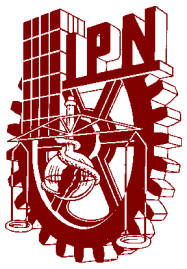

INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Resolución automática de la homonimia morfológica para el español

TESIS

Que para obtener el grado de:

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A

José Ernesto Gómez Balderas

Director:

Dr. Grigori Sidorov

Codirector:

Dr. Héctor Jiménez Salazar

México D.F. Abril de 2007

Agradecimientos:

A mi madre *Evangelina Rosario Balderas Cruz*.

A mi padre *Víctor Gómez Naranjo*.

A mi hermano *Víctor Hugo Gómez Balderas*.

A la familia Balderas Marañon: *Lilia, Ciro, Liliana, Ciro A. Mariana.*

A mis familiares.

A mis profesores

Al *Dr. Grigori Sidorov*, por su valiosa orientación en mi formación profesional y en el desarrollo de esta tesis.

A mi codirector, *Dr. Héctor Jiménez Salazar*, por su apoyo en la dirección de esta tesis.

Contenido

Resumen

Presentamos un modelo y un método para resolver el problema de etiquetación de categorías gramaticales usando solamente información gramatical. El problema consiste en la asignación de manera correcta, de la etiqueta de una categoría gramatical a cada palabra. El problema no es simple debido a la gran extensión de la homonimia morfológica, por ejemplo la palabra *trabajo* puede ser verbo o sustantivo, etc. La resolución de todos los tipos de homonimia se realiza por un lector humano de acuerdo al contexto de las palabras.

Tratamos de encontrar una forma para la resolución automática de la homonimia morfológica, usando la metodología de las redes neuronales y el algoritmo de retropropagación. Experimentamos con datos en español.

El sistema de etiquetación implementado utiliza información del contexto correspondiente a la categoría gramatical. Esta información es representada en forma de etiqueta. Las aplicaciones potenciales de este sistema existen en muchas áreas incluyendo reconocimiento del habla, síntesis del habla, traducción, desambiguación y recuperación de información.

Usamos el conocimiento explicito de las etiquetas de los contextos antecedentes y subsecuentes para representar las entradas de las redes neuronales. En el experimento, usamos distintas redes neuronales con el propósito de asignar etiquetas de clases gramaticales a palabras desconocidas o escoger la etiqueta correcta de un conjunto de etiquetas asignadas por un analizador morfológico. Las redes neuronales usan un número diferente de etiquetas de contextos hacia la izquierda y hacia la derecha (hasta 3). La decisión final es tomada en base a la "votación" de las redes. De la misma manera, llevamos a cabo más experimentos usando estas redes con pesos adicionales.

Los resultados de este trabajo están divididos en dos partes. En la primera parte, analizamos los resultados de la ejecución de la etiquetación de categorías gramaticales, es decir asignamos etiquetas en base al contexto. En la segunda parte, analizamos los resultados obtenidos de la resolución automática de la homonimia morfológica, es decir escogimos la etiqueta correcta de un conjunto de posibles etiquetas. El método propuesto tiene un alto nivel de precisión cuando existe información completa en los contextos anteriores y posteriores de hasta 46% para la primera tarea y hasta 76% para la segunda tarea.

La principal contribución de este trabajo comparado con otras propuestas es el uso de etiquetas de contextos solamente, ignorando la información léxica. Esto nos permite reducir significativamente el espacio de decisión.

Abstract

We present a model and a method for solving the problem of part of speech tagging using the grammatical information only. The problem consists in assigning the correct tag of a grammatical category to each word. The problem is not simple due to the widely spread morphological homonymy, for example, *work* can be verb or noun, etc. The resolution of all kinds of homonymy is carried out by a human reader according to words from the context.

We try to find a way for automatic resolution of morphologic homonymy using neural networks methodology with back propagation algorithm. We experimented on Spanish data.

We implemented tagging system that uses the corresponding information of parts of speech of the context. This information is represented in form of tags. Potential applications of this system exist in many areas, including speech recognition, speech synthesis, translation, disambiguation and information retrieval.

We used the explicit knowledge of antecedent and subsequent contexts tags for representing input of neural networks. In the experiment, we used various neural networks, with the purpose of assigning grammatical class tags to unknown words or to choose the correct tag from a set of tags assigned by a morphological analyzer. The neural networks use different number of context tags to the left and to the right (till 3). The final decision is taken on the basis of "voting" of the networks. In the same manner, we conducted more experiments using these networks with additional weights.

The results of this work are divided in two parts. In the first part, we analyze the results of performance of a part of speech tagging, i.e., we assign the tags on the basis of the context. In the second part, we analyze results in automatic morphologic homonymy resolution, i.e., we choose the correct tag of the set of possible tags. Proposed method has high precision, when there is complete information available of the previous and subsequent contexts, namely, till 46% for the first task, and till 76% for the second task.

The main contribution of this work as compared to other approaches is the usage of the context tags only, thus, ignoring lexical information. It allows for reducing significantly the decision space.

1 Introducción

1.1 Motivación

La motivación para realizar este proyecto se debe a que dentro del área de procesamiento del lenguaje natural, los modelos conexionistas han sido probados ampliamente sin éxito, comparado con otros métodos; siempre pensando que han sido tratados en varias aplicaciones como agrupamientos semánticos y elecciones de preposiciones.

La etiquetación de clases gramaticales de palabras (*part of speech tagging POS*) es de gran interés en un extenso número de aplicaciones, como en el acceso a base de datos textuales, análisis sintáctico robusto y análisis sintáctico en general. Los lingüistas usan reglas manuales para la etiquetación de categorías gramaticales, muchas veces con ayuda de un corpus.

Probaremos qué resultados son factibles de descubrir al usar una red neuronal entrenada con un corpus etiquetado. El entrenamiento de esta red debe ser completamente automático, la información lingüística será codificada directamente en un conjunto de patrones obtenidos del analizador morfológico AGME (Analizador y Generador de Morfología del Español) el cual implementa un modelo desarrollado en el Laboratorio de Lenguaje Natural del Centro de Investigación en Computación del Instituto Politécnico Nacional, que consiste en la preparación de la hipótesis durante el análisis y su verificación, usando un conjunto de reglas de generación. La ventaja del modelo de análisis a través de la generación, son la simplicidad y la facilidad de implementación.

1.2 Objetivo general

Nuestro objetivo general es encontrar un método no supervisado para la etiquetación de clases gramaticales, que dependa de propiedades distribucionales generales del texto, estas propiedades deben ser invariantes a través de los lenguajes.

Implementamos un sistema de etiquetación, que solamente utiliza información correspondiente a las clases gramaticales de las palabras, esta información es representada en un formato de etiquetas.

Se propone una solución empírica para la etiquetación de categorías gramaticales, únicamente con información pertinente a las clases gramaticales a las que corresponden las palabras en un corpus y se propone continuar, con algunos métodos para mejorar el trabajo.

1.3 Objetivos específicos

En este proyecto consideramos que existe una manera de atacar el problema de etiquetación automática usando distintas redes neuronales sugiriendo restricciones en las clases gramaticales de palabras posibles y desconocidas, obtenidas de su contexto.

El objetivo específico es desarrollar el modelo, el método y el software para un sistema computacional, idóneo para resolver el problema de la etiquetación de clases gramaticales, únicamente asignando una etiqueta que corresponde a la categoría gramatical a la cual pertenece una palabra, esta asignación se debe efectuar de manera correcta. Con la metodología de las redes neuronales y usando el algoritmo de retropropagación, tratamos de encontrar una solución al problema de la resolución automática de la homonimia morfológica para el idioma español.

Para desarrollar el modelo de este sistema, es necesario implementar distintos tipos de redes neuronales, realizar diferentes experimentos en cada una de las redes modificando algunos de sus parámetros. También es necesario analizar y comparar los resultados obtenidos en las redes. Este proceso es imprescindible para desarrollar el método de resolución, necesario para resolver el problema planteado.

No sólo hay que separar la homonimia, sino también tomar en consideración enteramente el contenido comunicativo de la oración.

1.4 Hipótesis

El uso explícito de etiquetas de contextos antecedentes y subsecuentes, representados en redes neuronales, con un amplio uso de características gramaticales léxicas, obtiene una solución de la homonimia morfológica, con probabilidades en los contextos de cada una de las diferentes redes empleadas.

1.5 Estructura de la tesis

Los experimentos que hemos realizado usan diferentes redes neuronales, con la finalidad de asignar etiquetas de clases gramaticales a palabras desconocidas. De la misma manera usamos estas redes, pero con un peso adicional, con lo que obtenemos más información morfológica que depende de los contextos, que se utilizan en cada una de las redes.

La tesis contiene un antecedente sobre el funcionamiento de las redes neuronales, sus propiedades, así como la etiquetación de clases gramaticales en la sección del marco teórico.

La sección de estado del arte contiene un resumen de los trabajos que utilizan distintas metodologías para resolver el problema de la etiquetación de categorías gramaticales. En esta sección se encuentra una introducción a la morfología lingüística, se describen las categorías gramaticales y la morfología del lenguaje español. También en esta sección se explican los distintos tipos de homonimia que existen.

En la sección de método de resolución describimos el modelo de red neuronal de retropropagación y la implementación de los métodos de los objetos que forman la red neuronal. También explicamos con detalle, todos los algoritmos necesarios para la implementación de las redes neuronales utilizadas.

En la sección método experimental, describimos el modelo computacional de la red neuronal, la preparación de los datos de entrada para su manipulación, también se describe el entrenamiento de todas las redes, que depende de los contextos en que se encuentren las etiquetas de palabras, dentro de la frase. En esta sección especificamos la fase experimental y las diversas pruebas realizadas. Esto se realizo para dos problemas uno de ellos fue la etiquetación de clases gramaticales con redes neuronales y el otro fue para la resolución de la homonimia morfológica.

El análisis de los resultados es mostrado en la sección del mismo nombre, aquí analizamos todas las salidas obtenidas por distintas redes y con diferentes parámetros de entrada para cada una de las redes, con la metodología del algoritmo de retropropagación. En esta parte se analizan los resultados de los dos problemas planteados anteriormente. También se describen los resultados obtenidos con otros etiquetadores los cuales son el etiquetador TnT (Trigrams 'n' Tags) y el método de etiquetación de Brill,

En la sección de conclusiones encontramos los aspectos más importantes de este trabajo e interpretamos los resultados obtenidos. En un primer caso, interpretamos los

resultados que encontramos en la etiquetación de las clases gramaticales y en un segundo caso, se interpretaron los resultados obtenidos, al asignarle pesos a las salidas de las redes, para verificar la resolución automática de la homonimia morfológica. Todo esto se realizo para distintos contextos.

En la sección de trabajo futuro, se describen algunos aspectos importantes que estamos tomando en cuenta, para la continuación de este trabajo de investigación. Estos aspectos son probar este sistema para un corpus etiquetado de otro lenguaje y comparar nuestros resultados con una metodología de reconocimiento de patrones.

Tenemos una sección donde se encuentran todas las referencias utilizadas para la realización de este trabajo.

Contamos con tres apéndices. En el apéndice A, se encuentra la implementación de una red de retropropagación, con la propagación hacia delante y hacia tras del error, que son las funciones básicas necesarias de las redes de retropropagación. En el apéndice B describimos la formalización matemática de la red neuronal de retropropagación. El apéndice C, contiene todas las clases gramaticales del idioma español, así como sus propiedades y características más importantes.

2 Marco Teórico: Redes neuronales

1.1 Historia de las redes neuronales

El fundamento de las redes neuronales en estudios neurobiológicos se remontan a hace más de un siglo. Por muchas décadas, los biólogos han especulado sobre el trabajo del sistema nervioso. En el siglo pasado una declaración de William James (1890) es particularmente intuitiva y se refleja en los trabajos de muchos investigadores [1].

La cantidad de actividad de cualquier punto en la corteza cerebral es la suma de las tendencias de todos los puntos que descargan en él, tales tendencias son proporcionales a

1. él número de veces que la excitación de otros puntos pudieran tener, acompañados estos del punto en cuestión;
2. la intensidad de tales excitaciones y
3. la ausencia de funcionalidad de algún punto rival desconectado con el primer punto, dentro del cual las descargas pudieran ser desviadas.

Otro frente de ataque proviene de psicólogos procurando entender exactamente cómo aprenden, olvidan, reconocen y otras de tales tareas dotadas a los animales.

McCulloch y Pitts (1943) son acreditados como los desarrolladores del primer modelo matemático de una simple neurona. La regla de aprendizaje incremental de Hebb (1949), modifica los pesos de las conexiones examinando si dos nodos conectados están simultáneamente encendidos o apagados. El modelo neuronal del *perceptrón,* de Rosenblatt (1958) y la regla de aprendizaje, está basada en descenso de gradiente, *compensación* o *castigo*. La simplicidad de este esquema fue también su justo castigo, lo cual es demostrado por Minsky y Papert (1969) ya que existen ciertas tareas simples de reconocimiento de patrones, que un simple perceptrón no puede realizar. Un problema similar pero en la regla de aprendizaje fue encarado por Widrof-Hoff (1960,1962).

2.1 Neuronas Biológicas

Una neurona es una célula viva y como tal, contiene los mismos elementos que forman parte de todas las células biológicas. En general, una neurona consta de un cuerpo

celular más o menos esférico, de 5 a 10 micras de diámetro, del que salen una rama principal, el *axón,* y varias ramas más cortas, llamadas *dendritas*. A su vez, el axón puede producir ramas en torno a su punto de arranque y con frecuencia se ramifica extensamente cerca de su extremo.

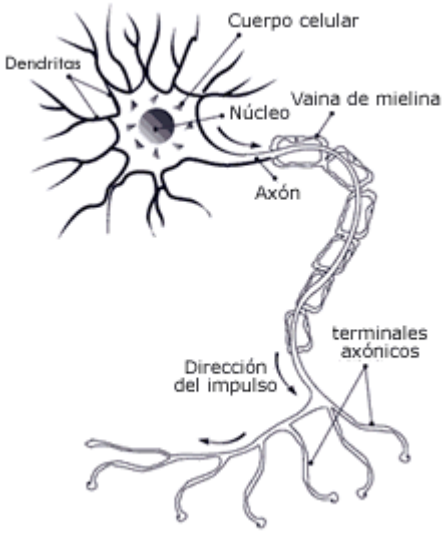

Figura 2-1 Neurona Biológica

Una de las características que diferencian a las neuronas del resto de las células vivas, es su capacidad de comunicarse. En términos generales, las dendritas y el cuerpo celular reciben señales de entrada; el cuerpo celular las combina, integra y emite señales de salida.

Cuando se alcanza cierto grado de excitación en las dendritas de una neurona ésta provoca un impulso electroquímico en su axón. Cuando el impulso llega al final del axón, se produce la segregación de sustancias químicas que se encuentran almacenadas en los terminales de los axones. Estos neurotransmisores reaccionan con los receptores que se encuentran en la célula o la dendrita, a la que está conectado el axón. El axón transporta esas señales a las terminales axónicas, que se encargan de distribuir información a un nuevo conjunto de neuronas. Por lo general, una neurona recibe información de miles de otras neuronas y a su vez envía información a miles de neuronas. La información viaja entre las neuronas en forma de impulsos electroquímicos, estos impulsos están formados por iones de sodio y potasio. Algunos de los neurotransmisores que hay en el cerebro son: la serotónina, la dopámina y la acetilcólina.

2.2 Modelos de redes neuronales

Cada neurona envía impulsos a un gran número de neuronas (divergencia) y recibe impulsos procedentes de muchas neuronas (convergencia). Esta sencilla idea parece ser el fundamento de toda la actividad del sistema nervioso central.

Dado que las conexiones sinápticas pueden ser tanto excitatorias como inhibitorias, estos circuitos hacen posibles que los sistemas de control puedan tener tanto retroalimentación positiva como negativa. Por supuesto, estos circuitos tan sencillos no describen adecuadamente la gran complejidad de la neuroanatomía.

En el trabajo de McCulloch y Pitts, por primera vez se trata al cerebro como un organismo computacional. La teoría de McCulloch-Pitts se basa en cinco suposiciones:

- La actividad de una neurona es un proceso todo-nada.
- Es preciso que un número fijo de sinapsis (>1) sean excitadas dentro de un período de adición latente para que se excite una neurona.
- El único retraso significativo dentro del sistema nervioso es el retardo sináptico.
- La actividad de una sinapsis absolutamente inhibitoria evita la excitación de la neurona en ese momento.
- La estructura de la red de interconexión no cambia con el tiempo.

Las redes neuronales siempre modelan algún sistema neurobiológico, asumiendo simplificaciones sobre redes neuronales biológicas. Tales simplificaciones son necesarias para entender las propiedades previstas e intentar cualquier análisis matemático. Incluso si todas las propiedades de las neuronas son conocidas, la simplificación puede ser necesaria para su tratamiento.

