promedio después de que se une $C_k(i)$ y $C_k(j)$ a través de la fórmula (5.21), de todas las posibles fusiones.

- Se examinan todos los pares, hasta encontrar el par, $i < j$, para el que la pérdida de información mutua promedio, $\boldsymbol{L_k(i,j) \equiv I_k - I_k(i,j)}$, sea menor.

- Al encontrar dicho par se realiza la fusión y con el nuevo conjunto de clases se repite el procedimiento anterior hasta un número determinado de clases. Si se continúa dicho procedimiento se obtendrá de nuevo el vocabulario.

A continuación se mostrará como sería el procedimiento para obtener la primera fusión

Tabla 5.5 PRIMERA POSIBLE FUSIÓN (C_1C_2)

$m\|l$	C_1C_2	C_3	C_4	C_5	C_6	C_7	$Pr_k(m)$
C_1C_2	6/22	2/22	1/22	1/22	3/22	0	13/22
C_3	3/22	0	0	0	0	0	3/22
C_4	1/22	0	0	0	0	0	1/22
C_5	0	1/22	0	0	0	0	1/22
C_6	3/22	0	0	0	0	0	3/22
C_7	1/22	0	0	0	0	0	1/22
$Pl_k(l)$	14/22	3/22	1/22	1/22	3/22	0	

Tabla 5.6 INFORMACION MUTUA PROMEDIO

$m\|l$	C_1C_2	C_3	C_4	C_5	C_6	C_7	$\sum_l q_k(l,m)$
C_1C_2	-0,03804	0,00476	0,01038	0,01038	0,03115	0	0,0186
C_3	0,02676	0	0	0	0	0	0,02676
C_4	0,00892	0	0	0	0	0	0,00892
C_5	0	0,03933	0	0	0	0	0,03933
C_6	0,02676	0	0	0	0	0	0,02676
C_7	0,00892	0	0	0	0	0	0,00892
$\sum_m q_k(l,m)$	0,03332	0,04409	0,01038	0,01038	0,03115	0	0,2587

La pérdida de información mutua es $L_k(C_1, C_2) = 0{,}127254$

Tabla 5.7 SEGUNDA POSIBLE FUSIÓN (C_2C_3)

$m\|l$	C_1	C_2C_3	C_4	C_5	C_6	C_7	$Pr_k(m)$
C_1	0	2/22	1/22	1/22	3/22	0	7/22
C_2C_3	5/22	4/22	0	0	0	0	9/22
C_4	1/22	0	0	0	0	0	1/22
C_5	0	1/22	0	0	0	0	1/22
C_6	1/22	2/22	0	0	0	0	3/22
C_7	1/22	0	0	0	0	0	1/22
$Pl_k(l)$	8/22	9/22	1/22	1/22	3/22	0	

Tabla 5.8 INFORMACION MUTUA PROMEDIO

$m\|l$	C_1	C_2C_3	C_4	C_5	C_6	C_7	$\sum_l q_k(l,m)$
C_1	0	-0,01417	0,0226	0,0226	0,0678	0	0,0988
C_2	0,04183	0,00654	0	0	0	0	0,0483
C_4	0,019969	0	0	0	0	0	0,019969
C_5	0	0,01764	0	0	0	0	0,01764
C_6	-0,001717	0,01928	0	0	0	0	0,0176
C_7	0,019969	0	0	0	0	0	0,019969
$\sum_m q_k(l,m)$	0,080051	0,02929	0,0226	0,0226	0,0678	0	

La pérdida de información mutua es $L_k(C_2, C_3) = 0{,}05874$

Tabla 5.9 TERCERA POSIBLE FUSIÓN (C_3C_4)

$m\|l$	C_1	C_2	C_3C_4	C_5	C_6	C_7	$Pr_k(m)$
C_1	0	0	3/22	1/22	3/22	0	7/22
C_2	4/22	2/22	0	0	0	0	6/22
C_3C_4	2/22	2/22	0	0	0	0	4/22
C_5	0	0	1/22	0	0	0	1/22
C_6	1/22	2/22	0	0	0	0	3/22
C_7	1/22	0	0	0	0	0	1/22
$Pl_k(l)$	8/22	6/22	4/22	1/22	3/22	0	

Tabla 5.10 INFORMACION MUTUA PROMEDIO

$m\|l$	C_1	C_2	C_3C_4	C_5	C_6	C_7	$\sum_l q_k(l,m)$
C_1	0	0	0,050779	0,0226	0,0678	0	0,1411
C_2	0,04786	0,00792	0	0	0	0	0,05578
C_3C_4	0,01257	0,02393	0	0	0	0	0,0365
C_5	0	0	0,03365	0	0	0	0,03365
C_6	-0,001717	0,03528	0	0	0	0	0,033563
C_7	0,019969	0	0	0	0	0	0,019969
$\sum_m q_k(l,m)$	0,078682	0,06713	0,084429	0,0226	0,0678	0	

La pérdida de información mutua es $L_k(C_3, C_4) = 0{,}023733$

Tabla 5.11 CUARTA POSIBLE FUSIÓN (C_4C_5)

$m\|l$	C_1	C_2	C_3	C_4C_5	C_6	C_7	$Pr_k(m)$
C_1	0	0	2/22	2/22	3/22	0	7/22
C_2	4/22	2/22	0	0	0	0	6/22
C_3	1/22	2/22	0	0	0	0	3/22
C_4C_5	1/22	0	1/22	0	0	0	2/22
C_6	1/22	2/22	0	0	0	0	3/22
C_7	1/22	0	0	0	0	0	1/22
$Pl_k(l)$	8/22	6/22	3/22	2/22	3/22	0	

Tabla 5.12 INFORMACION MUTUA PROMEDIO

$m\|l$	C_1	C_2	C_3	C_4C_5	C_6	C_7	$\sum_l q_k(l,m)$
C_1	0	0	0,0292	0,04521	0,0678	0	0,14221
C_2	0,04786	0,00792	0	0	0	0	0,05578
C_3	-0,001717	0,03528	0	0	0	0	0,033563
C_4C_5	0,00628	0	0,02564	0	0	0	0,03192
C_6	-0,001717	0,03528	0	0	0	0	0,033563
C_7	0,019969	0	0	0	0	0	0,019969
$\sum_m q_k(l,m)$	0,070675	0,07848	0,06853	0,04521	0,0678	0	

La pérdida de información mutua es $L_k(C_4, C_5) = 0{,}027369$

Tabla 5.13 QUINTA POSIBLE FUSIÓN (C_5C_6)