Generalmente se pueden encontrar tres tipos de neuronas, aquellas que reciben estímulos externos, relacionadas con el aparato sensorial, que tomarán la información de entrada. Dicha información se transmite a ciertos elementos internos que se ocupan de su procesado, llamados unidades ocultas. Una vez finalizado el período de procesado, la información llega a las unidades de salida, cuya misión es dar la respuesta del sistema.

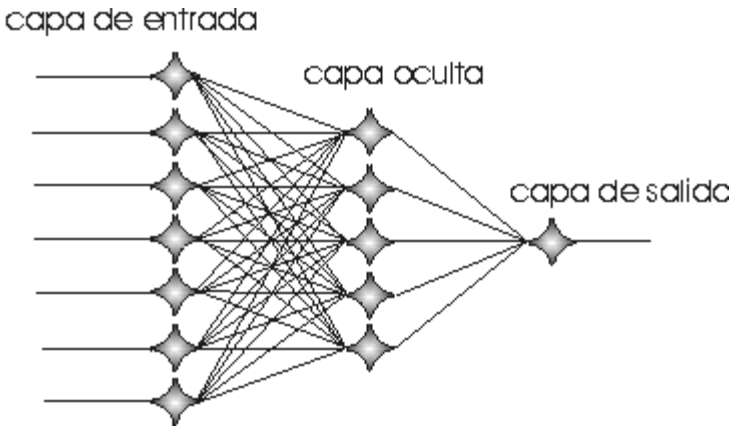

Figura 2-2 Estructura de una red neuronal

Considerando la *i*-ésima neurona, ésta se caracteriza en cualquier instante por un valor numérico denominado valor o estado de activación $a_i(t)$. Asociado a cada unidad, existe una función de salida f_i, que transforma el estado actual de activación en una salida, y_i. Dicha señal, es enviada a través de los canales unidireccionales a otras unidades de la red; en estos canales la señal se modifica de acuerdo a la sinapsis (peso w_{ji}) asociada a cada uno de ellos según una determinada regla. Las señales que llegan a la unidad *j*-ésima se combinan entre ellas generando así la *entrada total*, Net_j.

$$Net_j = \sum_i y_i * w_{ji}$$

Una *función de activación*, *F*, determina el nuevo estado de activación $a_j(t+1)$ de la neurona, teniendo en cuenta la entrada total calculada y el anterior estado de activación $a_j(t)$. La dinámica que rige la actualización de los estados de las unidades puede ser de dos tipos: modo asíncrono y modo síncrono.

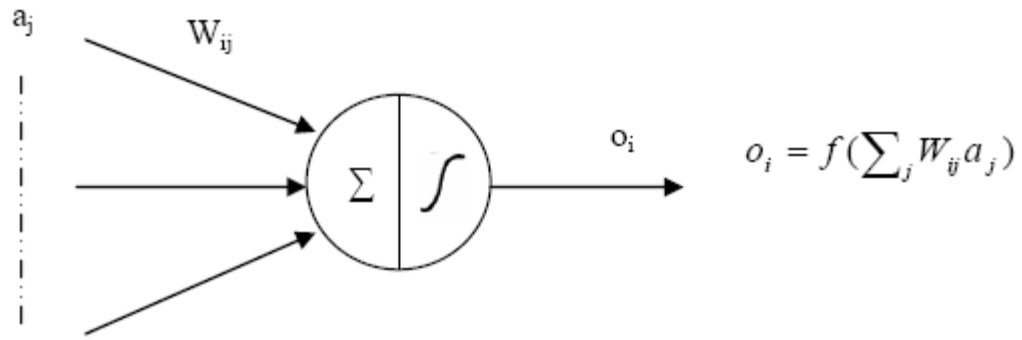

Figura 2-3 Entradas y salidas de una neurona

Se reconoce como capa o nivel a un conjunto de neuronas cuyas entradas provienen de la misma fuente y cuyas salidas se dirigen al mismo destino.

La activación de una unidad U_i en el tiempo t se designa por $a_i(t)$; es decir:

$$A(t) = (a_1(t), a_2(t), \ldots a_i(t), \ldots\ldots, a_N(t))$$

Los valores de activación pueden ser continuos o discretos.

Entre las neuronas que forman una red neuronal artificial existe un conjunto de conexiones que unen unas con otras. Asociada con cada unidad U_i hay una función de salida $f_i(a_i(t))$, que transforma el estado actual de activación $a_i(t)$ en una señal de salida $y_i(t)$; es decir:

$$y_i(t) = f_i(a_i(t))$$

El vector que contiene las salidas de todas las neuronas en un instante t es:

$$Y(t) = (f_1(a_1(t)), f_2(a_2(t)), \ldots f_i(a_i(t)), \ldots\ldots, f_N(a_N(t)))$$

Dado el estado de activación $a_i(t)$ de la unidad U_i y la entrada total que llega a ella, Net_i, el estado de activación siguiente $a_i(t+1)$, se obtiene aplicando una *función de activación F*.

$$a_i(t+1) = F_i(a_i(t), Net_i)$$

El estado de activación anterior no se tiene en cuenta. Según esto, la salida de una neurona *i* (y_i) quedará según la siguiente expresión:

$$y_i(t+1) = f(Net_i) = f\left(\sum_{j=1}^{N} w_{ij} y_j(t)\right)$$

Normalmente la función de activación no está centrada en el origen del eje que representa el valor de la entrada, sino que existe cierto desplazamiento debido a las características internas de la propia neurona y que no es igual a todas ellas. Este valor se denota como θi y representa el umbral de activación de la neurona *i*.

$$y_i(t+1) = f(Net_i - \theta i) = f\left(\sum_{j=1}^{N} w_{ij} y_j(t) - \theta i\right)$$

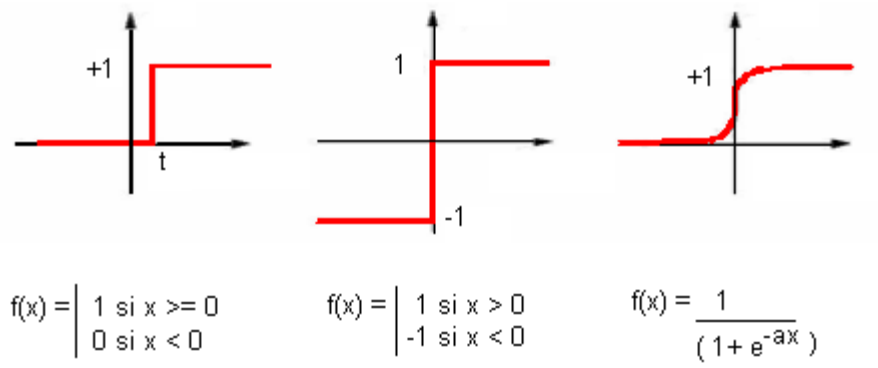

Figura 2-4 Algunas funciones de transferencia

2.2.1 Regla de aprendizaje

La modificación del comportamiento inducido por la interacción con el entorno y como resultado de experiencias, conduce al establecimiento de nuevos modelos de respuesta a estímulos externos.

En el caso de las redes neuronales artificiales, se puede considerar que el conocimiento se encuentra representado en los pesos de las conexiones entre neuronas. Todo proceso de aprendizaje implica cierto número de cambios en estas conexiones. Se aprende modificando los valores de los pesos de la red.

2.2.2 Formas de conexión entre neuronas

La conectividad entre los nodos de una red neuronal está relacionada con la forma en que las salidas de las neuronas están canalizadas para convertirse en entradas de otras neuronas. La señal de salida de un nodo puede ser una entrada de otro elemento de proceso o incluso ser una entrada de sí mismo (conexión auto recurrente). Cuando ninguna salida de las neuronas es entrada de las neuronas del mismo nivel o de niveles precedentes, la red se describe como de propagación hacia delante. Cuando las salidas pueden ser conectadas como entradas de neuronas de niveles previos o del mismo nivel, incluyéndose ellas mismas, la red es de propagación hacia atrás.

2.3 Etiquetación de clases gramaticales

Un sistema de etiquetación de clases gramaticales es un sistema el cual automáticamente asigna la clase gramatical a la que corresponde una determinada palabra, esto lo hace usando información contextual.

El propósito general de un etiquetador de clases gramaticales es asociar a cada palabra en un texto con su categoría morfosintáctica, representada como una etiqueta.

El proceso de etiquetación consiste de tres pasos:

- Tokenización: romper el texto en símbolos (tokens)
- Búsqueda léxica: proporcionar todas las etiquetas potenciales a cada símbolo
- Desambiguación: asignar a cada símbolo una sola etiqueta

La etiquetación automática de un texto es un primer paso importante en el descubrimiento de estructuras lingüísticas en un corpus grande. La información de las clases gramaticales, facilita un análisis de alto nivel, tal como el reconocimiento de frases sustantivas y otros patrones en el texto.

Para que una función de etiquetación, sea un componente práctico en un sistema de procesamiento de lenguaje natural, creemos que el etiquetador debe ser:

- Robusto: El corpus del texto contiene construcciones que no son gramaticales, frases aisladas y datos que no son lingüísticos. El corpus debe contener palabras que sean desconocidas al etiquetador.
- Eficiente: Si un etiquetador es usado para analizar grandes corpus, este debe ser eficiente, funcionando en tiempo lineal en el número de palabras etiquetadas. Cualquier entrenamiento requerido debe ser rápido, habilitando un tiempo de respuesta con un nuevo corpus.
- Exacto: Un etiquetador debe intentar asignar la etiqueta de clases gramaticales correcta a cada palabra encontrada.
- Sintonizado: Un etiquetador debe ser capaz de tomar ventaja de la perspicacia lingüística. Debe ser capaz de corregir errores sistemáticos al suplir consejos apropiados *a priori*. Debe ser posible dar diferentes recomendaciones, para diferentes corpus.

- Reutilizable: El esfuerzo requerido para crear nuevos objetivos de etiquetación en un nuevo corpus, en un nuevo conjunto de etiquetas y en nuevos lenguajes debe ser mínimo.

Aplicaciones potenciales de etiquetadores de clases gramaticales, existen en muchas áreas incluyendo reconocimiento del habla, síntesis del habla, traducción, desambiguación y recuperación de información.

En este trabajo se aplicará el paradigma de las redes neuronales al problema de etiquetación de las clases gramaticales en un texto.

3 Estado del arte

Las investigaciones en etiquetación de clases gramaticales (*POS tagging,* por sus siglas en inglés) están muy relacionadas al corpus lingüístico.

Por algún tiempo *POS tagging* fue considerado una parte inseparable del procesamiento de lenguaje natural, debido a que existen ciertos casos donde la clase gramatical correcta, no puede ser decidida sin entender la semántica o pragmática del contexto. Esto es extremadamente costoso, especialmente porque analizar los niveles altos es mucho más difícil cuando múltiples clases gramaticales deben ser consideradas por cada palabra.

A mediados de 1980, investigadores en Europa empezaron a usar modelos de Markov ocultos (*HMM*), para desambiguar *POS tagging*, cuando trabajaban para etiquetar el Lancaster-Oslo-Bergen Corpus of British English. *HMM* involucra casos de conteo y hacer una tabla de probabilidades de ciertas secuencias. Un *HMM* avanzado, aprende las diversas posibilidades, no sólo en pares sino en tripletas o secuencias largas. Cuando varias palabras ambiguas ocurren juntas, estas posibilidades se multiplican. Sin embargo esto es fácil para enumerar cualquier combinación y asignar una probabilidad relativa a cada una, al multiplicar las probabilidades unidas de cada elección en turno.

En 1987, Steve DeRose y Ken Church independientemente desarrollaron algoritmos de programación dinámicos para resolver el mismo problema en un menor tiempo. Este método es similar al algoritmo Viterbi, conocido en algún tiempo en otros campos. De Rose uso una tabla de pares, mientras Church uso una tabla de tripletas y un método ingenioso para estimar los valores raros o no existentes en el corpus Brown. Ambos métodos consiguen una precisión del 95%. La disertación DeRose en 1990 en la universidad de Brown, incluye análisis de tipos de error específicos, probabilidades y otros datos relacionados, repitiendo este trabajo para el griego, donde probó una similitud efectiva.

Estas búsquedas fueron sorpresivamente negativas al campo del procesamiento de lenguaje natural, la efectividad reportada fue más alta que la efectividad típica de algoritmos muy sofisticados que integraban elecciones de *POS*, con muchos niveles de análisis lingüísticos. Los métodos DeRose y Church fallaron para algunos de los casos conocidos, donde la semántica es requerida pero fueron insignificantes. Esto convenció a

muchos en el campo de *POS tagging* pudiendo provechosamente estar separado de otros niveles de procesamiento; consecuentemente se simplificó la teoría y práctica del análisis computarizado del lenguaje y esto fomentó a los investigadores para encontrar formas de separar otras piezas fácilmente.

Actualmente es válido que los etiquetadores funcionen con un alto grado de precisión. Sin embargo, esto se presenta debido a que el suceso en ejecución puede ser abrumador, atribuido a una tarea general de lexicalización. Es más Charniak, Hendrickson, Jacobson y Perkowits (1993), notaron que una simple estrategia de elegir la etiqueta más parecida para cada palabra en un texto, produce un 90% de efectividad.

Aunque un modelo *uni*-grama alcanza una efectividad total del 90%, esta depende en gran parte de la información léxica, a esto sigue la ineficacia en textos no estándares que contienen muchas terminologías de dominio especifico.

Veronis e Ide (1990) usan un enfoque con redes neuronales extrayendo información léxica de un diccionario. Estos resultados identifican el sentido correcto de una palabra en solo 71.74% de todos los casos.

Nakamura et al (1990), utilizaron la predicción de la categoría de palabra usando una arquitectura de red neuronal llamada *NETgram*, que es una arquitectura basada en retropropagación. La categoría gramatical de la palabra precedente fue usada para predecir la categoría de la siguiente palabra. Ellos reportaron un índice de reconocimiento de palabras del 68%.

La regla basada en aproximaciones de Voutilainen, (1992), ha conseguido un índice de clasificación del 99.7%, el fracaso de su método se debe, al excesivo tiempo de consumo para desarrollar reglas y el sistema producido es altamente dependiente del lenguaje. También es necesario un lexicón muy grande, el lexicón usado cubre cerca del 95% de todos los lexemas que aparecen en el texto.

Martin Eineborg y Björn Gambäck (1994), describen una serie de experimentos con tres arquitecturas distintas de redes de retropropagación, usadas para la tarea del reconocimiento de palabras desconocidas en un sistema de procesamiento de lenguaje natural. En la primera tarea efectuada, las redes clasificaron palabras de un conjunto de clases gramaticales (sustantivo, adjetivo o verbo) únicamente, esta tarea la realizo con una tasa de efectividad de 93.6%. La segunda tarea involucró un conjunto más grande que constaba de 13 categorías posibles de palabras, obteniendo una tasa de 96.4%.

La regla más conocida basada en etiquetación Brill (1994), trabaja en dos etapas: esta asigna la etiqueta más parecida a cada palabra en el texto, entonces aplica las reglas de transformación del tipo "Remplaza la etiqueta *X*, por la etiqueta *Y*, al aparecer el entorno *Z*". Al aparecer el entorno, este abarca tres tokens secuenciales, en cada dirección y se refiere a palabras, etiquetas o propiedades de palabras dentro de la región. Este etiquetador consigue una efectividad promedio igual a 96.5% en el corpus Wall Street Journal (*WSJ*). Sin embargo su ejecución depende de un vocabulario extenso.

La etiquetación estocástica es una aplicación clásica de los modelos de Markov. Brants (2000), argumenta que los modelos de Markov de segundo orden pueden también conseguir una alta eficacia, siempre que sea complementada por técnicas de suavizado y por mecanismos para manejar palabras desconocidas.

La etiquetación con etiquetas *n-Trigramas* (*TnT*) emplea palabras desconocidas, por la estimación de la probabilidad de la etiqueta, por el sufijo de la palabra desconocida y su capitalización. Tiene una efectividad de 96.7% reportado por el etiquetador TnT, en el corpus WSJ, parece ser un resultado de sobre-entrenamiento. Esto es el máximo funcionamiento obtenido al entrenar TnT, con solo el 2.9% de palabras desconocidas, en el corpus de prueba.

La etiquetación de clases gramaticales depende de una variedad de características subléxicas, también para la probabilidad de las secuencias *etiqueta/etiqueta* o *etiqueta/palabra*. En general todos los etiquetadores que existen han incorporado tal información, hasta cierto punto. Los modelos de campos aleatorios condicionales Conditional Random Fields, (CRF), Lafferty, McCallum & Pereira (2002), superan al etiquetador de modelos de Markov en palabras desconocidas, por la excesiva dependencia en características ortográficas y morfológicas. Los autores perciben que los etiquetadores basados en el modelo CRF, son potencialmente flexibles, debido a que ellos pueden ser combinados con características de algoritmos de inducción. Sin embargo, el entrenamiento es complejo y de bajo costo computacional. Esto es confuso, en la contribución relativa de características en este modelo.

El etiquetador de máxima entropía *MaxEnt*, propuesto por Ratnaparkhi (1996), informa de la unión de distribución de etiquetadores de clases gramaticales y características de una oración con un modelo exponencial. Las características observadas frecuentemente en un corpus son seleccionadas de un fondo común. Los parámetros del modelo son

estimados usando un procedimiento intensivo computacional de escalamiento iterativo generalizado (Generalizad Iterative Scaling GIS) de entrenamiento, para maximizar la probabilidad condicional del conjunto de entrenamiento, dado por el modelo. El etiquetador MaxEnt tiene una efectividad de 96.6%.

Las categorías en que puede dividirse *POS tagging* están: basadas en reglas, estocástica y enfoque neuronal. Algunos corrientes importantes para *POS tagging* incluyen los algoritmos Viterbi, Brill, Baulm-Welch. Los etiquetadores con modelos ocultos de Markov HMM y modelos no ocultos de Markov pueden ser implementados usando el algoritmo Viterbi

En este trabajo ocupamos el enfoque neuronal para clasificar los distintos tipos de clases gramaticales, de un corpus.

3.1 Categorías gramaticales y morfología

Podemos definir una palabra como un elemento lingüístico, formado por uno o varios morfemas, dotado de acento, capaz de formar por sí solo un enunciado, perfectamente delimitable desde el punto de vista fónico y cuya transcripción gráfica en español no admite la menor indeterminación: constituye una unidad limitada por espacios vacíos [15].

Los lingüistas agrupan las palabras de un lenguaje dentro de clases, las cuales muestran un comportamiento sintáctico similar y un tipo semántico. Estas clases de palabras son llamadas categorías sintácticas o gramaticales, (*part of speech POS*). Tres categorías gramaticales importante son: sustantivo, verbo y adjetivo.

Las clases de palabras son normalmente divididas dentro de dos categorías. Las abiertas o categorías léxicas que son los sustantivos, verbos y adjetivos. A los cuales nuevas palabras son aumentadas. Las cerradas o categorías funcionales son las preposiciones y determinantes, las cuales tienen un claro uso gramatical. Las categorías de palabras están sistemáticamente relacionadas con procesos morfológicos.

La morfología estudia la estructura de las palabras y su relación con las categorías gramáticas del lenguaje. El objetivo del análisis morfológico automático es llevar a cabo una clasificación morfológica de una forma de palabra. Por ejemplo, el análisis de la forma

gatos resulta en *gato+Noun+Masc+Pl*, que nos indica que se trata de un sustantivo plural con género masculino y su forma normalizada (lema) es gato.

Los principales tipos de procesos morfológicos son inflexión, derivación y combinación. La inflexión son modificaciones sistemáticas de la raíz, la derivación resulta de un cambio radical de la categoría sintáctica y a menudo un cambio en significado, la combinación se refiere a la combinación de dos o mas palabras en una nueva palabra.

Los lenguajes según sus características morfológicas básicamente y dependiendo de la tendencia en la manera de la combinación de morfemas se clasifican en aglutinativos y flexivos.

Se dice que un lenguaje es aglutinativo si:

• Cada morfema expresa un sólo valor de una categoría gramatical.

• No existen alternaciones de raíces o las alternaciones cumplen con las reglas morfonológicas que no dependen de la raíz específica, como, por ejemplo, armonía de vocales, etc.

• Los morfemas se concatenan sin alteraciones.

• La raíz existe como palabra, sin concatenarse con morfemas adicionales algunos.

Unos ejemplos de los lenguajes aglutinativos son las lenguas turcas (el turco, kazakh, kirguiz, etc.) o el húngaro.

Por otro lado, un lenguaje es flexivo si:

• Cada morfema puede expresar varios valores de las categorías gramaticales. Por ejemplo, el morfema -mos en español expresa cumulativamente los valores de las categorías persona (tercera) y número (plural).

• Alternaciones de raíces no son previsibles, sin saber las propiedades de la raíz específica no se puede decir qué tipo de alternación se presentará.

• Los morfemas pueden concatenarse con ciertos procesos morfonológicos no estándares en la juntura de morfemas.

• La raíz no existe como palabra sin morfemas adicionales (por ejemplo, escrib- no existe como palabra sin -ir, -iste, -ía, etc.).

Unos ejemplos de los lenguajes flexivos son lenguas eslavas (ruso, checo, ucraniano, etc.) o románicas (latín, portugués, español, etc.). Normalmente, esta clasificación de lenguajes refleja las tendencias; es decir, muy raras veces un lenguaje es absolutamente aglutinativo o flexivo. Por ejemplo, el finlandés es un lenguaje básicamente

aglutinativo, aunque con algunos rasgos de un lenguaje flexivo por ejemplo, varios valores de las categorías gramaticales pueden unirse en el mismo morfema.

3.2 Homonimia

Las palabras con la misma representación textual pero diferente significado son llamadas palabras homónimas con respecto a otras, este fenómeno es llamado homonimia. Grandes fragmentos de textos pueden ser homónimos.

Para explicar el fenómeno de la homonimia con más detalle, debemos recurrir a los términos estrictos de *lexema* y *forma de palabra* en lugar del término *palabra*. Podemos distinguir los siguientes casos importantes de homonimia:

- *Homonimia léxico-morfológica:* dos formas de palabra, pertenecen a dos diferentes lexemas. Este es el caso más general de homonimia. Por ejemplo la cadena de texto *aviso* es la forma de palabra del verbo *avisar* y del sustantivo *a*viso. La forma de palabra *clasificación* pertenece a dos lexemas *clasificación*$_1$ 'proceso de clasificación' y *clasificación*$_2$ 'resultado de clasificación', aunque la forma de palabra clasificaciones pertenece solamente a *clasificación*$_2$, ya que *clasificación*$_1$ no tiene forma plural. Se debe notar que no tiene ninguna relevancia si el nombre del lexema coincide con la forma de palabra homónima específica o no. Otro caso de homonimia léxico-morfológica es representado por dos diferentes lexemas cuyos conjuntos de formas de palabras se intersecan en más de una forma de palabra. Por ejemplo, los lexemas *rodar* y *rueda* cubren dos formas de palabras homónimas, *rueda* y *ruedas*.
- *Homonimia léxica simple:* dos o más lexemas tienen los mismos conjuntos de formas de palabras, como *real*$_1$ 'verdadero' y *real*$_2$ 'realeza' (ambas tienen el mismo conjunto de formas de palabra {*real, reales*}).
- *Homonimia morfo-sintáctica:* los conjuntos completos de formas de palabras, son los mismos para dos o más lexemas, pero estos lexemas difieren en significado y en una o más propiedades morfosintácticas. Por ejemplo (el) *frente* y (la) *frente*, difieren en la adición de significado en género, lo cual influye en las propiedades sintácticas de los lexemas.

- *Homonimia morfológica simple:* dos o más formas de palabras, son diferentes miembros del conjunto de formas de palabras para el mismo lexema. Por ejemplo *fáciles* es la forma de palabras para masculino plural y *fácil* es la forma de palabra para femenino plural del adjetivo. Debemos admitir este tipo de homonimia, debido a que las formas de palabras de los adjetivos en español generalmente difieren en género.

La resolución de todos estos tipos de homonimia es realizada por el lector o escritor según el contexto de la forma de palabra o basada en situaciones extra lingüísticas en las cuales esta forma es usada. Las operaciones mentales son inmediatas y bastante efectivas. Sin embargo, la resolución de tales ambigüedades por computadora requiere métodos sofisticados.

La resolución de homonimia (ambigüedad en general) es uno de los problemas más difíciles de lingüística computacional y debe ser tratada como una parte esencial e integral del proceso del entendimiento del lenguaje.

Sin resolución automática de homonimia, todos los intentos para entender el lenguaje natural serán altamente propensos a errores y tendrán limitada utilidad.

Nuestra tarea es crear reglas formales para la categorización de palabras dadas, a diferencia de su alternativa morfológica y situación sintáctica.

La implicación de la asignación de tales lexemas homónimos a ciertas clases de palabras, no significa un problema simple de selección de restricción a nivel superficial. Cada partícula modal ha preservado mucho de su étimo sintáctico y sus propiedades semánticas.

En el campo de la lingüística computacional, los lexemas homónimos usualmente forman entradas separadas en los diccionarios. Los analizadores lingüísticos deben resolver la homonimia automáticamente, escogiendo la opción correcta a través de aquellas descritas en el diccionario.

4 Método de resolución

4.1 Funcionamiento de una red de retropropagación

El funcionamiento de una red de retropropagación (*RRP*) consiste en el aprendizaje de un conjunto definido de pares entrada-salida, dados como ejemplo, empleando un ciclo de propagación-adaptación de dos fases: primero se aplica un patrón de entrada como estímulo para la primera capa de las neuronas de la red, este se va propagando a través de todas las capas superiores hasta generar una salida, y se compara el resultado obtenido en las neuronas de salida con la salida que se desea obtener, para calcular el valor del error de cada neurona de salida. A continuación, estos errores se transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de las capa intermedia que contribuyan directamente a la salida, recibiendo el porcentaje del error aproximado a la participación de la neurona intermedia en la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total. Basándose en el valor del error recibido, se reajustan los pesos de conexión de cada neurona, de manera que en la siguiente vez que se presente el mismo patrón, la salida este más cercana a la deseada; disminuyendo el error.

La importancia de la red de retropropagación consiste en su capacidad de adaptar automáticamente los pesos de las neuronas de las capas intermedias para aprender la relación que existe entre un conjunto de patrones dados como ejemplo y sus salidas correspondientes.

La red debe encontrar una representación interna que le permita generar las salidas deseadas cuando se le dan las entradas de entrenamiento y que se pueda aplicar, además, a entradas no presentadas durante la etapa de aprendizaje, para clasificarlas según las características que compartan con los ejemplos de entrenamiento.

Para el entrenamiento de una red es necesario tener un conjunto de patrones y las salidas deseadas correspondientes a ese patrón, más una función de error que mide las diferencias entre la salida de las redes y los valores deseados. Los pasos básicos son los siguientes:

1. Presentar un patrón de entrenamiento y propagar éste a través de la red para obtener las salidas.
2. Comparar las salidas con los valores deseados y calcular el error.
3. Calcular las derivadas $\partial E / \partial w_{ij}$ del error con respecto a los pesos.
4. Ajustar los pesos para minimizar el error.
5. Repetir hasta que el error sea aceptablemente pequeño o el tiempo se haya agotado.

La función de suma de cuadrados es una buena elección para la función de error.

$$E_{SSE} = \sum_p \sum_i (d_{pi} - y_{pi})^2$$

Donde *p* indexa los patrones en el conjunto de entrenamiento, *i* indexa los nodos de salida, d_{pi} e y_{pi} son respectivamente el objetivo y la salida actual de la red para el *i*-ésimo nodo de salida del patrón *p*. El error cuadrado promedio

$$E_{MSE} = \frac{1}{PN} E_{SSE}$$

normaliza E_{SSE} para el número de patrones de entrenamiento *P* y salidas de red *N*. Algunas ventajas de las funciones *SSE* y *MSE* incluyen una variedad diferencial y el hecho de que los costos dependen sólo de la magnitud del error. En particular, una desviación de una magnitud dada tiene el mismo costo independiente de errores en otras salidas.

4.2 Propagación hacia delante

En la propagación hacia delante, la red calcula una salida basada en las entradas actuales. Cada nodo *i* calcula una suma de pesos a_i de sus entradas y pasa a través de una no linealidad para obtener el nodo de salida y_i

$$a_i = \sum_{j<i} w_{ij} y_j$$

$$y_i = f(a_i)$$

Normalmente *f* es una función monotónica limitada. El índice *j* en la suma, corre sobre todos los índices $j < i$ de nodos que podrían enviar salidas al nodo *i*. Si no existe conexión del nodo *j*, el peso w_{ij} es *0*.

Cada nodo es evaluado en orden, empezando con el primer nodo oculto y continuando hasta el último nodo de salida. En redes etiquetadas, la primera capa oculta es actualizada con base en entradas externas, la segunda capa oculta es actualizada con base en las salidas de la primera capa oculta y así hasta que la capa de salida sea actualizada con base en las salidas de la última capa.

4.2.1 Cálculo del error

A menos que la red esté perfectamente entrenada, las salidas de la red podrían diferir un tanto de las salidas deseadas. La importancia de estas medidas es calculada por una función de error *E*. Usaremos la función de error E_{SSE}

$$E = \frac{1}{2} \sum_p \sum_i (d_{pi} - y_{pi})^2$$

Una de las razones de conveniencia de E_{SSE} , es que el error en patrones diferentes y diferentes salidas, son independientes. El error total es sólo la suma de errores cuadrados individuales

$$E = \sum_p E_p$$

$$E_p = \frac{1}{2} \sum_i (d_{pi} - y_{pi})^2$$

4.2.2 Cálculo de la derivada

Habiendo obtenido las salidas y calculado el error, el siguiente paso es calcular la derivada del error con respecto a los pesos. Notemos que $E_{SSE} = \sum_p E_p$ es sólo la suma de errores de patrones individuales y la derivada total es sólo la suma derivadas por patrón

$$\frac{\partial E}{\partial w_{ij}} = \sum_p \frac{\partial E_p}{\partial w_{ij}}$$

Lo que hace la retropropagación eficiente, es porque la operación se descompone y también el orden de los pasos. La derivación puede ser escrita

$$\frac{\partial E_p}{\partial w_{ij}} = \sum_k \frac{\partial E_p}{\partial a_k} \frac{\partial a_k}{\partial w_{ij}}$$

Donde el índice *k* corre sobre todos los nodos de salida y a_j es la suma de entrada de pesos para el nodo *j* obtenido en la ecuación $E_{SSE} = \sum_p \sum_i (d_{pi} - y_{pi})^2$. Es conveniente para el primer cálculo un valor δ_i para cada nodo *i*

$$\delta_i = \frac{\partial E_p}{\partial a_i} = \frac{\partial E_p}{\partial y_i} \frac{\partial y_i}{\partial a_i}$$

La cual mide la contribución de a_i al error en el patrón actual. Por simplicidad, los patrones indexados por *p*,son omitidos en y_i , a_i y otras variables.

Para nodos de salida, $\partial E_p / \partial a_k$ es obtenido directamente

$$\partial_k = -(d_{pk} - y_{pk}) f'_k$$

(para nodos de salida)

El primer término es obtenido de la ecuación

$$\frac{\partial E_p}{\partial y_k} = -(d_{pk} - y_{pk})$$

El segundo término

$$\frac{\partial y_k}{\partial a_k} = f'(a_k)$$

es sólo la pendiente $f'_k = f'(a_k)$ de la no linealidad del nodo así como su valor de activación actual. La técnica de retropropagación requiere el uso de neuronas cuya función de activación sea continua y por lo tanto diferenciable, generalmente la función utilizada es de tipo sigmoidal: $y = \frac{1}{1+e^{-x}}$ esta función sigmoidal es conveniente para usar porque f' es una función simple del nodo de salida: $f'(a_k) = y(1-y)$, donde $y = f(a_k)$.

Para nodos ocultos, δ_i es obtenido indirectamente. Los nodos ocultos pueden influir al error sólo a través de sus efectos en nodos *k* para los cuales envían a sus conexiones de salida

$$\delta_i = \frac{\partial E_p}{\partial a_i} = \sum_k \frac{\partial E_p}{\partial a_k} \frac{\partial a_k}{\partial a_i}$$

Pero el primer factor es sólo δ_k del nodo k

$$\delta_i = \sum_k \delta_k \frac{\delta a_k}{\delta a_i}$$

El segundo factor es obtenido al notar si el nodo *i* conecta directamente al nodo *k,* entonces $\delta a_k / \delta a_i = f' w_{ki}$, de otra forma es cero. Y terminamos con

$$\delta_i = f' \sum_k w_{ki} \delta_k$$

(para nodos ocultos)

En otras palabras, δ_i es la suma de pesos de los valores δ_k , del nodo *k* el cual tiene conexión w_{ki}.

Como δ_k debe ser calculado antes de δ_i , $i < k$, el proceso empieza de los nodos de salida y trabaja hacia atrás a través de las entradas, de aquí el nombre de retropropagación. Primero los valores δ son calculados para nodos de salida, entonces los valores son calculados para nodos, eliminados dos pasos de las salidas y así sucesivamente.

Para resumir,

$$\begin{array}{l} -(d_{pk} - y_{pk}) f' i \\ f' i \sum_k w_{ki} \delta_k \\ \delta_i = ¿[¿]¿\{\} \end{array} \quad \text{Para nodos de salida y nodos ocultos}$$

Para nodos de salida, δ_i depende sólo del error $d_i - y_i$ y la pendiente local f'_i de la función de activación del nodo.