$m\|l$	C_1	C_2	C_3	C_4	C_5C_6	C_7	$Pr_k(m)$
C_1	0	0	2/22	1/22	4/22	0	7/22
C_2	4/22	2/22	0	0	0	0	6/22
C_3	1/22	2/22	0	0	0	0	3/22
C_4	1/22	0	0	0	0	0	1/22
C_5C_6	1/22	2/22	1/22	0	0	0	4/22
C_7	1/22	0	0	0	0	0	1/22
$Pl_k(l)$	8/22	6/22	3/22	1/22	4/22	0	

Tabla 5.14 INFORMACION MUTUA PROMEDIO

$m\|l$	C_1	C_2	C_3	C_4	C_5C_6	C_7	$\sum_l q_k(l,m)$
C_1	0	0	0,0292	0,0226	0,09042	0	0,14222
C_2	0,04786	0,00792	0	0	0	0	0,05578
C_3	-0,001717	0,03528	0	0	0	0	0,033563
C_4	0,019969	0	0	0	0	0	0,019969
C_5C_6	-0,001717	0,02393	0,01196	0	0	0	0,034173
C_7	0,019969	0	0	0	0	0	0,019969
$\sum_m q_k(l,m)$	0,084364	0,07848	0,06853	0,0226	0,09042	0	

La pérdida de información mutua es $L_k(C_5, C_6) = 0{,}044373$

Tabla 5.15 SEXTA POSIBLE FUSIÓN (C_6C_7)

$m\|l$	C_1	C_2	C_3	C_4	C_5	C_6C_7	$Pr_k(m)$
C_1	0	0	2/22	1/22	1/22	3/22	7/22
C_2	4/22	2/22	0	0	0	0	6/22
C_3	1/22	2/22	0	0	0	0	3/22
C_4	1/22	0	0	0	0	0	1/22
C_5	0	0	1/22	0	0	0	1/22
C_6	1/22	2/22	0	0	0	0	3/22
$Pl_k(l)$	8/22	6/22	3/22	1/22	1/22	3/22	

Tabla 5.16 INFORMACION MUTUA PROMEDIO

$m\|l$	C_1	C_2	C_3	C_4	C_5	C_6C_7	$\sum_l q_k(l,m)$
C_1	0	0	0,0292	0,0226	0,0226	0,0678	0,1422
C_2	0,04786	0,00792	0	0	0	0	0,05578
C_3	-0,001717	0,03528	0	0	0	0	0,033563
C_4	0,019969	0	0	0	0	0	0,019969
C_5	0	0	0,03933	0	0	0	0,03933
C_6C_7	0,012573	0,023931	0	0	0	0	0,0365
$\sum_m q_k(l,m)$	0,078687	0,07848	0,06853	0,0226	0,0226	0,0678	

La pérdida de información mutua es $L_k(C_6, C_7) = 0{,}0019299$

Después de obtener la pérdida de información, de cada posible fusión, se compara cada una de ellas, y se escoge la menor, quedando entonces en este caso la SEXTA FUSIÓN, para así obtener un conjunto de seis nuevas clases, si se sigue el procedimiento anterior partiendo de cada una de las fusiones encontradas hasta V-1 fusiones volvemos a tener el vocabulario inicial.

Para la segunda fusión se obtuvieron los siguientes resultados de la pérdida de información:

$$L_k(C_1, C_2) = 0{,}056017 \qquad L_k(C_2, C_3) = 0{,}059433 \qquad L_k(C_3, C_4) = 0{,}023733$$
$$L_k(C_4, C_5) = 0{,}027869 \qquad L_k(C_5, C_6) = 0{,}04937$$

Por lo que la segunda fusión sería C_3, C_4, obteniendose ahora cinco nuevas clases. Como se puede observar es un procedimiento un tanto tedioso, por lo que solo se mostrarán los resultados obtenidos para encontrar cada una de las fusiones.

A continuación se muestra un árbol que permite ver como se fueron dando las fusiones y que elementos hacen parte de cada clase, hasta llegar al vocabulario, aúnque en la práctica no se lleve hasta este caso. Para uso práctico se trabaja dando un número fijo de clases.

ÁRBOL OBTENIDO DESPUÉS DE V-1 FUSIONES

5.6 Clasificación bajo el criterio de la perplejidad

En este apartado se estudia otro método de clasificación por clases de palabras bajo el criterio de la perplejidad junto con el comportamiento de maximizar la función de maxima verosimilitud como criterio estadístico sobre un gran corpus.

Para ello se particiona el vocabulario en un número de clases **G**, es decir se construye una función que asigna una palabra **w** a su clase $G(w)$ $G : w \rightarrow G(w)$ o también $G : w \rightarrow g_w$.

En el caso de bigramas la probabilidad en clase se calcula mediante la siguiente expresión:

$$P(w_n|w_{n-1}) = P_0(w_n|g_{w_n}) \cdot P_1\left(g_{w_n}|g_{w_{n-1}}\right)$$

Donde:
$P_0(w_n|g_{w_n})$ es la probabilidad que w_n ocurra dado que g_{w_n} ya ha ocurrido, es decir la probabilidad que aparezca la palabra w_n dado que ya apareció alguna palabra perteneciente a la clase g_{w_n}.
$P_1\left(g_{w_n}|g_{w_{n-1}}\right)$ es la probabilidad que g_{w_n} ocurra dado que $g_{w_{n-1}}$ ya ha ocurrido, es decir la probabilidad que aparezca una palabra perteneciente a la clase g_{w_n} dado que ya apareció una palabra perteneciente a la clase $g_{w_{n-1}}$.
Luego se construye la función de máxima verosimilitud como criterio estadístico para obtener clases de palabras. La estimación de máxima verosimilitud para el modelo de bigramas mediante clases se obtiene por la ecuación:

$$F_{bi}(G) = \sum_{v,w} C(v,w) \cdot logp(w \mid v)$$
$$= \sum_{w} C(w) \cdot logP_0(w \mid g_w) + \sum_{g_v,g_w} C(g_v,g_w) \cdot logP_1(g_w \mid g_v)$$

Recordando que el subíndice *bi* especifica que se utiliza como medida estadística los bigramas. Donde las probabilidades $P_0(w \mid g_w)$ y $P_1(g_w \mid g_v)$ se pueden estimar como

las frecuencias relativas:

$P_0(w \mid g_w) = \frac{C(w)}{C(g_w)}$

$P_1(g_w \mid g_v) = \frac{C(g_v, g_w)}{C(g_v)}$

Al usar estos resultados se puede expresar la función logaritmo de la verosimilitud $F_{bi}(G)$ de la siguiente manera:

$$\sum_w C(w) \cdot logP_0(w \mid g_w) + \sum_{g_v, g_w} C(g_v, g_w) \cdot logP_1(g_w \mid g_v)$$

$$= \sum_w N(w) \cdot log\frac{C(w)}{C(g_w)} + \sum_{g_v, g_w} N(g_v, g_w) \cdot log\frac{C(g_v, g_w)}{C(g_v)}$$

Obteniendo las clases al optimizar el logaritmo de verosimilitud del bi-grama:

$$F_{bi} = \sum_{g_v, g_w} [C(g_v, g_w) \cdot logC(g_v, g_w)] - 2\sum_g [C(g) \cdot logC(g)] + \sum_w [C(w) \cdot logC(w)]$$ [MNL99]

Donde $C(g_v, g_w)$ es el conteo en el corpus de cuantas veces aparece una palabra de la clase g_v seguida de una palabra de la clase g_w, $C(g)$ es el conteo de clases de palabras g construidas y $C(w)$ es el conteo de palabras w observadas en el corpus.