Para nodos ocultos, δ_i es una suma de pesos de δ de todos los nodos conectados a tiempo de su propia pendiente f'_i . Porque la forma en que los nodos están indexados, todos los valores δ pueden ser actualizados en un simple barrido a través de los nodos en orden inverso. Normalmente no es necesario calcular los valores δ para la capa de entrada, ya que el proceso usualmente se detiene en la primera capa oculta. Obtenido los δ de los nodos, es un fácil encontrar las derivadas parciales $\delta E_p / \delta_w$ con respecto a los pesos. El segundo término de δ_i es $\delta a_k / \delta w_{ij}$. Porque a_k es una simple suma lineal, esto es cero si $k \neq i$; de lo contrario:

$$\frac{\delta a_i}{\delta w_{ij}} = y_j$$

Donde y_j es la salida del nodo de activación *j*. Finalmente, de $\frac{\partial E_p}{\partial w_{ij}} = \sum_k \frac{\partial E_p}{\partial a_k} \frac{\partial a_k}{\partial w_{ij}}$, el patrón de la derivada del error E_p con respecto a los pesos w_{ij} es

$$\frac{\delta E_p}{\delta w_{ij}} = \delta_i y_j$$

el producto del valor δ del nodo *i* y la salida del nodo *j*.

4.2.3 El algoritmo de actualización de pesos

El siguiente paso después de haber obtenido las derivadas es la actualización de los pesos para disminuir el error. Como hemos visto el término retropropagación se refiere a un método eficiente de calcular las derivadas $\delta E / \delta w$ y un algoritmo de optimización que usa estas derivadas para ajustar los pesos para reducir el error.

El método de optimización de retropropagación es básicamente equivalente al descenso de gradiente. Por definición, la gradiente de *E* apunta en dirección del incremento *E* mas rápido. Con el fin de minimizar *E* los pesos son ajustados en la dirección opuesta, la formula de actualización de los pesos es

$$\Delta w_{ij} = -\eta \frac{\delta E}{\delta w_{ij}}$$

donde la razón de aprendizaje $\eta > 0$ es una constante positiva pequeña. Si la derivada es positiva entonces los pesos cambian a negativos y viceversa. Esta aproximación de descenso de gradiente con η es infinitesimal. Pequeños valores de η significa un largo tiempo de aprendizaje. La red usualmente se inicializa con pequeños pesos aleatorios. Valores aleatorios son necesarios para romper la simetría mientras que pequeños valores son necesarios para evitar saturación inmediata de la no linealidad sigmoidal.

4.2.4 Aprendizaje por lotes

En el modo de aprendizaje por lotes, cada patrón *p* es evaluado para obtener los términos derivados $\delta E_p / \delta w$; ellos son sumados para obtener la derivada total

$$\frac{\delta E}{\delta w} = \sum_p \frac{\delta E_p}{\delta w}$$

y sólo entonces los pesos son actualizados. Esto viene de la regla de derivadas para la suma. El término individual $\delta E_p / \delta w$ es obtenido por la aplicación del método a cada patrón de *p*.

Los pasos básicos son

- Para cada patrón *p* en el conjunto de entrenamiento,
  1. aplicar el patrón *p* e iniciar la propagación para obtener la salida de la red y
  2. calcular el error del patrón *Ep* y realizar la retropropagación para obtener el patrón simple de derivadas $\delta E_p / \delta w$
- Sumar todos los patrones simples para obtener la derivada total.
- Actualizar los pesos
- Repetir

Cada paso a través del conjunto de entrenamiento es llamado época.

El gradiente es calculado exactamente y los pesos cambiados son proporcionales al gradiente, también el aprendizaje por lotes aproxima el descenso gradiente cuando el tamaño de los pasos η es pequeño. En general, cada actualización de peso reduce el error por sólo una pequeña cantidad así que muchas épocas son necesarias para minimizar el error.

4.2.5 Aprendizaje en línea

En el aprendizaje en línea, los pesos son actualizados después de cada presentación del patrón. Generalmente un patrón *p* es escogido aleatoriamente y presentado a la red. La salida es comparada con el objetivo para el cual el patrón y los errores son retropropagados, para obtener el patrón de la derivada simple $\delta E_p / \delta w$. Los pesos son entonces actualizados inmediatamente, usando la gradiente del error del patrón simple. Generalmente, los patrones son presentados de manera aleatoria, constantemente cambiando el orden para evitar efectos cíclicos.

Los pasos son:

- Escoger un patrón *p* aleatorio del conjunto de entrenamiento
  1. aplicar el patrón *p* e iniciar la propagación para obtener las salidas de la red *y*

2. calcular el error del patrón E_p y realizar la retropropagación para obtener el patrón simple de derivadas $\delta E_p / \delta w$
3. Actualización de los pesos inmediatamente usando un patrón simple de derivadas $w(t+1)=w(t)-\eta \frac{\delta E_p}{\delta w}$ $w(t+1)=w(t)-\eta \frac{\delta E_p}{\delta w}$

- Repetir

Una ventaja de este enfoque es que no es necesario almacenar y sumar individualmente las contribuciones $\delta E_p / \delta w$. Cada patrón derivado es evaluado, usado inmediatamente y después es descartado. Otra posible ventaja es que la actualización de muchos pesos ocurre en una cantidad de tiempo finita

5 El método experimental

Con la metodología de las redes neuronales y empleando el algoritmo de retropropagación pretendemos acertar a la solución del problema de la resolución automática de la homonimia morfológica para el español.

Para desarrollar este modelo es necesario implementar distintos tipos de redes neuronales, realizar diferentes experimentos y realizar modificaciones a los diferentes parámetros que se encuentran en cada una de las redes. Los resultados producidos por las redes, son examinados.

Con todos estos pasos se creo un método de resolución, para realizar una aproximación a la solución del problema planteado anteriormente.

En esta sección describimos la funcionalidad de los objetos implementados para el funcionamiento de estas redes así como las características y propiedades de estos objetos.

5.1 Implementación de la red neuronal de retropropagación

5.1.1 Clase TCapa

En esta clase se crean las capas de la red neuronal, cada capa contiene *n* número de neuronas, que son almacenadas en un vector de tamaño *n*. Todas las neuronas de la capa de entrada sólo reciben las entradas de la red.

Las neuronas de las capas intermedias reciben entradas de todas las neuronas de la capa anterior y envían su salida a todas las neuronas de la capa posterior, las neuronas de la capa de salida no envían su salida a una capa posterior.

En esta clase contamos con el constructor que inicializa los valores del número de neuronas y las neuronas igual a cero. El destructor de esta clase cuenta con el método *VaciarCapa( )* el cual elimina todas las neuronas de cada una de las capas, vaciando el vector donde se almacenan las neuronas. Es en esta clase donde se introducen cada una de las neuronas en las capas, con el método *LeerBinario(Archivo)* que tiene como parámetro el nombre del archivo que contiene las entradas de las neuronas y *EscribirBinario(Archivo)* el cual escribe los datos miembro de cada uno de los objetos neurona en un archivo, estos son almacenados en forma binaria.

5.1.2 Clase TMisDatos

En esta clase tenemos el valor de todos los datos almacenados en un vector, también copiamos el valor de un vector de caracteres con un tamaño total igual a DATA_SIZE+1, a otro vector de la misma magnitud para almacenamiento. En este vector el valor de DATA_SIZE nos indica el número total de neuronas que existen en nuestra capa de entrada. En esta clase se realiza el almacenamiento de cada dato de entrada de las neuronas, que forman parte de la capa de entrada, este dato de entrada es almacenado en un vector de caracteres. Esto se debe a que las entradas de nuestras neuronas son caracteres de tipo binario.

5.1.3 Clase TNeurona

En esta clase abstraemos a una neurona como un elemento computacional individual. El dato miembro *salida* de la neurona corresponde a la frecuencia de descarga de la neurona y el dato miembro *peso* corresponde a la fuerza de la conexión entre las neuronas, estas cantidades son representadas en el tipo *double*. El peso es almacenado en una estructura de tipo vector, para el cual contamos con el valor del número de pesos que pueden ser almacenados.

El estado de activación de cada neurona esta asociado con los términos de error, *delta (retropropagación del error)*, que funciona de manera diferente, dependiendo si es unidad de salida o unidad oculta. Esta clase cuenta con un constructor que inicializa el número de pesos y el vector de pesos, el destructor contiene un método llamado *VaciarNeurona( )* el cual elimina los datos almacenados en el vector de pesos.

5.1.4 Clase TRed

Aquí se implementan las uniones, que mantienen unidos a los objetos neurona de nuestras redes neuronales de retropropagación. Tenemos un vector de objetos TCapa, en el cual se encuentran almacenadas las capas que a su vez almacenan los objetos TNeurona, que forman la TRed.

En otro vector almacenamos los *errores* producidos por la red. También tenemos los datos miembros DATA_SIZE, los cuales son del tipo *int* y nos indica el tamaño máximo de los datos de entrada en la red neuronal, el parámetro del error máximo tolerado se almacena en la variable ErrorMaxTolerado.

En esta clase se asignan los valores a todos los parámetros necesarios para el funcionamiento de la red de retropropagación. Los cuales son:

- el factor de aprendizaje,
- el número de capas intermedias,
- el número de neuronas en cada capa,
- número de neuronas de entradas a la red,
- número de neuronas de salida de la red.

Todos estos valores son ajustables por el usuario del sistema, aunque también pueden ser asignados automáticamente. El porcentaje de entrenamiento y el factor de aprendizaje son valores de tipo *double*, los demás son valores de tipo *entero*.

Esta clase tiene un vector que almacena los datos que pertenecen a las salidas esperadas, para después compararlas con el vector de salidas de la red y encontrar el error que existe entre estos datos.

Esta clase cuenta con un apuntador del tipo TTodosDatos, que nos permite acceder a los valores de la clase TMisDatos, que tiene las entradas de la red.

Los métodos con los que cuenta esta clase son, *PreparaRed( ),* el cual tiene como parámetros todos los datos de inicialización de los datos de la red, que ya hemos mencionado, el método *PrepararDatos( )* nos regresa un valor verdadero si se introducen correctamente los valores de las entradas y los valores de los vectores de *salidas_esperadas* y de *errores*, de otro modo nos regresa un valor falso.

El método *CicloForward( )* propaga hacia adelante las entradas de la red, también actualiza los pesos y la señal de salida de la red. En el método *LeerDatosdeArchivo (Archivo)* se leen los datos de un archivo que contiene las entradas ya codificadas y se almacenan los datos en el vector de tipo TMisDatos. El método *EscribirSalidas(Archivo),* escribe en el archivo las salidas obtenidas de la red, en los distintos ciclos.

En el método *CicloDeEntrenar( )* se realiza el entrenamiento de los datos y se repite este proceso hasta que el término del error ε, es aceptablemente pequeño, este método regresa el valor del error calculado, para cada uno de los patrones aprendidos. También se llevan a cabo otras procesos como actualización del valor delta δ y la actualización de los nuevos valores para los vectores de pesos y salidas. Usamos dos formas de retropropagación, en la primera forma se calculan los valores deltas para cada una de las

capas y en la segunda forma, modificamos los pesos y propagamos hacia atrás, capa por capa.

El constructor de TRed inicializa los miembros de la clase Red, el destructor hace un llamado al método *VaciarRed( ),* el cual elimina todas las neuronas almacenadas en cada una de las capas, de la misma manera elimina los datos almacenados en los vectores de *salidas_esperadas* y *errores*. Asimismo elimina los datos del vector *todos_datos*.

5.1.5 Clase TTodosDatos

Esta clase contiene como miembros a un vector de apuntadores de objetos *TMisDatos*, así como el número de datos que se pueden almacenar en el vector. En el destructor de esta clase se eliminan los datos almacenados en el vector *datos* y el dato miembro del número de datos *NumDatos* se inicializa con 0.

5.2 Preparación de los datos de entrada

Los datos de entrada se obtuvieron del corpus *all.tgd*, del cual, cada palabra del corpus fue dividida en las siguientes partes: palabra, lema y lexema, cada una de estas partes fueron colocadas en el archivo *all_win.tgd* el cual tiene un tamaño de 100 Mb.

El siguiente paso, fue obtener el primer dato que correspondía a la clase gramatical a la cual pertenece el lexema de esta palabra. Esto se realizo para cada una de las palabras en el corpus *all_win.tgd*. Este dato solamente podría ser de un tipo. Los distintas tipos de etiquetas posibles que corresponden a las clases gramaticales a la cual pueden pertenecen las palabras, son 15 y se listan abajo:

A para adjetivos.
C para conjunciones
D para determinantes
I para interjecciones
M para numerales
N para sustantivos
P para pronombres
R para adverbios
S para preposiciones

T para artículos
V para verbos
W para datos
X para extranjero
Y para verbo auxiliar
Z para valor numérico

Al obtener estos datos se almacenaban en un nuevo archivo llamado *Corpus_ltr.tgd* con un tamaño de 2.70MB, este archivo es un corpus que contiene solamente el valor de las etiquetas. Cada etiqueta tenia un código binario asociado, mediante el cual se ponía un 1 en la posición correspondiente al dato en nuestra lista y a las demás posiciones se les asignaba un 0, por ejemplo para la etiqueta M, que se encuentra en la quinta posición de nuestra lista, se le asignaba un uno en la quinta posición y las demás posiciones contienen el valor de cero, así *M = 000010000000000*. Para crear esta codificación creamos una función llamada *generar_0_1(Dato)* la cual recibe como parámetro la etiqueta del dato a convertir. Esta codificación es necesaria debido a que a cada tipo de categoría gramatical le corresponden solamente 15 neuronas de entrada y cada entrada acepta únicamente un valor binario. De esta manera se creaba un vector binario 15-dimensional. Así que cada etiqueta que correspondía a una categoría gramatical era codificada. Todas las codificaciones de las etiquetas que formaban parte de una frase eran concatenadas y de esta manera formar parte de los datos de entrada de nuestras redes neuronales. Nuestros identificadores para delimitar las frases compuestas de etiquetas fueron todos los signos de puntuación (, . : ; ¿ ? ¡ ! ( ){ }[ ] < >).

5.3 Preparación del corpus con homonimia

Para este trabajo ocupamos el corpus *all_sgh.mrf* el cual es un corpus que obtuvimos a partir Corpus Lexesp. El corpus Lexesp es un corpus de lengua escrita del español que consta de un total de 5,586,158 formas, de diferentes géneros (noticias, revistas, deportes, literatura, artículos científicos, etc). Lexesp consta de 97,035 formas analizadas morfológicamente de manera automática y desambiguada manualmente. La meta del proyecto Lexesp es crear una gran base de datos del lenguaje usado de manera

ordenada, para permitir y fortalecer las actividades de investigación en distintos campos abarcando desde la lingüística hasta la medicina, a través de la psicología y la inteligencia artificial entre otros, este proyecto se realizó en el laboratorio de lingüística computacional de la Univesitat Politécnica de Catalunya [46].

El corpus lingüístico *all_win.tgd* es subconjunto del corpus *all_sgh.mrf* con un tamaño de 183 Mb. Él cual contiene más información morfológica de las palabras que las que se encuentran en su corpus subconjunto. El corpus

En cada línea de estos corpus aparece una palabra, de la cual se escribe también todas las raíces de la forma de palabra para cada lexema, más la información morfológica. La información morfológica aparece de la siguiente manera:

Tabla 5-5.1 Información morfológica

Stem	Word	Info	Mark1	Mark2
Gato	gato	N		
Gafa	gafas	N	P	
Acert	acertar	VI	M1	1
Aciert	acertar	VI	M1	2

El campo *Word* contiene el lema, el campo *Stem* contiene la raíz, el campo *Info* contiene la clase gramatical, los campos *Mark1, Mark2* contienen las marcas gramaticales adicionales. Por ejemplo, el campo *Mark1* del segundo registro que es (P) indica que se trata de un *pluralia tantum* (es decir, si se tratara de generar su singular, obtendríamos un error) y para los últimos dos registros indica el modelo de conjugación semi-irregular al que pertenece el verbo. El campo *Mark2* para los últimos dos registros señala la raíz original (1) y la segunda raíz posible (2).

El corpus *all_sgh.mrf* se encuentra codificado de la siguiente manera:

```
1. La la 810 TDFS0 ella B1020 PP3FSR00
2. elección elección 010 NCFS000
3. de de A1 SPS00
4. esa ese 3100 DD3FS00 ese B1026 PD3FS000
5. novela novela 010 NCFS000 novelar 6201032 VMMP2S0
6. de de A1 SPS00
7. E._M._Forster e._m._forster 5 NP00000
8. no no 44 RG000
9. es ser 7202030 VAIP3S0
10.a a A1 SPS00
11.mi mi 3301 DP1CS00
12.juicio juicio 000 NCMS000
13.enteramente entero 43 RG000
14.satisfactoria satisfactorio 110 AQ0FS00
15.; ; E4 Fpc
16.gran gran 130 AQ0CS00
17. parte parte 010 NCFS000 partir 6201032 VMMP2S0
18.de de A1 SPS00
19.sus su 3311 DP3CP00
20.calidades calidad 011 NCFP000
21.literarias literario 111 AQ0FP00
22.se él B3320 PP3CNR00
23.pierde perder 6201032 VMMP2S0 perder 6202030 VMIP3S0
24.al al A0 SPCMS
25.pasar pasar 6223503 VMN0000
26.al al A0 SPCMS
27.cine cine 000 NCMS000
28.; ; E4 Fpc
```

Que es la codificación del texto:

"La elección de esa novela de E. M. Forster no es a mi juicio enteramente satisfactoria; gran parte de sus cualidades literarias se pierde al pasar al cine;"

En este tipo de codificación tenemos la palabra, de la cual se escriben los posibles lemas, más el resultado de su análisis morfológico como vemos en la primera línea tenemos la palabra *La*, después tenemos el lema *la* con el siguiente resultado de su análisis

TDFS0, donde *T* = artículo, *F* = femenino, *S* = singular, después aparece el lema *ella* con el resultado de su análisis que es PP3FSR00 donde *P* = pronombre, *3* = tercera persona, *F* = femenino, *S* = singular.

El corpus *all_win.tgd* esta codificado de la siguiente manera:

```
1. Quieres querer VMIP2S0
2. un un TIMS0
3. refresco refresco NCMS000
4. ? ? Fci
5. Fg
6. No no RG000
7. . . Fp
8. He haber VAIP1S0
9. tomado tomar VMPP0SM
10.tres tres NCCP000 tres NCMS000
11.en en SPS00
12.casa casa NCFS000
13.. . Fp
```

En este corpus sólo aparece un lema y su análisis, por ejemplo en la primera línea tenemos la palabra *Quieres* y enseguida el lema *querer* con el resultado de su análisis igual a VMIP2S0 donde *V* = verbo, *I* = infinitivo, *2* = segunda persona, *S* = singular. En la línea número 7 tenemos un singo de puntuación, en el cual no se realiza ningún análisis.

Conociendo como están codificados los dos corpus, nuestra tarea fue obtener un nuevo corpus el cual se crea como resultado de buscar las oraciones del corpus *all_win.tgd* dentro del corpus *all_sgh.mrf* y concatenar estas oraciones en el nuevo corpus de nombre *resultado.tgd* , el cual se encuentra codificado de la siguiente manera:

```
1.  Más más RG000 Más más 42 RG000 más 003 NCMN000
2.  difícil difícil AQ0CS00 difícil difícil 130 AQ0CS00
3.  es ser VAIP3S0 es ser 7202030 VAIP3S0
4.  que que CS00 que que 99 CS00 que B3323 PR3CN000
5.  haya haber VASP1S0 haber VASP3S0 haya haber 7200031 VASP1S0 haber VASP3S
6.  proporcionado proporcionar VMPP0SM proporcionado proporcionado 100 AQ0MS00
7.  alguna algún DI3FS00 alguna algún 3102 DI3FS00 alguno B1021 PI3FS000
8.  terapéutica terapéutico AQ0FS00 terapéutica terapéutica 010 NCFS000
9.  inseguridad inseguridad NCFS000 inseguridad inseguridad 010 NCFS000
10. a a SPS00 a a A1 SPS00
11. los el TDMP0 los el 801 TDMP0 Úl B0120 PP3MPR00
12. lectores lector NCMP000 lectores lector 001 NCMP000 lector 101 AQ0MP00
13. más más RG000 más más 42 RG000 más 003 NCMN000
14. seguros seguro AQ0MP00 seguros seguro 001 NCMP000 seguro 101 AQ0MP00
15. ,
```

Este corpus consta de la concatenación de las líneas con las mismas frases en los distintos corpus. Por lo que podemos ver la repetición de las palabras, así como otro lema y la información morfológica de este. Por ejemplo en la línea 1 tenemos la palabra *más* y el lema *más* como adverbio *(R)* y el lema *más* como sustantivo *(N)*.

El corpus *resultado.tgd* tiene un tamaño de 247 Mb. El cual fue creado de esta manera para poder describir la homonimia morfológica de las palabras, del cual sólo vamos a trabajar con las etiquetas de las clases gramaticales.

Del corpus *resultado.tgd* vamos a crear otro corpus de nombre *homonimia1.tgd* con un tamaño de 16.7 Mb. El cual esta formado solamente por las etiquetas de los lemas que nos produce el analizador morfológico AGME, si alguna etiqueta se repite, solamente nos quedamos con la primera y si la siguiente etiqueta es diferente la escribimos en el corpus. Si encontramos un signo de puntuación lo escribimos en el archivo, este signo nos servirá mas adelante para limitar las frases. Un ejemplo del resultado de las etiquetas sobre el fragmento del corpus *resultado.tgd* mostrado anteriormente es el siguiente:

1.RN
2.A
3.V
4.CP
5.V
6.VA
7.DP
8.AN
9.N
10.S
11.TP
12.NA
13.RN
14.AN
15.,

Esta es la información contenida en el corpus *homonimia1.tgd*, que como vemos son etiquetas de las clases gramaticales sin repetición.

5.4 Entrenamiento de la red

En esta parte describimos cada uno de los pasos necesarios para el entrenamiento de nuestras redes de retropropagación, para resolver el problema de etiquetación de clases gramaticales.

5.4.1 Paso 1

Inicializar los pesos de todas las redes con valores pequeños aleatorios usando $w_{i,j\to i+1,l}(0)=((Random(200000)\text{ - }100000)/100000)$.

5.4.2 Paso 2

Presentamos un patrón de entrada el cual se formo del lexema del corpus, generamos la codificación binaria y según la palabra y su contexto se asigna a una de las 8 redes que hemos tomado como base, aquí también tomamos la codificación de la salida deseada, que al mismo tiempo forma parte de los datos de entrada .

Las 8 redes que tenemos son:

palabra_deseada	C1				Red[0]
C1	palabra_deseada				Red[1]
palabra_deseada	C1	C2			Red[2]
C1	palabra_deseada	C2			Red[3]
C1	C2	palabra_deseada			Red[4]
C1	palabra_deseada	C2	C3		Red[5]
C1	C2	palabra_deseada	C3		Red[6]
C1	C2	palabra_deseada	C3	C4	Red[7]

Donde palabra_deseada es la posición donde se encuentra la etiqueta que queremos obtener como salida deseada. *Ci*, con $1 \le i \le 4$, son los posibles contextos tomados por las redes a partir de la palabra_deseada.

5.4.3 Paso 3

Calcular la salida actual de todas las redes, para ello presentamos las entradas codificadas correspondientes a cada tipo de red y calculamos las salidas de cada capa hasta llegar a la última capa de la red, la cual contiene los datos de salida de cada red.

- Se calculan las entradas netas para las neuronas ocultas procedentes de las neuronas de entrada. Para una neurona *j* oculta:

$$net_{p,j} = \sum_{i=1}^{N} w_{ji} x_{pi}$$

- Se calculan las salidas de las neuronas ocultas:

$$y_{p,j} = f_j(net_{p,j})$$

- Se calculan las salidas de las neuronas de salida:

$$net_{p,k} = \sum_{j=1}^{L} w_{k,j} y_{p,j}$$

$$y_{p,k} = fk(net_{p,k})$$

5.4.4 Paso 4

Calcular los términos de error para todas las neuronas. Si la neurona *k* es una neurona de la capa de salida, el valor delta es:

$$\delta^{o}{}_{p,k} = (d_{p,k} - y_{p,k}) f^{o}{}_{k}(net^{o}{}_{p,k})$$

Si la neurona *j* no es de salida

$$\delta_{p,j} = x_{p,i}(1 - x_{p,i}) \sum_{k} \delta^{o}{}_{p,k} w^{o}{}_{k,j}$$

5.4.5 Paso 5

Actualización de los pesos. Comenzamos por la capa de salida y trabajamos hacia atrás, hasta llegar a la capa de entrada ajustando los pesos de la forma siguiente:

Para los pesos de la capa de salida:

$$w^{o}_{k,j}(t+1) = w^{o}_{k,j}(t) + \Delta w^{o}_{k,j}(t+1)$$
$$\Delta w^{o}_{k,j}(t+1) = \alpha \delta^{o}_{p,k} y_{p,j}$$

Para los pesos de las neuronas de la capa oculta:

$$w_{j,i}(t+1) = w_{j,i}(t) + \Delta w_{j,i}(t+1)$$
$$\Delta w_{j,i}(t+1) = \alpha \delta_{p,j} x_{p,i}$$

5.4.6 Paso 6

El proceso se repite hasta que el promedio de los datos obtenidos de la función de error de todas las redes, sean menores que el valor *0.001*. La función de error es $Ep = \frac{1}{2}\sum_{i}(d_i - o_i)^2$ donde d_i es el *i*-ésimo elemento de la salida deseada y o_i es el *i*-ésimo elemento de la red de salida.

5.5 El experimento

Describiremos el proceso que se siguió al seleccionar los recursos que permitieron implementar el sistema, para poder evaluar el modelo desarrollado.

El corpus lingüístico *all_win.tgd* usado para el entrenamiento, es un corpus completamente etiquetado por el analizador morfológico para el español AGME.

La morfología del español no es materia trivial. Como lenguaje flexivo, el español muestra una gran variedad de procesos morfológicos, particularmente los no concatenativos. Algunos de los problemas que se presentan en un procesador morfológico del español, a decir de [11], son:

- Un paradigma verbal muy complejo. Para tiempos simples, hay alrededor de 61 formas flexivas, incluyendo el duplicado subjuntivo pasado imperfecto (6 formas). Si agregamos las 45 posibles formas para tiempos compuestos, hay 112 formas flexivas posibles para cada verbo. Pero en nuestro caso ignoramos los tiempos compuestos, porque cada cadena de caracteres en nuestro nivel se procesa por separado.
- La frecuente irregularidad de raíces y terminaciones verbales. Verbos muy comunes, como tener, poner, poder, hacer, etc., tienen hasta 7 raíces: hac-er, hag-o, hice, ha-ré, hi-zo, haz, hech-o.
- Huecos en algunos paradigmas verbales. En los llamados verbos defectivos algunas formas se pierden o simplemente no se usan. Por ejemplo, los verbos meteorológicos como llover, nevar, etc., son conjugados sólo en tercera persona del singular. Otros son más peculiares, como abolir que falla en primera, segunda y tercera persona del singular y tercera del plural del presente de indicativo, en presente del subjuntivo y en la segunda persona del singular de la forma imperativa. En otros verbos, los tiempos compuestos se excluyen del paradigma, como en soler.
- Participios pasados duplicados. Una cantidad de verbos tienen dos formas alternas, ambas correctas, como impreso, imprimido. En tales casos, el análisis debe tratar las dos formas como correctas.

Existen algunos verbos altamente irregulares que pueden ser manejados sólo al incluir sus formas directamente en el diccionario (como ser, haber, etc.). Algunos sustantivos y adjetivos presentan formas alternativas correctas para el plural (ej. bambú, bambús, bambúes).Hay un pequeño grupo (3%) de sustantivos invariantes con la misma forma para el singular y el plural (ej. crisis). Por otro lado, 30% de los adjetivos presentan la misma forma para el masculino y el femenino (ej. azul). Existen también los *singularia tantum*, donde sólo se usa la forma singular, como en estrés; y los *pluralia tantum*, donde sólo se usa la forma de plural, como en matemáticas. A diferencia de la morfología verbal,

los procesos nominales no producen cambios internos en la raíz causado por la adición de un sufijo de género o plural, a pesar de que puede haber muchos alomorfos producidos por cambios de ortografía (luz, luc-es). Obviamente, para el sistema de análisis automático se tratan como raíces diferentes.

Todos estos fenómenos sugieren que no hay un modelo simple (unificado como el modelo de dos niveles) para el tratamiento automático de la morfología del español.

El modelo general de análisis morfológico mostrado en la figura 6-1 e implementado en nuestra aplicación, es simple: dependiendo de la forma de palabra de entrada, se formula alguna hipótesis de acuerdo con la información del diccionario y otros criterios y se generan las formas correspondientes para tal hipótesis. Por ejemplo, para la flexión *-amos* y la información del diccionario para la raíz que corresponde al verbo de la clase 1, se genera la hipótesis de primera persona, plural, indicativo presente (entre otras), etc. Las formas generadas según las hipótesis se comparan con la original, en caso de coincidencia las hipótesis son correctas. Más detalladamente, dada una cadena de letras (forma de palabra), la analizamos de la siguiente manera:

1. Quitar letra por letra (también siempre se verifica la hipótesis de la flexión $\emptyset$).
2. Verificar si existe flexión.
3. Si existe flexión entonces leer del diccionario la información de la raíz y llenar la estructura de datos correspondiente (si no existe la raíz, regresar al paso 1).
4. Si no existe la flexión, regresar al paso 1.
5. Formular hipótesis.
6. Generar la correspondiente forma gramatical de acuerdo a nuestra hipótesis y la información del diccionario.
7. Si el resultado obtenido coincide con la forma de entrada entonces la hipótesis es aceptada. De otra forma, el proceso se repite desde el paso 3 con otra raíz homónima (si la hay) o desde el paso 1 con otra hipótesis sobre la flexión.

Figura 5-5.1 Proceso de análisis morfológico

AGME no sobre-analiza, es decir sólo se procesan las formas correctas. La complejidad del sistema morfológico de un lenguaje para la tarea de análisis automático no depende del número de las clases gramaticales ni de la homonimia de las flexiones, sino del número y tipo de las alternaciones en raíces, las cuales no se puede saber sin consultar el diccionario.

AGME produce información del lema e información gramatical. A partir de este corpus creamos un archivo que cuenta solamente con la categoría gramatical de cada palabra del corpus, llamado *Corpus_ltr.tgd*. Las posibles categorías son adjetivo, conjunción, determinante, intersección, numeral, sustantivo, pronombre, adverbio, preposición, artículo, verbo, dato, extranjero, verbo auxiliar y número. Toda la información de este archivo, se dividía en frases. Para obtener una frase, era necesario encontrar cualquier signo de puntuación, con lo que las palabras que se encontraban entre dos signos de puntuación formaban una frase. Cada frase forma parte de nuestros datos de entrada, las cuales se introducían en las diferentes redes dependiendo del contexto de cada frase. El entrenamiento se llevo a cabo de la siguiente manera, al conocer el etiquetado de cada una de las palabras en una frase y la cantidad de palabras que forman la frase, se introducían las etiquetas de las palabras en cada una de las redes, donde cada red limitaba el tamaño de la frase, esto debido a la capacidad de entrada de cada red. Si el número de palabras en la frase era mayor que 5, esta frase era dividida en frases con un tamaño menor o igual que 5. Al estar divididas las frases en etiquetas de clases gramaticales, se introducían las etiquetas de cada palabra en todas las redes, donde cada red era capaz de restringir la entrada de algunas palabras etiquetadas. Cada red eliminaba una etiqueta de una palabra (*P*), esta

etiqueta formaba parte de nuestra salida deseada, para esta red. Las demás etiquetas de palabras, formaban parte de los diferentes contextos, los cuales eran identificados con las letras *C1, C2, C3, C4,* todo esto se puede observar en la tabla 5-2.

Tabla 5-5.2 Entrenamiento de las redes

Frase	Salida deseada	Red
P-C1	P	0
C1-P	P	1
P-C1-C2	P	2
C1-P-C2	P	3
C1-C2-P	P	4
C1-P-C2-C3	P	5
C1-C2-P-C3	P	6
C1-C2-P-C3-C4	P	7

Esto se repetía hasta que el error mínimo era menor que 0.001 ó el archivo con las frases se haya leído completamente más de 200 veces.

Al acabar el entrenamiento de las redes, la fase siguiente es la de prueba del etiquetador, para probar el etiquetador tomamos el archivo *Corpus_ltr.tgd* el cual contiene todas las frases y estas contienen palabras con sus respectivas etiquetas de la clases gramaticales a la que pertenece. Encontramos un total de 797,698 frases que se acoplaron e introdujeron en las distintas redes, todo esto se realizaba dentro de un ciclo de lectura del archivo, donde se eliminaba una etiqueta de palabra de la frase. El valor eliminado que es conocido por nosotros se colocaba como un dato desconocido, en los datos de entrada de las redes, este dato desconocido nos servia como referencia, para saber si nuestro resultado de la etiquetación era correcto.

La red acertaba si el valor de la etiqueta desconocida era igual a la etiqueta obtenida en la salida de la red. Si en mas de una red se colocaban frases distintas, entonces se hacia un promedio a las salidas de las redes en las que se acoplo la frase y el resultado máximo del promedio era tomado como el valor de la etiqueta a la palabra desconocida, este resultado, se comparaba con el valor deseado y si era la misma etiqueta de palabra entonces se contaba como un acierto de lo contrario era un resultado erróneo. De esta manera se verificaban los aciertos de nuestro modelo.