La optimización se realiza mediante el algoritmo de intercambio [MNL99] siguiente:

Algoritmo de agrupamiento de Ney

Inicio
Empezar con alguna asignación inicial $w \rightarrow g_w$
Hacer
Para cada palabra w del vocabulario hacer
Para cada clase k hacer
Intercambiar palabra w de la clase g_w *a la clase k*
Actualizar conteos

Calcular perplejidad

Intercambiar la palabra w a la clase k con menor perplejidad

Hasta que el criterio de detención (F_{bi}) se cumpla.

Fin

El objetivo de este algoritmo es es encontrar un clasificación por clases talque la perplejidad de la clase modelo es la menor en todo el corpus de entrenamiento. El algoritmo emplea una técnica de optimización al mover cada uno de los elementos del vocabulario tentativamente a cada clase de palabra existente y asignándolo a la clase con menor perplejidad [NLM97]. El procedimiento se repite hasta que el criterio de parada se conozca, es decir que, $F\,(bi)$ cambie sustancialmente.

Se puede ver que el algoritmo inicia con una asignación donde cada palabra del vocabulario es su propia clase de palabra. Se consideran las $(G-1)$ palabras más frecuentes, y cada una de estas palabras se definen como su propia clase pero poco a poco al cambiar las palabras a diferentes clases y calcular la perplejidad vamos obteniendo un valor de F_{bi} con un cambio significativo y con una perplejidad mucho menor al calculado para la primera asignación.

Al observar que si se remueve una palabra w de su clase g_w afecta sólamente los conteos $C\,(g, g_w)$ y $C\,(g_w, g)$, y que la mayoría de estos conteos son cero, que se mantienen al agregar la palabra w a la clase k.

Ejemplo:
En el siguiente ejemplo se explicará en detalle los pasos del algoritmo de agrupamiento anterior:

Si se tiene el siguiente vocabulario y texto de entrenamiento:

$$\boldsymbol{V} = \{a, b, c, d, e, f, g\}$$

TEXTO DE ENTRENAMIENTO ($T = 23$)

$$a \quad b \quad c \quad a \quad d \quad a \quad f \quad a \quad b \quad b \quad c \quad a \quad b \quad f \quad a \quad c \quad e \quad a \quad b \quad b \quad f \quad a \quad g$$

La asignación inicial será que cada palabra del vocabulario será su clase así:

$$g_1 = \{a\}, \quad g_2 = \{b\}, \quad g_3 = \{c\},$$

$$g_4 = \{d\}, \quad g_5 = \{e\}, \quad g_6 = \{f\}, \quad g_7 = \{g\}$$

Donde se calcula el primer valor de maxima verosimilitud F_{bi} así:

$$F_{bi} = \sum_{g_v, g_w} \left[C\left(g_v, g_w\right) . log C\left(g_v, g_w\right) \right] - 2 \cdot 7 \cdot log 7 + 23 \cdot log 23$$
$$F_{bi} = 25,7362$$

Con los siguientes conteos:

$$C\left(g_1, g_2\right) = 4, C\left(g_2, g_3\right) = 2 \ ,$$
$$C\left(g_3, g_1\right) = 2, C\left(g_1, g_4\right) = 1, C\left(g_4, g_1\right) = 1$$
$$C\left(g_1, g_6\right) = 1, C\left(g_6, g_1\right) = 3, C\left(g_2, g_2\right) = 2,$$
$$C\left(g_2, g_6\right) = 2, C\left(g_1, g_3\right) = 1$$
$$C\left(g_3, g_5\right) = 1, C\left(g_5, g_1\right) = 1, C\left(g_1, g_7\right) = 1$$

y con una perplejidad inicial de:

$$PP\left(T\right) = \left(\prod_{i=1}^{7} P\left(g_i\right) \right)^{-\frac{1}{23}} = \left(P\left(g_1\right) \cdot P\left(g_2\right) \cdots P\left(g_7\right) \right)^{-\frac{1}{23}}$$
$$\left(\frac{C(g_1)}{T} \cdot \frac{C(g_2)}{T} \cdots \frac{C(g_7)}{T} \right)^{-\frac{1}{23}} = \left(8,5115 \times 10^{-7}\right)^{-\frac{1}{23}} = \left(1,2688 \times 10^{-7}\right)^{-\frac{1}{23}} = 1,9945$$

La palabra $g_1 = \{a\}$ se cambia a las clases g_2,g_3, g_4,g_5,g_6,
g_7 y en cada uno de los cambios calculamos la perplejidad y actualizamos los conteos $C\left(g_v, g_w\right)$ tal como se muestra a continuación:

Primer cambio de la primera palabra:

$$g_1 = \{a, b\}, \quad g_2 = \{c\}, \quad g_3 = \{d\}, \ g_4 = \{e\}, \quad g_5 = \{f\}, \ g_6 = \{g\}.$$

Con la siguiente perplejidad:

$$PP = \left(\frac{14}{T} \cdot \frac{3}{T} \cdot \frac{1}{T} \cdot \frac{1}{T} \cdot \frac{3}{T} \cdot \frac{1}{T}\right)^{-\frac{1}{23}} = (8,5115 \times 10^{-7})^{-\frac{1}{23}} = 1,8361$$

en el siguiente cambio:

$$g_1 = \{a, c\}, \quad g_2 = \{b\}, \quad g_3 = \{d\}, \quad g_4 = \{e\}, \quad g_5 = \{f\}, \quad g_6 = \{g\}.$$

y la siguiente perplejidad:

$$PP = \left(\frac{11}{T} \cdot \frac{6}{T} \cdot \frac{1}{T} \cdot \frac{1}{T} \cdot \frac{3}{T} \cdot \frac{1}{T}\right)^{-\frac{1}{23}} = (1,3375 \times 10^{-6})^{-\frac{1}{23}} = 1,8004$$

y en el siguiente:

$$g_1 = \{a, d\}, \quad g_2 = \{b\}, \; g_3 = \{c\}, \quad g_4 = \{e\}, \; g_5 = \{f\}, \; g_6 = \{g\}.$$

y la siguiente perplejidad:

$$PP = \left(\frac{9}{T} \cdot \frac{6}{T} \cdot \frac{3}{T} \cdot \frac{1}{T} \cdot \frac{3}{T} \cdot \frac{1}{T}\right)^{-\frac{1}{23}} = (3,2829 \times 10^{-6})^{-\frac{1}{23}} = 1,7315$$

y en el siguiente:

$$g_1 = \{a, e\}, \quad g_2 = \{b\}, \; g_3 = \{c\}, \quad g_4 = \{d\}, \quad g_5 = \{f\}, \; g_6 = \{g\}.$$

y la siguiente perplejidad:

$$PP = \left(\frac{9}{T} \cdot \frac{6}{T} \cdot \frac{3}{T} \cdot \frac{1}{T} \cdot \frac{3}{T} \cdot \frac{1}{T}\right)^{-\frac{1}{23}} = (3,2829 \times 10^{-6})^{-\frac{1}{23}} = 1,7315$$

y en el siguiente:

$$g_1 = \{a, f\}, \quad g_2 = \{b\}, \quad g_3 = \{c\}, \; g_4 = \{d\}, \quad g_5 = \{e\}, \; g_6 = \{g\}.$$

y la siguiente perplejidad:

$$PP = \left(\frac{11}{T} \cdot \frac{6}{T} \cdot \frac{3}{T} \cdot \frac{1}{T} \cdot \frac{1}{T} \cdot \frac{1}{T}\right)^{-\frac{1}{23}} = (1,3375 \times 10^{-6})^{-\frac{1}{23}} = 1,8004$$

y en el siguiente:

$$g_1 = \{a, g\}, \quad g_2 = \{b\}, \quad g_3 = \{c\}, \quad g_4 = \{d\}, \quad g_5 = \{e\}, \quad g_6 = \{f\}.$$

y la siguiente perplejidad:

$$PP = \left(\frac{9}{T} \cdot \frac{6}{T} \cdot \frac{3}{T} \cdot \frac{1}{T} \cdot \frac{1}{T} \cdot \frac{3}{T}\right)^{-\frac{1}{23}} = (3,2829 \times 10^{-6})^{-\frac{1}{23}} = 1,7315$$

A continuación se intercambia la palabra b perteneciente a $g_2 = \{b\}$ y así sucesivamente hasta encontrar las nuevas clases con la menor perplejidad. La cual se presenta en el cambio:

$$g_1 = \{a\},\ g_2 = \{b\},\ \ g_3 = \{c\},\ \ g_4 = \{f\},\ \ g_5 = \{e, d\},\ \ g_6 = \{g\}.$$

Siendo estas las nuevas clases y con un nuevo valor de maxima verosimilitud F_{bi}

$$F_{bi} = \sum_{g_v, g_w} \left[C\left(g_v, g_w\right) .logC\left(g_v, g_w\right)\right] - 2 \cdot 6 \cdot log6 + 23 \cdot log23$$
$$F_{bi} = 28,2297$$

Ahora con estas nuevas clases realizamos de nuevo los cambios de cada palabra del vocabulario en cada una de las 6 clases nuevas, iniciando con la palabra a de la clase $g_1 = \{a\}$, encontrando nuevas fusiones y un número menor de clases. Las cuales se pueden observar a continuación:

5 clases :

$$g_1 = \{a\},\ \ g_2 = \{b\},\ g_3 = \{c\},\ \ g_4 = \{f\},\ g_5 = \{e, d, g\}.$$

De nuevo con la palabra a perteneciente a la clase g_1 realizamos los cambios para encontrar la menor perplejidad.

4 clases:

$$g_1 = \{a\},\ \ g_2 = \{b\},\ \ g_3 = \{c, f\},\ \ g_4 = \{e, d, g\}.$$

3 clases:

$$g_1 = \{a\},\ g_2 = \{c, f\},\ \ g_3 = \{b, d, e, g\}.$$

Y por último estas son las dos clases finales calculadas:

$$g_1 = \{a, c, f\},\ \ g_2 = \{b, d, e, g\}.$$

ÁRBOL OBTENIDO DESPUÉS DE V-1 FUSIONES

Capítulo 6

INTEGRACIÓN DE N-GRAMAS CON GIP

La aplicación de las GIP para la modelización del lenguaje presenta ciertos problemas que requieren soluciones eficientes. En primer lugar, se plantean problemas de interpretación; es decir, cómo determinar la relación entre las palabras del lenguaje. En segundo lugar, está el problema de integración; es decir, cómo realizar la interpretación de forma eficiente. Para abordar el problema de interpretación se han hecho propuestas basadas en el cálculo de la probabilidad del prefijo de una cadena, mientras que para tratar el problema de la integración se han propuesto un modelos híbridos que combinan modelos de n-gramas con modelos estructurales. En este capítulo se van a estudiar y realizar propuestas relacionadas com ambos problemas.

6.1 Las GIP en el Modelado del Lenguaje

Un problema importante asociado a la modelización del lenguaje consiste en evaluar:

$$\Pr(w_k \mid w_1 w_2 ... w_{k-1}) \tag{6.1}$$

es decir, determinar la probabilidad de que en el instante k se observe la palabra w_k suponiendo que previamente hemos observado la secuencia de palabras $w_1 w_2 ... w_{k-1}$.