Supongamos como un ejemplo que tenemos la frase que contiene las etiquetas *YVNSN*, las cuales serán introducidas como datos de entrada a cada una de las 8 redes de la siguiente manera:

Tabla 5-5.3 Ejemplo de datos de entrada

Subfrases	Etiqueta deseada	Red
_V	Y	0
_VN	Y	2
_N	V	0
Y_	V	1
_NS	V	2
Y_N	V	3
Y_NS	V	5
_S	N	0
V_	N	1
_SN	N	2
V_S	N	3
YV_	N	4
V_SN	N	5
YV_S	N	6
YV_SN	N	7

Como se puede observa en la tabla anterior, la subfrase YVN, se acoplo en las redes 0,1,3 con los datos de entrada _V, Y_, Y_N, los resultados de estas 3 redes se promediaron para obtener la salida de la etiqueta V (Verbo), la etiqueta asignada a la palabra desconocida fue la salida de mayor valor.

En problemas multiclase como el de este trabajo, cada nodo de salida corresponde a una clase separada y la clase a la cual el patrón es asignado debe ser indicada por el nodo con la salida más alta, para ese patrón. También pudimos verificar en que redes y en que contextos se obtenían mejores resultados.

Tabla 5-5.4 Prueba de las redes

Frase	Salida deseada	Red
Pdes.-C1	Etiqueta(P)	0
C1-Pdes	Etiqueta(P)	1
Pdes-C1-C2	Etiqueta(P)	2
C1-Pdes-C2	Etiqueta(P)	3
C1-C2-Pdes	Etiqueta(P)	4
C1-Pdes-C2-C3	Etiqueta(P)	5
C1-C2-Pdes-C3	Etiqueta(P)	6
C1-C2-Pdes-C3-C4	Etiqueta(P)	7
Pdes = Etiqueta desconocida		

5.6 Resolución de la homonimia morfológica

Para el problema de resolución de homonimia morfológica trabajamos con el corpus *homonimia1.tgd* con un tamaño de 16.7 Mb. En este corpus se encuentran todas las etiquetas posibles, de los lemas de las palabras en la frase. El siguiente paso, fue almacenar las etiquetas correctas de las frases, ya que estas nos van a servir como referencia.

Al tener guardadas las etiquetas correctas de las palabras, en una frase, obteníamos las distintas etiquetas, que correspondían a los posibles lemas, estas etiquetas se almacenaban como etiquetas variables.

También se almacenaba un identificador para las etiquetas variables, ya que era necesario saber a que palabra de la frase pertenecía esta etiqueta. Para resolver este problema, el identificador nos indicaba en que palabra de la frase se encontraba esta etiqueta y por que valor era posible sustituirle.

Al contar con esta información de la etiqueta correcta y las posibles etiquetas, almacenadas como variables, cambiamos un valor correcto de la etiqueta de la palabra por sólo un valor de las etiquetas posibles, que forman parte de nuestras etiquetas variables. La frase que surgía en esta etiquetación se introdujo en las distintas redes, respetando su acoplamiento y además tomando en cuenta que a la etiqueta variable, le correspondía la posición de la etiqueta desconocida y las demás etiquetas formaban parte de los contextos.

Tabla 5-5.5 Configuración de las redes

Red	Configuración
0	EV-C1
1	C1-EV
2	EV-C1-C2
3	C1-EV-C2
4	C1-C2-EV
5	C1-EV-C2-C3
6	C1-C2-EV-C3
7	C1-C2-EV-C3-C4

En la tabla 5-5 tenemos en la primera columna los nombres de la redes y en la segunda columna tenemos las configuraciones de cada una de las redes. Donde EV es el valor de la etiqueta variable y C1, C2, C3, C4 son los diferentes contextos.

Por ejemplo si tenemos en nuestro archivo las siguientes etiquetas, correspondientes a una frase:

V

TN

NTV

,

TVM

N

,

Lo que hacemos es guardar el valor correcto de las etiquetas, que para este caso son las etiquetas "*VTN,TN*" las cuales se encuentran en la primera posición de las líneas del archivo.

Tabla 5-5.6 Tipos de etiquetas

Etiquetas correctas	Etiquetas variables
V	
T	N
N	TV
.	.
T	VM
N	

Las etiquetas variables se sustituían solamente una vez y en una posición de alguna etiqueta correcta, con lo que para el ejemplo anterior se forman las siguientes configuraciones:

Tabla 5-5.7 Tipos de configuraciones

Red	Configuración	Entradas a Red
0	EV-C1	VT, NN, TN
1	C1-EV	VN, TT, TV
2	EV-C1-C2	
3	C1-EV-C2	VNN
4	C1-C2-EV	VTT, VTV
5	C1-EV-C2-C3	
6	C1-C2-EV-C3	
7	C1-C2-EV-C3-C4	

Este procedimiento se repetía por cada frase que se encontraba en el corpus *homonimia1.tgd*, si el tamaño de la frase era mayor que 5 etiquetas, entonces la frase se reducía a un tamaño de 5 etiquetas y con el resto de la frase se creaba otra frase, que se acoplaba de la misma manera a las distintas redes. Esto se realizo ya que el tamaño de las redes esta limitado a 4 contextos más una variable, lo que nos da un total de 5 etiquetas como máximo.

En este corpus se configuraron de esta manera 1,830,349 frases las cuales se evaluaron en las distintas redes.

6 Resultados

6.1 Análisis de resultados

En esta parte describiremos el procedimiento empleado para determinar la validación de las ideas expuestas anteriormente. Primero describiremos el tipo de evaluación a efectuar y las métricas seleccionadas.

La medida de certeza usada, es la fracción de muestras acertadas:

$$A = \frac{No\,.\,A}{Total}$$

Donde el numerador *No.A* es el número de muestras acertadas y el denominador de esta fracción es el número total de muestras. Las salidas son interpretadas como probabilidades por lo que el error de medida elegido es la probabilidad de certeza.

El resultado de este trabajo esta dividido en dos partes, en la primera parte analizamos los resultados obtenidos al trabajar con la etiquetación de clases gramaticales, utilizando redes neuronales con el algoritmo de retropropagación y en la segunda parte describimos el análisis de los resultados para la resolución automática de la homonimia morfológica.

En la primera parte modificamos los valores de algunos de los parámetros importantes de la red de retropropagación como son, el número de capas, el factor de aprendizaje y el número de iteraciones ocupadas para el entrenamiento de las redes. También se analizan las etiquetas de palabras con mayor cantidad de aciertos.

En la segunda parte describimos el análisis de los resultados para la resolución automática de la homonimia morfológica, ocupando redes neuronales y el algoritmo de retropropagación. La entrada a estas redes contienen las etiquetas de las palabras que conforman las frases, donde solo un valor de estas etiquetas es erróneo y esta entrada se ejecuta en la red de retropropagación, obteniendo un resultado en la salida de la red, el cual es comparado con la etiqueta correcta, y se puede conocer la certeza o falsedad del resultado. Aquí ocupamos la misma medida de certeza, para contar nuestros valores ciertos y erróneos totales.

6.2 Resultados de la etiquetación con redes neuronales

Encontramos un total de 797,698 frases de las cuales para redes neuronales con 3 capas ocultas y un factor de aprendizaje igual a 0.25, obtuvimos 341,535 aciertos que equivale al 42.8% y 456,163 resultados erróneos que equivale a un 57.2%.

En redes neuronales con 2 capas ocultas, para la misma cantidad de frases y con un factor de aprendizaje de 0.25 encontramos 369,893 aciertos que equivale a un 46.3% y 427,805 resultados erróneos equivalente a un 53.7%.

Para un ciclo de entrenamiento de las redes que constaba de la lectura del archivo del corpus etiquetado durante 250 veces, obtuvimos 370,017 aciertos que nos da un porcentaje de 46.38% y un total de 427,681 valores erróneos equivalente al 53.62%.

Al verificar estos resultados nos dimos cuenta que los valores de las palabras etiquetadas como adjetivos, intersecciones, numerales sustantivos, extranjeros y verbos auxiliares nunca aparecen como valor máximo.

Obtuvimos los valores de la fracción de muestras acertadas las cuales aparecen en la siguiente tabla.

Tabla 6-1 Medida de aciertos

Palabra	Muestras acertadas	Fracción de medidas acertadas
Conjunción	1297	0.6258
Determinante	6705	0.2667
Pronombre	136552	0.5470
Adverbio	23978	0.3863
Preposición	1370	0.3827
Artículo	63335	0.4616
Verbo	78068	0.4390
Dato	58508	0.4229
Número	204	0.3653

La siguiente tabla muestra los resultados al variar los valores del factor de aprendizaje, el número de capas ocultas y las iteraciones.

Tabla 6-2 Progreso de los resultados

Número de iteraciones	Capas intermedias	Factor de aprendizaje	%Fracción de muestras acertadas

250	3	0.25	46.38
150	3	0.25	46.30
150	2	0.5	46.61
100	3	0.25	42.80
100	2	0.25	46.37
100	2	0.5	46.01
100	2	0.75	46.10
100	2	0.9	45.78
100	1	0.9	46.28
100	1	0.5	46.16

6.3 Resultados obtenidos con otros etiquetadores

6.3.1 Resultados obtenidos con el etiquetador TnT

TnT (Trigrams 'n' Tags) es un etiquetador estadístico de clases gramaticales. Se entrena en diferentes lenguajes y con cualquier conjunto de etiquetas.

El lexicón es creado durante la generación de parámetros en el programa *tnt-para*. Este contiene la frecuencia de las palabras y sus etiquetas, que ocurren en el corpus de entrenamiento. Estas frecuencias son usadas durante la etiquetación para determinar probabilidades léxicas. La primera columna de este archivo es el token, el número que aparece en la segunda columna es la frecuencia de este token, en el corpus de entrenamiento. En el resto de las columnas se encuentran las etiquetas que ocurren con los tokens (columnas impares), unidas con sus frecuencias (columnas pares).

%% Comentario

%%

NNP 62028

NNP CC 2885

NNP CC CD 28

NNP CC NN 51

.....

El archivo anterior contiene la extension .lex.

El archivo *n-gram* es creado durante la generación de parámetros. Este archivo contiene las frecuencias contextuales para *uni-, bi-* y *tri-*grams. Una línea de este archivo contiene dos (unigram), tres (bigram) o cuatro (trigram) columnas. La última columna contiene la frecuencia del *n-gram* particular en el corpus de entrenamiento. Este archivo contiene la extensión .123. El programa TnT consiste de dos pasos:

- Generación de parámetros
- Etiquetación

6.3.1.1 Generación de parámetros

La generación de parámetros requiere un corpus de entrenamiento etiquetado correctamente. El corpus de entrenamiento debe ser grande y también debe asignar la etiqueta correcta.

Nuestro corpus *cpsh.tt* estaba codificado de la siguiente manera, en la primera columna se encontraban las etiquetas y en la segunda columnas se encontraban las etiquetas correctas, cada frase era separada por una (,). Este corpus contenía 2,977,931 frases etiquetadas correctamente.

%% Corpus con las etiquetas correctas

V V

T T

N N

, ,

R R

, ,

V V

V V

N N

S S

N N

Al introducir este corpus en nuestro generador de parámetros obtuvimos los siguientes archivos *cpsh.lex* y *cpsh.123.*

Con esto, se crean los modelos de los parámetros.

6.3.1.2 Etiquetación

El proceso de etiquetación requiere los archivos con los modelos de los parámetros léxicos y frecuencias contextuales (*cpsh.123* y *cpsh.lex*). El etiquetador usa solamente la primera columna en el archivo de entrada.

La manera de etiquetar es la siguiente:

./tnt *[opción] modelo corpus*

Donde *modelo* es el modelo que vamos a utilizar. Se debe encontrar en el directorio los archivos *modelo.123* y *modelo.lex.* En nuestro caso ocupamos el modelo *cpsh.*

Corpus es el archivo de texto que será etiquetado. Para este trabajo constaba solamente de una columna, formada solamente por etiquetas, que podrían ser alguna de el siguiente conjunto {*A,C,D,I,M,N,P,R,S,T,V,W,X,Y,Z*}.

Para nuestro trabajo ocupamos la opción *-u2* la cual combina estadísticas de todas las etiquetas, para el manejo de etiquetas desconocidas. A la etiqueta desconocida que se le iba asignar una etiqueta le asignamos la letra *B.*

Tabla 6-3 Configuración de las frases a etiquetar

Frases sin alguna etiqueta	***Configuración de las frases para TnT***	***Frases con etiquetas correctas***
_T	*BT*	*VT*
_TN	*BTN*	*VTN*
_N	*BN*	*TN*
*V_*	*VB*	*VT*
V_N	*VBN*	*VTN*
*T_*	*TB*	*TN*
*VT_*	*VTB*	*VTN*
_V	*BV*	*YV*
_VN	*BVN*	*YVN*
_N	*BN*	*VN*
*Y_*	*YB*	*YV*
_NS	*BNS*	*VNS*

El siguiente paso es asignar una etiqueta a las etiquetas desconocidas en este archivo. Para realizar esto necesitamos el modelo *cpsh.*

A partir de este archivo y conociendo las verdaderas etiquetas, contamos los valores acertados y los valores falsos:

Tabla 6-4 Resultados de la etiquetación

Frases con etiquetas correctas	***Resultado de etiquetación de TnT***	***Valores acertados***
VT	ST	0
VTN	STN	0
TN	TN	1
VT	VN	0
VTN	VTN	1
TN	TN	1
VTN	VTN	1
YV	PV	0
YVN	PVN	0
VN	TN	0
YV	YN	0
VNS	TNS	0

De esta manera creamos un archivo que contenía los resultados de la etiquetación a las etiquetas desconocidas. El nombre de este archivo es *salet3a.tts* , el cual contenía 171,735 frases etiquetadas correctamente equivalente a un 6% del total del corpus y 2,806,196 frases etiquetadas erróneamente lo que equivale 94%.

6.3.2 Resultados obtenidos con el método de etiquetación Brill

Para el etiquetador de Brill usamos el corpus *sal_cp_bt.txt* el cual fue obtenido a partir del corpus *sal_cps*. Para crear el corpus *sal_cp_bt.txt* fue necesario adaptar la estructura del corpus *sal_cps* para poder establecer la configuración de entrada, indispensable para la ejecución del programa etiquetador de Brill. La configuración necesaria se realizó de la siguiente manera:

-->//VTN

/ //T

/ //T //N

En esta configuración la primera línea contiene las etiquetas correctas para las etiquetas en la frase. En la siguiente línea asignamos el carácter '/'a la etiqueta desconocida, a la cual el etiquetador asignará una etiqueta, un doble carácter '//' le indica al etiquetador que la etiqueta que sigue después de estos caracteres, es una etiqueta correcta ya asignada previamente y no necesita una signación. De esta manera configuramos 1,712,961 frases.

No usamos los archivos Lexicon y Bigrams ya que solamente teníamos la información referente a etiquetas y no a palabras, en el corpus que deseamos etiquetar. Con las frases y un archivo de reglas contextuales para las etiquetas en español pudimos ejecutar el etiquetador de Brill.

Para saber si las etiquetas fueron asignadas correctamente, comparamos los resultados obtenidos por línea y buscamos esta cadena en la cadena con las etiquetas correctas la cual contenía el símbolo "-->". Si encontramos esta cadena como subcadena de la cadena de etiquetas correctas entonces este era un resultado correcto si no este era un resultado erróneo.

Por ejemplo:

-->/VTN (Etiquetas correctas para alguna frase)

//R /T (Resultado erróneo de la etiquetación)

//R /T /N (Resultado erróneo de la etiquetación)

//T /N (Resultado correcto de la etiquetación)

En el archivo *output.txt* a cada etiqueta desconocida se le asignaba un valor de acuerdo a las reglas creadas por el etiquetador. Obtuvimos los siguientes resultados: de las

1,712,961 frases solo 265,392 fueron etiquetadas correctamente equivalente al 15.5% y 14,475,69 frases fueron etiquetadas erróneas equivalente al 84.5%.

6.4 Resolución automática de la homonimia morfológica

Para nuestro problema de resolución de homonimia morfológica, del corpus *homonimia1.tgd* encontramos un total de 1,830,349 frases que se codificaron en las distintas redes dependiendo de su contexto.

Al tener como entrada una frase a cualquiera de las redes, se obtenía como salida un vector 15-dimensional, el cual era el resultado de la ejecución de esa red, con esa entrada. El valor más grande en ese vector, es el valor correspondiente a nuestra etiqueta. Si la frase se acoplaba a más de una red, entonces el vector del resultado total es igual al promedio de todos los vectores de salida, de las redes que se acoplaron a esta frase.

Si el valor de esta etiqueta, que fue el resultado de las redes o la red según fue el caso, es igual al valor de la etiqueta almacenada como correcta, entonces esto es un acierto y de otra manera se le da un valor erróneo.