Para calcular la expresión (6.1) es habitual simplificarla haciendo restricciones sobre su historia, esto es, sobre $w_1 w_2 ... w_{k-1}$. La restricción más común para realizar el cálculo

es limitar la historia anterior a un pequeño conjunto de palabras. Los modelos de n-gramas permiten aproximar la expresión (6.1) calculando la probabilidad considerando únicamente las n-1 palabras anteriores, ya que el número de parámetros a estimar crece exponencialmente con el valor de n, por lo que sólo es posible considerar valores pequeños de n; en consecuencia, sólo son capaces de recoger dependencias locales en función del valor de n.

$$\Pr(w_k \mid w_{k-n+1}...w_{k-1}) \tag{6.2}$$

Otra forma de aproximar la expresión (6.1) consiste en imponer restricciones sintácticas sobre la historia anterior para limitar las posibles relaciones entre palabras. Las GIP son una alternativa adecuada para representar relaciones sintácticas entre las palabras. Con este tipo de modelos la expresión (6.1) quedaría:

$$\Pr(w_k \mid w_1 w_2...w_{k-1}, G_p) \tag{6.3}$$

La probabilidad de que la palabra w_k se observe, se determina a partir de la relación que define una GIP G_p entre la subcadena $w_1 w_2...w_{k-1}$ y dicha palabra. Esta aproximación presenta varias ventajas: en primer lugar, este tipo de modelos permite representar de forma compacta y eficiente, relaciones a largo término entre las palabras de la cadena; en segundo lugar; como se ha visto en capítulos anteriores, existen algoritmos potentes que permiten la estimación de modelos a partir de una muestra de aprendizaje; y en tercer lugar; existen algoritmos robustos que permiten la evaluación de la expresión (6.3) y posibilitan una integración eficiente de este tipo de modelos. Sin embargo, el coste de los algoritmos hace que la utilización se vea enormemente limitada. Además, para tareas reales complejas es necesario un elevado número de parámetros, y en consecuencia, una gran cantidad de datos para estimarlos adecuadamente. Estos problemas se acentúan en tareas reales con grandes vocabularios, por lo que la aplicación de las GIP en el Modelado de lenguaje sin apoyo de algún otro mecanismo resulta poco adecuado.

En trabajos recientes se han propuesto modelos híbridos que combinan modelo de n-gramas con modelos estructurales para aproximar la expresión (6.1). Ambos tipos de modelos se combinan para aportar cada uno el nivel de especialidad que maneja. El modelo de n-gramas da cuenta de las relaciones entre palabras del léxico, donde quedan mejor representadas las restricciones locales. Por su lado el modelo estructural da cuenta de las restricciones entre categorías de palabras, donde quedan mejor representadas las restricciones a más largo término.

Esta idea resulta muy atractiva, dado que trabajar con un modelo estructural a nivel de categorías implica varias ventajas. Al agrupar las palabras en un conjunto relativamente pequeño de categorías, se reduce notablemente el conjunto de parámetros necesarios para representar las relaciones sintácticas. De esta forma se reduce la talla del modelo y es posible estimarlo mejor. Un inconveniente de esta propuesta es la necesidad de disponer de un corpus de datos etiqueteado con categorías para estimar los parámetros del modelo.

A continuación vamos a presentar una propuesta en la cual la probabilidad de una palabra se calcula con un modelo híbrido que combina el modelo de n-gramas a nivel de palabras con una GIP a nivel de categorias.

6.2 Modelo híbrido de n-gramas y GIP

En el modelo que vamos a plantear, la probabilidad de una palabra se define así [BS00]:

$$\begin{aligned} &\Pr(w_k \mid w_1 w_2 ... w_{k-1}) \\ &\qquad = \lambda \ \Pr\left(w_k \mid w_{k-n} ... w_{k-1}\right) + (\ 1 - \lambda\ ) \Pr\left(w_k \mid g_{w_1} ... g_{w_{k-1}}\right) \end{aligned} \tag{6.4}$$

El primer sumando de esta expresión es un modelo de n-gramas que recoge las dependencias locales a nivel de palabras.

En el segundo sumando, se hace uso de una función de etiquetado que asocia una categoría a cada palabra. La expresión $\Pr\big(w_k \mid g_{w_1}...g_{w_{k-1}}\big)$ pretende recoger las relaciones estructurales a largo término entre las palabras del texto.

En la expresión (6.4), $0 \leq \lambda \leq 1$, es un factor de peso que depende de la tarea.

Suponiendo que tenemos un método para etiquetar las palabras, vamos a separar el problema del etiquetado del cómputo de $\Pr\big(w_k \mid g_{w_1}...g_{w_{k-1}}\big)$. De esta forma el segundo sumando lo aproximaremos como:

$$\begin{aligned}\Pr\big(w_k \mid g_{w_1}...g_{w_{k-1}}\big) &= \Pr(w_k \mid g_1...g_{k-1}). \\ &= \Pr(w_k \mid g_k)\Pr(g_k \mid g_1...g_{k-1})\end{aligned}$$

donde g_i, $1 \leq \mathrm{i} \leq \mathrm{k}$, denotará la etiqueta asociada a la i-ésima palabra.

La expresión anterior se interpreta en los siguientes términos: la probabilidad de que se dé una palabra se aproxima a partir de la probabilidad de la categoría a la que pertenece (considerando un modelo gramatical de categorías) multiplicado por la probabilidad de que en ésta categoría se dé esta palabra. Las simplificaciones introducidas permiten evaluar todas las expresiones como describimos a continuación.

El valor $\Pr(w_k \mid g_k)$ corresponde a la probabilidad de la clasificación de la palabra w_k en la categoría g_k. Para la estimación de los parámetros de los correspondientes modelos es necesario disponer de un conjunto de datos etiquetados en términos de categorias. De esta forma, el valor $\Pr(w_k \mid g_k)$ se estima a partir de las frecuencias absolutas del corpus. Para una palabra w del vocabulario, la probabilidad de que esta palabra se clasifique en la categoría g, será:

$$\Pr(w \mid g) = \frac{N(w,g)}{\sum_{w'} N(w',g)}$$

donde N(w,g) es el número de veces que la palabra w ha sido etiquetada con la etiqueta g.

La evaluación de la expresión $\Pr(g_k \mid g_1...g_{k-1})$ puede realizarse con el algoritmo LRI:

$$\Pr(g_k \mid g_1...g_{k-1}) = \bar{Pr}\,(g_k \mid g_1...g_{k-1}, G_p)$$

o bien, puede aproximarse con el algoritmo VLRI

$$\Pr(g_k \mid g_1...g_{k-1}) = \hat{\Pr}\,(g_k \mid g_1...g_{k-1}, G_p)$$

En la expresiones anteriores, la GIP G_p puede estimarse con alguno de los métodos estudiados en los capítulos anteriores. El uso de categorías permite reducir considerablemente el número de parámetros de la GIP, lo cual posibilita la aplicación de los algoritmos LRI y VLRI con un coste razonable.
Con todo ello la expresión (6.4) se aproxima como:

$$\begin{aligned} &\Pr\,(w_k \mid w_1 w_2...w_{k-1}) \\ &\quad = \lambda\ \Pr\,(w_k \mid w_{k-n}...w_{k-1}) + (\ 1 - \lambda\ )\ \Pr\,(w_k \mid g_k)\ \Pr\,(g_k \mid g_1...g_{k-1}, G_p) \end{aligned} \tag{6.5}$$

En la expresión (6.5) Jelinek y Lafferty [JL91] plantean el cálculo de $\Pr\,(g_k \mid g_1...g_{k-1}, G_p)$ como:

$$\Pr\,(g_k | g_1...g_{k-1}, G_p) = \frac{\Pr\,(S \stackrel{*}{\Longrightarrow} g_1.....g_{k-1}g_k..... \mid G_p)}{\Pr\,(S \stackrel{*}{\Longrightarrow} g_1.....g_k..... \mid G_p)} \tag{6.6}$$

Esta aproximación se caracteriza porque el cómputo de la siguiente categoría considera toda la historia anterior, aportando de esta forma la mayor cantidad posible de información. Para computar la expresión (6.6) se presenta el algoritmo LRI que permite calcular $\Pr\,(S \stackrel{*}{\Longrightarrow} x_1.....x_k..... \mid G_p)$, esto es, la probabilidad de generar el prefijo $x_1.....x_k.....$

6.3 Probabilidad de una subcadena inicial: el algoritmo LRI

Sea G_p una GIP y x una cadena de $L(G_p)$. Se define la probabilidad de que el no terminal $A \in N$ derive directamente en el no terminal $B \in N$, como no terminal más a la izquierda en una de sus reglas, como:

$$R(A \longrightarrow B) = \sum_{C \in N} p\ (A \longrightarrow BC) \tag{6.7}$$

Se define la probabilidad de que B sea el no terminal más a la izquierda, en cualquier forma sentencial que se pueda derivar a partir de A, como:

$$\begin{aligned} T(A \Longrightarrow B) &= R(A \longrightarrow B) + \sum_{C_1 \in N} R(A \longrightarrow C_1) R(C_1 \longrightarrow B) + \ldots\ldots \\ &\quad + \sum_{C_1 \ldots C_k \in N} R(A \longrightarrow C_1) R(C_1 \longrightarrow C_2) \ldots R(C_k \longrightarrow B) + \ldots \\ &= \sum_{\alpha \in (N \cup \sum)^+} \Pr\ (\ A \stackrel{*}{\Longrightarrow} B\alpha \mid G_p\ ). \end{aligned} \tag{6.8}$$

Se define la probabilidad de que $BC,\ C \in N$, pueda ser la subcadena inicial de todas las formas sentenciales derivadas de A como:

$$T(A \Longrightarrow BC) = p\ (A \longrightarrow BC) + \sum_{D \in N} T(A \Longrightarrow D)\ p\ (D \longrightarrow BC)$$

Finalmente se define la probabilidad de generación de cadenas a partir de A cuya subcadena inicial sea $x_i \ldots\ldots x_j$ como:

$$e\ (A << i, j\ ) = \Pr\ (A \stackrel{*}{\Longrightarrow} x_i \ldots . x_j \ldots . \mid G_p).$$

Con todo ello, el algoritmo LRI se define como:

$$e\ (A << i, i\ ) = p\ (A \longrightarrow x_i) + \sum_{B \in N} T(A \Longrightarrow B)\ p\ (B \longrightarrow x_i). \tag{6.9}$$

$$e\ (A <<i,j\ ) = \sum_{B,C\in N} T(A \Longrightarrow BC) \sum_{k=i}^{j-1} e\ (B<i,k>\ )\ e\ (C <<k+1,j).$$

$$1 \leq i < j.$$

Con lo que Pr $(x_1...x_k... \mid G_p) = e(S <<1,k)$.

El coste del algoritmo LRI es dos veces el coste del algoritmo Inside y por tanto $O(|x|^3\,|P|)$.

6.4 Probabilidad de la mejor derivación que genera una subcadena inicial: el algoritmo VLRI

A continuación vamos a presentar un algoritmo para calcular la probabilidad de la mejor derivación que genera un prefijo. Este algoritmo se basa en la aplicación de un esquema de Viterbi al algoritmo LRI, por lo que se le denomina algoritmo VLRI.

Para la evaluación eficiente de esta probabilidad, $\hat{\Pr}(g_k \mid g_1...g_{k-1}, G_p)$, se propone un algoritmo similar al algoritmo LRI pero basado en un esquema de Viterbi.

Se define la máxima probabilidad de que $B \in N$ sea inicial de $A \in N$, a partir de una de sus reglas, como:

$$\hat{R}(A \rightarrow B) = \max_{C\in N} p(A \rightarrow BC) \tag{6.10}$$

Análogamente, se define la probabilidad de la mejor derivación en la que partiendo de A, B es el símbolo de más a la izquierda como:

$$\hat{T}(A \Rightarrow B) = \max(\hat{R}(A \rightarrow B), \max_{C_1\in N} \hat{R}(A \rightarrow C_1)\hat{R}(C_1 \rightarrow B)), ...,$$
$$\max_{C_1,..,C_k\in N}(\hat{R}(A \rightarrow C_1)...\hat{R}(C_k \rightarrow B)), ...)$$

$$= \max_{\alpha\in(N\cup\sum)^+} \hat{Pr}(A \stackrel{*}{\Rightarrow} B\alpha \mid G_p) \tag{6.11}$$

Se define la probabilidad de la mejor derivación en la que partiendo de A, BC ($C \in N$), sea subcadena inicial como:

$$\hat{T}(A \Rightarrow B) = \text{máx}(p(A \rightarrow BC), \underset{D \in N}{\text{máx}}(\hat{T}(A \Rightarrow D)p(D \rightarrow BC))). \tag{6.12}$$

Finalmente se define la probabilidad de la mejor derivación que genera la subcadena inicial $x_i.....x_j$ a partir de A dado G_p como:

$$\hat{e}(A << i, j) = \hat{\Pr}(x_i.....x_j... \mid G_p).$$

Con todo ello, el algoritmo VLRI propuesto será:

$$\hat{e}(A << i, i) = \text{máx}(p(A \rightarrow x_i), \underset{B \in N}{\text{máx}}(\hat{T}(A \Rightarrow B)p(B \rightarrow x_i)),$$

$$\hat{e}(A << i, j) = \underset{B,C \in N}{\text{máx}}(\hat{T}(A \Rightarrow BC) \tag{6.13}$$

$$\underset{k=i,..,j-1}{\text{máx}}(\hat{e}(B < i, k >)\hat{e}(C << k+1, j >)) \qquad 1 \leq i < j$$

Con lo que $\hat{\Pr}(x_1.....x_k... \mid G_p) = \hat{e}(A << i, j)$

El coste temporal del algoritmo es $O(|N|^3)$, mientras que su coste espacial es $O(|N|^2)$

El algoritmo VLRI puede utilizarse para aproximar la expresión (6.3) según la expresión:

$$\hat{Pr}\,(g_k|g_1...g_{k-1}, G_p) = \frac{\hat{\Pr}\,(S \overset{*}{\Longrightarrow} g_1.....g_{k-1}g_k..... \mid G_p)}{\hat{\Pr}\,(S \overset{*}{\Longrightarrow} g_1.....g_k..... \mid G_p)}$$

Una característica que diferencia al algoritmo VLRI del LRI es que permite obtener los argumentos que dan lugar a la maximización, y por lo tanto permite obtener la mejor interpretación de la cadena de entrada en cualquier momento del proceso de análisis.