Con esta metodología obtuvimos un total de 1,410,817 aciertos que corresponde a un 77.07% y en lo que corresponde a los valores falsos obtuvimos un total de 419,532 esto es equivalente a un 22.92%, del total de las frases que son 1,830,349. Esto se puede ver en la siguiente tabla:

Tabla 6-5 Resultados de las redes sin asignación de pesos

Entradas	Aciertos	Falsos	Total	% Aciertos	% Falsos
Sin pesos en las redes	1410817	419532	1830349	77.07	22.93

Al tener en cada una de las redes diferentes contextos, estos nos pueden proporcionar información adicional, para acotar los valores de los aciertos. Así que realizamos una asignación de pesos, equivalentes a las probabilidades de los contextos que intervienen en las etiquetas de las palabras en las frases. Con lo que a cada red dependiendo de sus contextos se le asigno diferente peso. Para los pesos en los contextos tomamos como referencia la posición de la etiqueta variable.

Este peso era multiplicado por el vector de salida de la red y el valor mayor de este vector era el valor de nuestra etiqueta, el cual era comparado con la etiqueta correcta. Si

este valor era el mismo entonces se contaba como valor de certeza, de otra manera era un valor erróneo.

Mediante la asignación de pesos a las distintas redes, obtuvimos distintos resultados. Asignamos los siguientes pesos a las distintas redes:

Tabla 6-6 Redes y el valor de sus pesos

Red	Peso
0	0.3
1	0.3
2	0.5
3	0.5
4	0.5
5	0.8
6	0.8
7	1

Con estos pesos obtuvimos una mejoría de 0.36%, al obtener 1,416,032 resultados correctos equivalente al 77.36% y un total de 414,317 resultados erróneos equivalentes al 22.64%.

Otra parte de nuestro trabajo fue la de asignación de pesos, a las distintas redes, ya que dependiendo de sus contextos podemos obtener más información que nos ayude a mejorar el rendimiento del sistema. Así que asignamos pesos a las redes de la siguiente manera:

- Cambiamos los valores de los pesos simétricos.
- Mayor peso a las redes con más contextos a la izquierda.
- Mayor peso a las redes con más contextos a la derecha.

Para la asignación de pesos simétricos en la red, asignamos los mismos pesos a las redes que tenían el igual número de contextos sin importar si estos se encontraban a la derecha o a la izquierda, los valores menores de estos pesos se encontraban dentro del intervalo abierto [0.2, 0.3], los valores medios para las redes se encontraban en el intervalo abierto [0.5, 0.8], los valores de los pesos mas altos se encontraban en el intervalo abierto [0.9, 1].

Para las configuraciones de la red con asignación de pesos de manera simétrica obtuvimos los resultados que se muestran en la siguiente tabla:

Tabla 6-7 Resultados de las redes con asignación de pesos simétricos

R0	R1	R2	R3	R4	R5	R6	R7	Aciertos	Falsos	Total	% Aciertos	% Falsos
0.3	0.3	0.5	0.5	0.5	0.8	0.8	1	1416032	414317	1830349	77.36	22.64
0.2	0.2	0.3	0.3	0.3	0.5	0.5	0.8	1407364	422985	1830349	76.89	23.11
0.5	0.5	0.7	0.7	0.7	0.8	0.8	0.9	1415603	414746	1830349	77.34	22.66
1	1	0.8	0.8	0.8	0.5	0.5	0.4	1402494	427855	1830349	76.62	23.38
1	1	0.8	0.5	0.5	0.8	1	1	1402437	427912	1830349	76.62	23.38

En la tabla anterior vemos que los valores de los pesos asignados se encuentran en las columnas de 1-7.

La asignación de los pesos, con un valor mayor a los contextos a la izquierda de la etiqueta variable, fueron asignados a las redes R1, R4, R6 y R7, las cuales tienen más de un contexto a la izquierda a partir de la posición de la etiqueta variable. Los valores de estos pesos altos se encontraban en el intervalo abierto [0.8, 1], los valores de los pesos medios se encontraban en el intervalo abierto [0.5, 0.6] y los valores mas bajos se tomaron del intervalo abierto [0.2, 0.3].

Para las configuraciones de la red con asignación de mayor peso a los contextos de la izquierda obtuvimos los siguientes resultados:

Tabla 6-8 Resultados de las redes con asignación de mayor peso a los contextos a la izquierda

R0	R1	R2	R3	R4	R5	R6	R7	Aciertos	Falsos	Total	% Aciertos	% Falsos
0.2	1	0.2	0.5	1	0.5	1	1	1385782	444567	1830349	75.71	24.29
0.5	1	0.5	0.8	1	0.8	1	0.8	1407105	423244	1830349	76.87	23.13
0.5	1	0.3	0.5	1	0.2	1	0.5	1399327	431022	1830349	76.45	23.56
0.5	1	0.3	0.5	1	0.2	0.8	0.5	1398009	432340	1830349	76.37	23.63
0.5	1	0.5	0.5	1	0.5	0.5	0.5	1400818	429531	1830349	76.53	23.47
0.2	1	0.2	0.5	1	0.2	0.5	0.5	1385489	444860	1830349	75.69	24.31

De la misma manera en que tomamos los valores de los pesos a los contextos a la izquierda, fueron asignados a los contextos a la derecha. La asignación de los pesos, con un valor mayor a los contextos a la derecha de la etiqueta variable, fueron asignados a las redes R0, R2, R5 y R7, las cuales tienen más de un contexto a la derecha a partir de la posición de la etiqueta variable. Los valores de estos pesos altos se encontraban en el intervalo abierto [0.8, 1], los valores de los pesos medios se encontraban en el intervalo abierto [0.5, 0.7] y los valores mas bajos se tomaron del intervalo abierto [0.2, 0.3].

Para las configuraciones de la red con asignación de mayor peso a los contextos de la izquierda obtuvimos los siguientes resultados:

Tabla 6-9 Resultados de las redes con asignación de mayor peso a los contextos a la derecha

R0	R1	R2	R3	R4	R5	R6	R7	Aciertos	Falsos	Total	% Acier-tos	% Falsos
1	0.2	1	0.5	0.2	0.8	0.5	0.5	1401041	429308	1830349	76.54	23.46
1	0.2	1	0.5	0.2	1	0.5	1	1397840	432509	1830349	76.37	23.63
1	0.2	1	0.8	0.2	0.8	0.5	0.8	1404681	425668	1830349	76.74	23.26
1	0.1	1	0.6	0.1	0.8	0.6	0.8	1395258	435091	1830349	76.22	23.78
0.8	0.1	0.8	0.6	0.1	0.6	0.6	0.6	1403239	427110	1830349	76.66	23.34
1	0.2	1	0.3	0.2	0.5	0.2	0.5	1385108	445241	1830349	75.67	24.38
0.4	0.4	0.6	0.6	0.6	1	1	1	1417690	412659	1830349	77.45	22.55

7 Conclusiones

Este trabajo de investigación permitió desarrollar un método de resolución en la etiquetación de clases gramaticales como una aplicación de redes neuronales, este método trabaja con un buen nivel de precisión, cuando existe información completa en los contextos anteriores y posteriores de una determinada frase.

Aplicamos un etiquetador de clases gramaticales para el español y probamos una mejora con un costo computacional pequeño.

La tarea involucraba la clasificación de las salidas de cada arquitectura de la red, en alguna de las 15 categorías posibles.

La solución al problema mas general usando retropropagación es una aproximación a la función muchas entradas, muchas salidas, donde cada nodo de salida en la red corresponde a una dimensión de salida del vector de salida y cada nodo de entrada en la red corresponde a una dimensión del vector de entrada. El número de capas y nodos ocultos son dependientes del problema.

El tiempo requerido para el entrenamiento es proporcional al producto del número de iteraciones, el número de pesos usados en la red y el número de muestras. Entonces los requerimientos computacionales para el entrenamiento pueden ser constantemente largos para algunas redes de tamaño razonable.

En problemas multiclase como el de esta investigación, cada nodo de salida corresponde a una clase separada y la clase a la cual el patrón es asignado debe ser indicada por el nodo con la salida más alta, para ese patrón.

Se obtuvieron buenos resultados en la evaluación de los experimentos, al separar palabras que tenían más de una etiqueta, con lo que resolvimos de alguna manera el problema de ambigüedad en la etiquetación. Para resolver este problema y obtener una etiqueta correcta, se verificaron los vectores de salida en cada una de las redes más los pesos que contenían cierta información, dependiendo de los contextos y la etiqueta que obteníamos como salida, era aquella localidad con el valor máximo en nuestro vector de salida.

Una vez implementado el sistema es posible experimentar con diversas estrategias de etiquetación esto con el fin de sacar el mayor provecho de él.

En este trabajo se describe una aportación para la etiquetación de clases gramaticales sin usar información léxica.

8 Trabajo Futuro

Como parte de nuestro trabajo futuro podemos tratar de adaptar nuestro sistema de etiquetación con la metodología de redes neuronales a otros lenguajes como el Inglés, el Francés, el Alemán, el Italiano y el Portugués. Para poder comparar los resultados de nuestro etiquetador en otros lenguajes. Para esto necesitaríamos un corpus etiquetado de estos lenguajes, existen varios corpus en inglés, siendo los más destacados por su utilización en algunos etiquetadores de clases gramaticales, el corpus Brown y el corpus del Wall Street Journal (WSJ).

En caso de que estos lenguajes contengan un número mayor o menor, de etiquetas de clases gramaticales, que las etiquetas que utiliza nuestro sistema, entonces tendremos que variar los vectores de entrada y de salida, sin variar la arquitectura de nuestras redes neuronales. Se seguiría respetando el orden de los contextos en las frases.

Debemos tener en cuenta que la relevancia de las características obtenidas por un etiquetador de clases gramaticales varía a través de los lenguajes. Por ejemplo lenguajes con una morfología muy rica podrían obtener una mayor dependencia en la información de sufijo o prefijo.

También deseamos implementar otros sistemas de clasificación para el problema de etiquetación de clases gramaticales. Una metodología propuesta es la de redes bayesianas las cuales se han probado para etiquetar clases gramaticales en el idioma inglés, con un error de 21.4 % en el corpus Brown [16], podríamos implementar esta metodología para nuestro corpus y comparar los resultados con nuestro sistema de redes neuronales.

Otra metodología con buenos resultados para problemas de reconocimiento de patrones es el uso de memorias asociativas, podemos implementar esta metodología para la etiquetación de clases gramaticales. Tomando como patrones diferentes las distintas clases gramaticales y tendríamos que reconocer los patrones dados en los contextos, para obtener una etiqueta. El siguiente paso sería comparar los resultados de la metodología de memorias asociativas, con los obtenidos en este trabajo de investigación.

9 Referencias

[1] Elements of Artificial Neural Networks, Kishan Mehrotra, Chilukuri K. Mohan, Sanjay Ranka. Page 4, The MIT Press 1997.

[2] Redes Neuronales James A: Freeman y David M. Skapura, Addison-Wesley Iberoamericana 1992.

[3] Neural Smithing, Russell D. Reed and Robert J. Marks. The MIT Press 1999.

[4] Redes Neuronales Artificiales, José R. Hilera, Víctor J. Martínez, Addison-Wesley Iberoamericana 1995.

[5] Tagging Experiments Using Neural Networks, Martin Eineborg and Björn Gambäck, 1994.

[6] Neural Networks, B. Müller and J. Reinhardt Springer-Verlag 1990.

[8] Some Advances in Transformation-Based Part of Speech Tagging, Eric Brill, 1994.

[9] A syntax-based part of speech analyzer, Atro Voutilainen, 1995.

[10] Revision Learning and its Application to Part of Speech tagging, Tetsuji Nakagawa and Taku Kudo and Yuji Matsumoto.

[11] Moreno, A. and J. Goñi. GRAMPAL: A Morphological Processor for Spanish Implemented in PROLOG. En: Mar Sessa y María Alpuente, editores, Proceedings of the Joint Conference on Declarative Programming (GULP-PRODE'95), pp. 321-331, Marina di Vietri (Italia), 1995.

[12] AGME: Un Sistema de Análisis y Generación de la Morfología del Español. Francisco Velásquez, Alexander Gelbukh, Grigori Sidorov. Centro de Investigación en Computación (CIC).

[13] The Handbook of Linguistics. Mark Aronoff and Janie rees-Miller, Blackwell publishers. 2001.

[14] Trend in Linguistics. Aspects of the Theory of Morphology. Igor Mel'cuk. Mouton de Gruyter 2006.

[15] Teoría y práctica de la lingüística moderna. Manuel Ortuño Martínez. Trillas 1980. Pág. 137.

[16] Part-Of–Speech Tagging with neural networks. Hehnut Schmid.

[17] Hybrid Neuro and Rule-Based Part of Speech Taggers. Qing Ma, Masaki Murata, Kiyotaka Uchimoto, Hitoshi Isahara.

[18] Distributional Part-of-Speech Tagging. Hinrich Schütze.

[19] Learning Part-of-Speech Guessing Rules from Lexicon: Extension to Non-Concatenative Operations. Andrei Mikheev.

[20] Incorporating POS Tagging into language modeling. Peter A. Heeman, James F. Allen.

[21] Primitive Part-Of-Speech Tagging using Word Length and Sentential Structure. Simon Cozens, Pusey Lane.

[22] TnT A Statistical Part-Of-Speech Tagger. Thorsten Brants.

[23] Inter-Annotator Agreement for a German Newspaper Corpus. Thorsten Brants.

[24] Unsupervised Learning of Disambiguation Rules for Part of Speech Tagging. Eric Brill.

[25] Word Sense Disambiguation with Very Large Neural Networks. Extracted from Machine Readable Dictionaries. Jean Veronis and Nancy M.

[26] Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data John Lafferty, Andrew McCallum and Fernando Pereira.

[27] Neural Networks for Wordform Recognition. Martin Eineborg, Björn Gambäck, 1994.

[28] Homonymy and Polysemy in Information Retrieval. Robert Krovetz.

[29] Memory-Based Morphological Analysis. Antal van den Bosch and Walter Daelemans.

[30] Particle Homonymy and Machine Translation. Károly Fäbricz.

[31] Neural Networks in APL. By Manuel Alfonseca.

[32] A Neural Network Model Of Serial Learning. Hilary A. Broadbent, Jim Lucas.

[33] Distributional Part-of-Speech Tagging. Hinrich Schfitze.

[34] Part-of-Speech Tagging with Neural Networks. Hehnut Schmid.

[35] Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. Kristina Toutanova Dan Klein.

[36] Neural network approach to word category prediction for english texts. Masami Nakamura, Katsuteru Maruyama, Takeshi Kawabata , Kiyohiro Shikano.

[37] Comparing a Linguistic and a Stochastic Tagger. Christer Samuelsson Atro Voutilainen.

[38] Revision Learning and its Application to Part-of-Speech Tagging. Tetsuji Nakagawa and Taku Kudo and Yuji Matsumoto.

[40] Hybrid Neuro and Rule-Based Part of Speech Taggers. Qing Ma, Masaki Murata, Kiyotaka Uchimoto, Hitoshi Isahara.

[41] A Multi-Neuro Tagger Using Variable Lengths of Contexts. Qing Ma and Hitoshi Isahara.

[42] Part of Speech Tagging Using a Network of Linear Separators. Dan Roth and Dmitry Zelenko.

[43] Neural Network Models of Categorical Perception. R.I. Damper and S.R. Harnad.

[44] Part-of-Speech Tagging with Minimal Lexicalization. Virginia Savova & Leonid Peshkin.

[45] Tagging accurately- Don't guess if you know. Pasi Tapanainen.

[46] Morphosyntactic Analysis and Parsing of Unrestricted Spanish Text, J. Atserias, J. Carmona.

10 Apéndices

Apéndice A. Implementación de una red de retropropagación

Propagación hacia delante

```
void propagacion_adelante( double *patrón_entrada)
{    /* Copiar los patrones de entrada a los nodos de entrada */
    for ( i = 0; i < numero_de_entradas; i++ )
        nodo_salida[i+1] = patrón[i];
    nodo_salida[0] = 1;
    /* Calculamos la salida de los nodos restantes */
    for ( i = primer_indice_oculto; i < numero_de_nodos; i++ )
    {
        double sum = 0;
        for ( j = 0; j < i; j++ )
            sum += peso[i][j]*nodo_salida[j];
        nodo_salida[i] = sigma (sum);
    }
}
```

Propagación del error hacia atrás

```
void retroprop_deltas( double *patrón_objetivo)
{
    /* Calcular el delta de los nodos para los nodos de salida */
    for ( i = ultimo_nodo_salida; i >= primer_nodo_salida; i-- )
    {
          double err = objetivo[i] – valor_nodo[i];
          delta[i] = err * valor_nodo[i] * ( 1-valor_nodo[i] );
                              /* periodo de pendiente sigmoidal */
    }
    /* calcular las deltas para nodos ocultos, trabajar en reversa */
    for ( i = ultimo_nodo_oculto; i >= primer_nodo_oculto; i-- )
    {
        delta[i] = 0;
        for ( k = i+1; k <= ultimo_nodo_salida; k++ )
              delta[i] += peso[k][i] * delta[k];
        delta[i] *= valor_nodo[i] * (1 – valor_nodo[i] );
                              /* periodo de pendiente sigmoidal */
    }
}
```

Apéndice B. Formalización de la red neuronal de retropropagación

La formalización de la arquitectura de la red neuronal de retropropagación es la siguiente:

Comenzamos con la formalización de la arquitectura de la red neuronal de retropropagación (NN), que puede ser definida en la siguiente 3-túpla:

Red de retropropagación $NN(S,P,T)$

donde S es el conjunto de patrones, usados para entrenamiento, prueba u operación de la red, P es el conjunto de parámetros de la red y T es la topología de la red.