Capítulo 7

RESULTADOS EXPERIMENTALES EN LA CLASIFICACIÓN DE UN VOCABULARIO POR CLASES

En este capítulo se presentará los resultados experimentales que se produjeron al implementar dos algoritmos de clasificación del vocabulario en clases de palabras, a fin aplicar el modelo híbrido de n-gramas.

7.1 Medida de evaluación del modelo

La medida de evaluación del modelo será la perplejidad y la infomación mutua promedio sobre un conjunto de prueba. La perplejidad es una medida aceptada como medida de calidad del modelo.

La perplejidad mide la capacidad del modelo para modelizar el lenguaje, determina que tan bien el modelo asigna probabilidades a las frases de algún texto no visto. Desde un punto de vista formal, la perplejidad es el inverso de la media geométrica de un factor de diversidad del lenguaje, lo que significa que un modelo con perplejidad P, tiene la misma dificultad que otro en el cual cada palabra puede ser seguida por P diferentes palabras con igual probabilidad.

En el caso de los algoritmos implementados, la perplejidad sobre un conjunto de prueba S, se define como:

$$Pr(S) = \left[\prod_{s \in S} P(s) \right]^{-\frac{1}{|S|}}$$

donde S es el conjunto de prueba formado por frases s y $|S|$ es el número de palabras del Vocabulario. La perplejidad es fácil de calcular, es una medida estadística bien comprendida y se calcula rápidamente cuando se implementa. Se ha demostrado que el uso de la perplejidad es viable como medida de la capacidad expresiva de cualquier modelo.

A continuación se describirá el corpus de datos sobre el cual se han realizado los experimentos, luego como se ha establecido el parámetro experimental necesario para el entrenamiento del modelo como es el tamaño de la muestra. Después se describe los resultados de la clasificación del Vocabulario por diferentes métodos considerados en éste trabajo, por último se describirán las perplejidades de los modelos en los cuales se implementó las diferentes clasificaciones realizadas con cada uno de los algoritmos.

7.2 Corpus de datos

El corpus utilizado es una parte del corpus denominado *Wall Street Journal* (WSJ), procesado en el proyecto Penn TreeBank [MSM93]. Se decidió hacer uso de ésta base de datos por que es ampliamente conocida y divulgada en trabajos similares; y además contiene gran cantidad de datos.

El WSJ es una colección de textos de ediciones de finales de los años 80 del periódico *Wall Street Journal*. El conjunto de datos comprende un millón de palabras que se encuentran en 25 directorios enumerados del 00 al 24. Este corpus está analizado y etiquetado automáticamente y revisado de forma manual. El etiquetado es de dos tipos: un etiquetado léxico con partes de habla y un etiquetado sintáctico. El tamaño del vocabulario es de aproximadamente 49.000 palabras, el vocabulario léxico consiste en 45 etiquetas y el sintáctico en 14 etiquetas.

Puesto que el tamaño del vocabulario del corpus original resultaba demasiado grande para los experimentos que se pretendía realizar [BDP90] se decidió extraer un corpus menor obtenido de la siguiente forma:

1. Se seleccionaron aleatoriamente 6 directorios para texto de entrenamiento y de prueba los seleccioonados son 03 - 06 - 07 - 11 - 19 - 21.

2. Del texto seleccionado anteriormente, una vez depurado, se extraen las primeras 30000 palabras que forman el Corpus de Entrenamiento (CE), de donde se registra un vocabulario de 4928.

3. De este mismo texto, una vez depurado, se extraen las últimas 13000 palabras que forman el Corpus de Prueba (CP).

En la siguiente tabla (tabla 7.1) se describen las características principales de los dos conjuntos que hemos utilizado, denominados Conjunto de Prueba (CP) y Conjunto de Entrenamiento (CE).

Tabla. 7.1 Corpus de Prueba y de Entrenamiento

Conjuntos	Número de frases	Número de palabras	Vocabulario
CP	1251	13000	
CE	3112	30000	4923

Un ejemplo de frase en WSJ se muestra en la siguiente figura:

Ejemplo 7.1:

```
(  (S
  (NP – SBJ
    (NP  (NNP Pierre)  (NNP Vinken)  )
    (ADJP
      (NP  (CD 61)  (NNS years)  )
      (JJ old)  ))
  (VP  (MD will)
    (VP  (VB join)
      (NP  (DT the)  (NN board))
      (PP – CLR(IN as)
    (NP  (DT a)  (JJ nonexecutive)  (NN director)  ))
      (NP – TMP  (NNP Nov.)  (CD 29)  )))))
```

Figura 7.1: La frase "Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29". Analizada y etiquetada en el proyecto Penn Treebank.

7.3 Algoritmos de clasificación

Debido al costo computacional para la estimación de una gramática por el elevado número de palabras a procesar, se emplearon dos algoritmos para agrupar en clases el vocabulario. Los algoritmos que se tratan es este trabajo son: Algoritmo de Ney y Algoritmo de Della Pietra, el cual reduce los parámetros en la estimación de la gramática.

7.3.1 Algoritmo de Della Pietra

Se presenta en la siguiente tabla las palabras agrupadas en las clases 189, 190 y 191 después de clasificar en 500 clases el vocabulario del Corpus de Entrenamiento (4928 palabras), haciendo uso del Algoritmo de Della Pietra.

Tabla. 7.2 Algunas clases de Palabras

CLASES	PALABRAS
32	BAN, BANCORP
33	BANK
34	BANKAMERICA, BANKER, BANKERS

7.3.2 Algoritmo de Ney

Se presenta en la siguiente tabla las palabras agrupadas en las clases 189, 190 y 191 después de clasificar en 500 clases el vocabulario del Corpus de Entrenamiento (4928 palabras), haciendo uso del Algoritmo de Ney.