Conjunto de patrones $S = \{ I, O \}$

donde I es un conjunto de patrones de entrada y O es el conjunto de patrones de salida deseadas.

Conjunto de entrada $I = \{p_{k,j}\}$

donde k es el número de patrón de entrada y j es el componente del patrón de entrada

Conjunto de salida $O = \{o_{k,j}\}$

donde k es el número del patrón de salida deseada y j es el componente del patrón de salida.

El conjunto de parámetros para red de retropropagación contiene dos valores: el término de momento (α) y la razón de aprendizaje (β).

Conjunto de parámetros $P = \{ \alpha , \beta \}$

Topología $T = (F, L)$

La topología de la red define la estructura de la red F y las interconexiones entre los nodos de la red L.

Estructura $F = \{c_0, c_1, \ldots . c_{LN-1}\}$

donde LN es el número de capas, c_0 representa la capa de entrada, c_{LN-1} representa la capa de salida y c_1 hasta c_{LN-2} representan las capas intermedias

Cluster (capa) $c_i = \{n_{i,j}\}$

Capa de entrada $c_0 = \{n_{0,i}\}$ *donde* $0 \le i \le \| p \|$

Nodos de entrada $n_{1,i}(t+1)=I_{(t \bmod \|I\|),i}$ *donde* $0 \le i \le \|p\|$

Nodos de las capas intermedias y de salida

$$n_{i,j}(t+1)=f\left(\sum_{k=0}^{k<\|c_{i-1}\|} n_{i-1,k}(t+1)\cdot w_{i-1,k\to i,j}(t)\right)$$

Donde $1 \le i < LN$

y $0 \le j < \|c_i\|$

Función de transferencia $f(x)=\dfrac{1.0}{1.0+e^{-x}}$

Valores de enlace iniciales

$w_{i,j\to i+1,l}(0)=((Random(200000) - 100000) / 100000)$

Donde $1 \le i < LN-1$

y $0 \le j < \|c_i\|$

y $0 \le l < \|c_{i+1}\|$

Todos los valores de los siguientes enlaces son calculadas al aumentar un cambio (Δ) al valor del enlace previo con un porcentaje del cambio previo (α)

Valores de enlaces

$w_{i,j\to i+1,l}(t+1)=w_{i,j\to i+1,l}(t)+\Delta_{i,j\to i+1,l}(t+1)+\alpha\Delta_{i,j\to i+1,l}(t)$

Donde $1 \le i < LN-1$

y $0 \le j < \|c_i\|$

y $0 \le l < \|c_{i+1}\|$

Δ **(Inicial)** $\Delta_{i,j\to i+1,l}(0)=0$

Donde $1 \le i < LN-1$

y $0 \le j < \|c_i\|$

y $0 \le l < \|c_{i+1}\|$

Δ $\Delta_{i,j\to i+1,l}(t+1)=\beta E_{i+1,l}(t+1)n_{i,j}(t+1)$

Donde $1 \le i < LN-1$

y $0 \le j < \|c_i\|$

y $0 \le l < \|c_{i+1}\|$

Error en la capa de salida $E_{LN-1,j}(t+1) = f'(n_{LN-1,j}(t+1))(o_{(t \bmod i),j} - n_{LN-1,j}(t+1))$

$f'(x)=x(1-x)$

Donde $0 \leq j < \|c_{LN-1}\|$

Error en capas intermedias

$$E_{i,j}(t+1) = f'(n_{i,j}(t+1)) \sum_{l=0}^{l<\|c_{i+1}\|} (w_{i,j \to i+1,l}(t) \cdot E_{i+1,l}(t+1))$$

$$f'(x) = x(1-x)$$

Donde $0 \leq j < \|c_i\|$

y $1 \leq i < LN-1$

Apéndice C. Clases gramaticales de palabras en español

Sustantivos

El sustantivo es una palabra que designa substancias; seres que pueden ser sujetos u objetos de una acción, un estado o cualquier accidente expresable con un verbo.

Esquema de clasificación del sustantivo

Concretos; Son los que designan seres u objetos que tienen una existencia real; dentro de los concretos cabe distinguir:

Comunes; aquellos que convienen a todos los individuos de una misma clase. Entre los que hay:

- Genéricos; los que poseen cierto número de cualidades comunes, que los distinguen de los demás. Entre los que hay:
    - Individuales; referidos a un individuo cualquiera de una especie
    - Colectivos; referidos a un grupo o conjunto de individuos de la misma especie
- De materia; los que no indican un objeto determinado, sino la sustancia, sin forma ni extensión, que posee las cualidades significadas por el nombre.

Propios; los que designan un individuo o ser determinado en el grupo genérico al cual pertenecen.

Abstractos; son los que indican cualidades permanentes (sustantivos de cualidad) o fenómenos, es decir; cualidades transitorias o accidentales (sustantivos de fenómeno) que se abstraen o separan mentalmente de los seres u objetos donde los observamos. Dentro de este grupo pueden figurar también los sustantivos numerales, ya múltiplos, partitivos o colectivos y los quebrados.

El número del sustantivo es la diferente forma que adopta el sustantivo para expresar si se refiere a un objeto (singular) o más de uno (plural).

Género del sustantivo

En español sólo hay, dos géneros: el masculino y el femenino. En los sustantivos no existe el género neutro; este género se presenta, sin embargo, en la sustantivación de los adjetivos y en ciertas formas pronominales, muy características.

Los demás sustantivos que no designan seres animados poseen gramaticalmente alguno de los dos géneros. En nuestra lengua el género es fundamentalmente etimológico.

En algunos lenguajes, el sustantivo aparece en diferentes formas cuando tienen diferentes funciones en una oración, estas formas son llamadas casos, algunos de estos son: nominativo, genitivo, acusativo y dativo.

Pronombres

Los pronombres son pequeñas clases de palabras que actúan como variables, estas se refieren a personas o cosas que de alguna manera sobresalen en el contexto del discurso.

Clasificación

El *pronombre* puede actuar como sustantivo o ejercer funciones de sustantivo, aunque también presenta formas adjetivas (los *posesivos*). Empero, hay casos en que los sustantivos no se pueden conmutar por pronombre: la aposición.

Los *pronombres personales* son las palabras *yo, tú, él, ella, ello; nosotros(as) vosotros(as), ellos(as), me, a mí, te, a ti, le, la, lo; les, las, los*. Las formas *yo, tú, me, a mí, te, a ti* representan a la persona que habla o a aquella a quien se habla. Las formas *él, ella, le, la, lo; les, las, los* representan a las personas o cosas de que se habla. La forma ello representa a una oración completa o un conjunto de antecedentes ya mencionados.

Los *pronombres demostrativos* **s**irven para mostrar o señalar la cosa designada por el nombre. Son: é*ste(a), esto(s), ésta(s), ése(a), eso(s), ésa(s), aquel, aquélla, aquello; aquéllos, aquéllas*. Todos ellos, en las formas masculinas y femeninas, se pueden usar como adjetivos o como pronombres.

Los *pronombres posesivos* denotan posesión como: *de* é*l, lo mío(a), tuyo(a), suyo (a), nuestro (a), suyo(a), los/las míos(as), tuyos(as), suyos(as), nuestros(as), suyos (as).*

Los *pronombres relativos* representan en una oración subordinada un elemento de la principal que queda especificado o calificado en aquélla. El relativo puede referirse o hacer relación a un nombre contenido en la oración principal, o puede consistir la relación entre las dos oraciones. Son pronombres relativos *cual, cuyo, que, quien, el cual, el que.*

Los *pronombres indefinidos* señalan con imprecisión la naturaleza, el número, la cantidad o el grado de los sustantivos a los que acompañan o aluden. Son: *alguien, nadie, cualquiera, quienquiera, otro, algo, nada.*

Los *pronombres numerales* **e**xpresan de modo preciso y exacto la cantidad de objetos designados por el nombre al que acompañan y delimitan o al que designan. Se distinguen varios tipos: cardinales, ordinales, fraccionarios, multiplicativos, distributivos y colectivos.

Los *pronombres interrogativos y exclamativos* se utilizan en frases exclamativas e interrogativas, en preguntas directas o indirectas. Son: *qué, quién, cuál, cuánto*.

Determinantes

Los *determinantes* describen la referencia particular de un sustantivo. Un subtipo de determinante es el artículo. El artículo es el nombre dado a las palabras e*l, la, lo, los, las; un, una, unos, unas, que,* sin tener por sí solas ninguna significación, acompañan al nombre participando de sus accidentes de género y número.

Clasificación de los determinantes

El artículo *determinado* acompaña a un nombre cuando éste designa una cosa única, abstracta o concreta, una cosa determinada o todas las de una especie. *El, la, lo, los, las.*

El artículo *indeterminado* acompaña a un nombre que se refiere a una o varias cosas indeterminadas numerables, de las que existen o pueden existir, además, otras. Se aplica a una cosa determinada para el que habla, si esa cosa es desconocida para la persona a quien se habla, y a la inversa. *Un, una, unos, unas*.

Adjetivos

El *adjetivo* es la palabra que dice algo del sustantivo al cual se aplica para calificarlo o para determinar a cuáles o cuántos de los designados con el mismo sustantivo se refiere el que habla; para determinarlo. Al igual que el sustantivo al que acompaña, el adjetivo tiene género y número; también posee sufijos o prefijos que lo modifican.

Clasificación de los adjetivos

Los *adjetivos demostrativos* delimitan la significación en el espacio o tiempo.

Los *adjetivos posesivos* se refieren al significado del nombre a las relaciones de posesión.

Los *adjetivos numerales* limitan la significación del nombre y la cuantifican u ordenan.

Los *adjetivos indefinidos* presentan al nombre de forma imprecisa: *algún, ningún, todo, cierto o semejante*.

Los *adjetivos calificativos* acompañan al sustantivo, apareciendo antes o después de este, o de forma independiente.

Los *adjetivos cuantificables* pueden presentar su cualidad en diferentes grados de intensidad.

Verbos

Los *verbos* son las clases de palabras que, desde el punto de vista semántico, sirven para expresar fundamentalmente acción, estado, pasión o proceso, aunque en algunos casos también expresan cualidad.

Lexemas y morfemas verbales

Lexema: es la parte de la forma verbal que contiene el significado básico del verbo, es decir, es la parte que nos informa de la acción que ocurre.

Los *morfemas* o *desinencias* del verbo son las terminaciones que se añaden al lexema para construir las distintas formas verbales. A estas terminaciones las llamaremos desinencias verbales. Las desinencias se obtienen al quitar el lexema a una forma verbal.

El tiempo de los verbos

Las formas verbales sitúan la acción en un tiempo determinado.

- El presente señala que la acción coincide con el momento en el que se está hablando.
- El pasado indica que la acción corresponde a un momento anterior al presente.
- El futuro se refiere a una acción situada en un tiempo que aún no ha llegado.

El modo de los verbos

Las formas verbales nos informan de la actitud que tiene el hablante cuando habla. Esta información depende del modo en que esté la forma verbal.

- Empleamos el modo indicativo cuando hablamos de acciones que consideramos reales o seguras.
- Empleamos el modo subjuntivo cuando nos referimos a acciones que consideramos posibles, deseables o dudosas.
- Empleamos el modo imperativo cuando dirigimos órdenes afirmativas al oyente.

Los tiempos verbales

Llamamos tiempos al conjunto de formas verbales que presentan la acción de la misma manera y corresponden a un mismo tiempo (pasado, presente o futuro). Cada tiempo verbal consta de seis formas que varían en número y persona.

Tiempos simples y tiempos compuestos

- Las formas verbales simples constan de una sola palabra.
- Las formas verbales compuestas constan de dos palabras: una forma del verbo haber y el participio del verbo que queremos conjugar.

Tiempos imperfectos y tiempos perfectos

- Tiempos imperfectos son los que presentan la acción sin acabar.
- Tiempos perfectos son los que presentan una acción ya terminada.

Tiempo verbal y tiempo real

Los tiempos verbales sitúan la acción en un tiempo real determinado. Las formas del presente se refieran a acciones actuales, las formas del futuro se refieren a acciones venideras y las formas del pretérito se refieren a acciones pasadas.

Muchas veces, sin embargo, empleamos los tiempos verbales con un valor distinto del tiempo real que les corresponde:

- Presente con valor de pasado. Se llama también presente histórico y se emplea para actualizar acciones ya pasadas.
- Presente con valor de futuro. Se emplea para referirnos a acciones venideras.
- Presente con valor habitual. Se emplea para referirnos a acciones que se repiten antes y después del momento en que hablamos.
- Presente con valor intemporal. Se emplea para referirnos a acciones que ocurren siempre.
- Presente con valor de mandato. Se emplea para dar órdenes.

Clases de verbos

- Verbos auxiliares: son verbos que ayudan a formar otros y añaden cierto significado.
- Verbos defectivos: son verbos que carecen en su conjugación de algunas formas verbales.

- Verbos regulares: son los verbos que mantienen igual el lexema o raíz en todas sus formas y siguen las mismas desinencias que los verbos modelo de la conjugación a la que pertenecen.
- Verbos irregulares: son los que no mantienen el mismo lexema de su infinitivo, no siguen las mismas desinencias de los verbos modelo o ambas cosas a la vez.

Adverbios

El *adverbio* es una palabra invariable que modifica el sentido de un verbo, el sentido de un adjetivo, el sentido de otro adverbio, la posición del que habla.

Clasificación

Adverbios de modo

Adverbios de cantidad (o de intensidad)

Adverbios de tiempo

Adverbios de lugar

Adverbios de afirmación

Adverbios de negación

Adverbios de duda

Preposiciones

La *preposición* es una palabra que no presenta variación y establece una relación entre otras dos palabras, una de las cuales expresa un complemento de la otra. La preposición precede siempre a su término formando una unidad que no puede destruirse sin alterar el sentido.

Clasificación

- *Preposiciones propias* constituidas por una sola palabra. *A, ante, bajo, cabe, con, contra, de, desde, en, entre, hacia, hasta, para, por, según, sin, so, sobre y tras.*
- *Preposiciones impropias*: adjetivos, adverbios y participios que funcionan como preposiciones. *Durante, mediante, vía*
- *Preposiciones inseparables o prefijas*: des-, ex-, extra- .

- *Locuciones prepositivas*: se forman utilizando la preposición de a los adverbios *encima, detrás, abajo...*

Conjunciones

Las *conjunciones*, además de su carácter relacionante, tienen cierto contenido semántico, por lo que pueden agruparse en las clases copulativas, disyuntivas, adversativas y consecutivas.

Tanto las preposiciones como las conjunciones pueden presentarse en la forma de frases o locuciones prepositivas y conjuntivas.

Interjecciones

En cuanto a la *interjección*, no se le considera clase de palabra, ya que por sí misma contiene todas las características sintácticas y semánticas del enunciado completo. Es un enunciado.

Estructura fraseológica

Las palabras son organizadas dentro de las frases. La sintaxis es el estudio de las regularidades y el orden de construcción de las palabras. Dos palabras producen una relación sintagmática si ellas pueden formar una frase.

Frases sustantivas

El sustantivo es el constituyente central que determina el carácter sintáctico de la frase. Las frases sustantivas son usualmente argumentos del verbo. Las frases sustantivas normalmente consisten de un determinante opcional, cero o mas frases adjetivas, un sustantivo principal y quizás algunos modificadores tales como frases preposicionales.

Frases preposicionales

Son encabezadas por una preposición y contienen una frase sustantiva complementaria. Pueden aparecer dentro de los tipos mayores. Son comunes en frases sustantivas y frases verbales donde expresan locaciones espaciales y temporales.

Frases verbales

El verbo es la parte principal. En general la frase verbal organiza todos los elementos de la oración que dependen sintácticamente del verbo.

Frases adjetivas

Frases adjetivas complejas son menos comunes. Su función es modificar el núcleo de una frase sustantiva.

Gramáticas de estructuras fraseológicas

Un análisis sintáctico de una oración, nos permite determinar el significado de una oración desde el significado de las palabras. Algunos lenguajes como Latín, Español o Ruso permiten diferentes maneras de ordenar las palabras en las oraciones sin un cambio en el significado. Estos tipos de lenguaje son llamados *lenguajes de orden libre de palabra.*