Tabla. 7.3 Algunas clases de Palabras

CLASES	PALABRAS
32	COMMERCIAL, DEFICIT, DEFENSE, THRIFT
33	MESA, SPENDING, PREDICTED, STATESWEST
34	RUBLES, AFL, FORWARD, AGREES, CAPACITY, PAYMENTS, CIO, HAHN

7.4 Perplejidad en los modelos de n-gramas de clases

Se presenta en la siguiente tabla los valores de perplejidad obtenidos para cada una del agrupamiento con cada uno de los algoritmos:

Tabla. 7.4 Cálculo de la Perplejidad

MODELO	PERPLEJIDAD
NEY	104.1544
DELLA PIETRA	129.7923

CONCLUSIONES

- El objetivo principal de este trabajo fué comparar el desempeño de dos modelos de lenguaje híbridos obtenidas a partir de la combinación de un modelo de bi - gramas y una gramática incontextual.
- Se estudió en detalle los algoritmos de entrenamiento para la estimación de gramáticas incontextuales Inside - Outside, Viterbi Score.
- Se estudiaron en detalle algoritmos de agrupación en clases para modelos de bigramas.
- Se estudió el método para incorporar mediante combinaciones lineales gramáticas y modelos de n - gramas.
- En los experimentos realizados sobre un conjunto del *Wall Street Journal* (WSJ), se encontró que el modelo en el cual la clasificación se realiza mediante el algoritmo de Ney tiene mayor capacidad expresiva que el desarrollado por Della Pietra.
- La clasificación obtenida por el algoritmo de Ney representa de mejor manera las restricciones locales del Corpus de Entrenamiento, con lo cual se reduce el conjunto de parámetros necesarios para representar las relaciones sintácticas en la aplicación del modelo híbrido de n - gramas y GIP.
- Experimentalmente se determinó el coste computacional de los dos algoritmos determinándose que el algoritmo de Ney tiene un coste computacional 19,75 % menor con relación al Della Pietra.
- La perplejidad del modelo híbrido bigramas - GIP es directamente proporcional a la de los modelos de n - gramas, de donde se deduce que el modelo híbrido con agrupaciones de clases por el algoritmo de Ney ofrece mayor información estructural.

BIBLIOGRAFÍA

[HU79] HOPCROFT Jhon E. Y ULLMAN Jefrey D., *INTRODUCCIÓN A LA TEORÍA DE AUTÓMATAS, LENGUAJES Y COMPUTACIÓN*, ADDISON WESLEY, 1979.

[SG86] SANCHÍS F. J. Y GALÁN C., *COMPILADORES: TEORÍA Y CONSTRUCCIÓN*, ED. PARANINFO, 1986.

[Bau88] L.E. Baum., *An inequality and associated maximization technique in statistical estimation for probabilistic functions of models parameters.*, In Proc. of ICASSP'88, pages 493 - 496, 1988.

[IMB97] ISASI P. , MARTINEZ P. Y BORRAJO D. , *LENGUAJES, GRAMÁTICAS Y AUTÓMATAS, UN ENFOQUE PRÁCTICO*, ADDISON-WESLEY, 1997.

[VAPP03] VIDAL Enrique, ALFONS Juan, PICÓ David Y PAREDES Roberto, *APRENDIZAJE Y PERCEPCIÓN TEMA 6: MÉTODOS SINTÁCTICOS / ESTRUCTURALES MODELOS DE MARKOV*, Universidad Politécnica De Valencia, 2003.

[SM01] SANCHÍS DE MIGUEL María Araceli, *INFORMÁTICA TEÓRICA TEMA 3: GRAMÁTICAS FORMALES*, Universidad Carlos III De Madrid, 2001.

[SJA99] SÁNCHEZ Joan Andreu, *ESTIMACIÓN DE GRAMÁTICAS INCONTEXTUALES PROBABILÍSTICAS Y SU APLICACIÓN EN MODELIZACIÓN DE LENGUAJES*, Universidad Politécnica De Valencia, 1999.

[AJ01] ACOSTA Javier, *MODELADO DE LENGUAJES CON REDES NEURONALES RECURRENTES*, Universidad Politécnica De Valencia, 2001.

[AFA01] AMAYA Fredy Angel, *ALGUNOS APORTES A LOS MODELOS DE LENGUAJE DE MÁXIMA ENTROPÍA DE FRASE COMPLETA*, Universidad Politécnica De Valencia, 2001.

[BDP90] BROWN Peter, DELLA PIETRA Vincent, DESOUZA Peter Y LAI Jenifer, *CLASS-BASED N-GRAM MODELS OF NATURAL LANGUAGE*, Computational Linguistics, Página Web acl.ldc.upenn.edu/, 1990.

[BS00] BENEDÍ José Miguel Y SÁNCHEZ Joan Andreu, *COMBINATION OF N-GRAMS AND STOCHASTIC CONTEXT-FREE GRAMMARS FOR LANGUAGE MODELING*, International conference on computational linguistics, Página web acl.ldc.upenn.edu/ Universidad Politécnica de Valencia, 2000.

[CG98] CHEN Stanley Y GOODMAN Joshua, *AN EMPIRICAL STUDY OF SMOOTHING TECHNIQUES FOR LANGUAGE MODELING*, Technical report TR-10-98, Center for Research in Computing Technology, Harvard University, Página Web research.microsoft.com/, 1998.

[MNL99] MARTIN Sven, NEY Hermman, HAMACHER Christoph, LIERMANN Jörg Y WESSEL Frank, *ASSESSMENT OF SMOOTHING METHODS AND COMPLEX STOCHASTIC*, Universidad de Tecnología Ahornstraße, Página Web www-i6.informatik.rwth-aachen.de, Alemania, 1999.

[NLM97] NEY Hermann, LIERMANN Jörg Y MARTIN Sven, *ALGORITHMS FOR BIGRAM AND TRIGRAM WORD CLUSTERING*, Proc. de EUROSPEECH, Conferencia europea sobre comunicación y tecnología, Página Web http://else.hebis.de/ comunicación presentada al ESCA por Elsevier Science, 1997.

[MSM93] MARCUS M, SANTORINI B. Y MARCINKIEWICZ M, *BUILDING A LARGE ANNOTATES CORPUS OF ENGLISH: THE PENN TREEBANK*, Computational Linguistics, 19(2): 313-330, 1993.

[BS03] BARRACHINA Sergio, *TÉCNICAS DE AGRUPAMIENTO BILINGÜE APLICADAS A LA INFERENCIA DE TRADUCTORES*, Universidad Jaume I, Castellón, 2003.

[JL91] JELINEK F Y LAFFERTY J.D, COMPUTATION OF THE PROBABILITY OF INITIAL SUBSTRING GENERATION BY STOCHASTIC CONTEXT-FREE GRAMMARS, Computational Linguistics, 1991.

[AM95] WEISS, Mark *ESTRUCTURA DE DATOS Y ALGORITMOS*, Addison - weisley iberoamericana. 1995.