

Desarrollo de un Tokenizador Mínimo para la Analítica de Voz en Interacciones Telefónicas en Español

Elías Cristaldo¹

¹ Ingeniería en Informática / Facultad Politécnica / Universidad Nacional de Asunción
Campus, San Lorenzo – Paraguay, Teléfono (+595-21) 588 7000

27 de Junio del 2024

Abstract. Este informe propone una solución para la implementación de los analizadores de voz. Utilizando conocimientos fundamentales del análisis léxico, se construye un tokenizador mínimo que funciona como una herramienta de analítica de voz. Este tokenizador se encargará de identificar los lexemas presentes en un texto que representa la interacción de una llamada telefónica y asignará las etiquetas correspondientes. La solución se desarrolla en tres etapas principales: la primera etapa corresponde al análisis del personal de atención. En esta fase, el texto correspondiente al personal de atención es analizado, y como resultado, se obtiene una puntuación relativa al personal. Como segunda etapa tenemos el análisis del cliente, de vuelta se analiza el texto ingresado, pero en este caso realtiva al cliente, obteniendo así una puntuación que refleja el nivel relativo de satisfacción del cliente respecto a la solución de sus necesidades. Como tercera y última etapa tenemos el análisis global de la llamada, en esta etapa se realiza un análisis combinando las puntuaciones obtenidas en las etapas anteriores, ofreciendo una evaluación integral de la llamada. El programa propuesto demuestra un buen desempeño en el análisis léxico del texto recibido como entrada. Además, puede ser ampliado para mejorar su capacidad de detección de patrones y, al ser combinado con otros algoritmos, alcanzar niveles óptimos de rendimiento.

Keywords: tokenizador mínimo, analítica de voz, análisis léxico, interacción telefónica, satisfacción del cliente, desempeño del personal, token.

1. INTRODUCCIÓN

En el ámbito de la analítica de conversaciones, la capacidad de procesar y analizar grandes volúmenes de interacciones telefónicas se ha vuelto una necesidad crítica para muchas organizaciones. Estas interacciones contienen una gran cantidad de información valiosa que puede ser utilizada para mejorar la calidad del servicio, entender mejor las necesidades de los clientes y optimizar procesos internos.

Para abordar esta necesidad, presentamos el desarrollo de un tokenizador mínimo, conocido como MNLPTK (Minimal Natural Language Processing Tokenizer). Este sistema está diseñado para actuar como una solución de analítica de voz (speech analytics), con el objetivo de identificar y pro-

cesar palabras (lexemas) en textos en idioma español que son el resultado de conversaciones telefónicas.

El **MNLPTK** se enfoca en la identificación precisa de lexemas en los textos de entrada, lo cual es un paso fundamental en el procesamiento del lenguaje natural (NLP). Una vez identificados los lexemas, el sistema los procesará para generar una ponderación sobre la llamada en general. Esta evaluación se basará en varios criterios que, en cierta medida, reflejan la calidad y el contenido de la interacción.

2. OBJETIVO Y ALCANCE

El objetivo de este proyecto es implementar un sistema de **Speech Analytics** que aproveche los conocimientos adquiridos en el análisis léxico de un compilador. Este sistema se diseñará para procesar y analizar datos de audio convertidos a texto, extrayendo información valiosa para la organización.

Aplicando principios de análisis léxico, se busca desarrollar un programa capaz de identificar patrones en las conversaciones y asignar puntuaciones coherentes a las llamadas. Este enfoque permitirá obtener un análisis detallado de la calidad del servicio y la atención al cliente, proporcionando información que pueda ayudar a mejorar continuamente el servicio ofrecido.

En cuanto al alcance del proyecto, se espera que el análisis se realice únicamente a nivel léxico. No se considerarán aspectos sintácticos ni semánticos, ya que el objetivo es desarrollar un tokenizador mínimo.

El aprendizaje también es un área que puede ampliarse en este proyecto. Actualmente, no se espera que el programa identifique variantes de género y número de las palabras de entrada reconocidas. Sin embargo, esta funcionalidad podría incorporarse en futuras iteraciones para mejorar la precisión y utilidad del análisis léxico.

3. METODOLOGÍA

Para la aplicación del **MNLPTK** se establecieron los siguientes puntos.

3.1. TOKENS DEFINIDOS PARA EL LENGUAJE DE ENTRADA

El MNLPTK contará con los siguientes tokens: **EXP_MALA**, **EXP_NEUTRA**, **EXP_BUENA** que irán destinados para el análisis de la interacción del cliente y **ATC_MALA**, **ATC_NEUTRA**, **ATC_BUENA** que corresponderán a la interacción del personal de atención al cliente.

3.2. DEFINICIÓN DE LA HERRAMIENTA A UTILIZAR PARA LA DEFINICIÓN DE LOS PATRONES

Para la representación de los patrones se opta por utilizar el algoritmo de simulación de un **AFD** para obtener los lexemas. En la Figura 1 podemos observar una categorización de los tokens por colores de acuerdo si es una palabra buena, neutra o mala. Esto último para entender a través de un ejemplo, el funcionamiento del algoritmo propuesto.



Figura 1: Tokens.

A continuación, en la Figura 2 se muestra a través de un ejemplo, la estructura que se utilizará en la implementación del AFD. En esta representación, cada nodo que corresponda a un estado final estará asociado una categoría específica. Es importante destacar que habrán dos estructuras similares: una para la atención al cliente, que manejará los tokens ATC_MALA, ATC_NEUTRA y ATC_BUENA. Otra otra para la experiencia del cliente, que incluirá los tokens EXP_MALA, EXP_NEUTRA y EXP_BUENA.

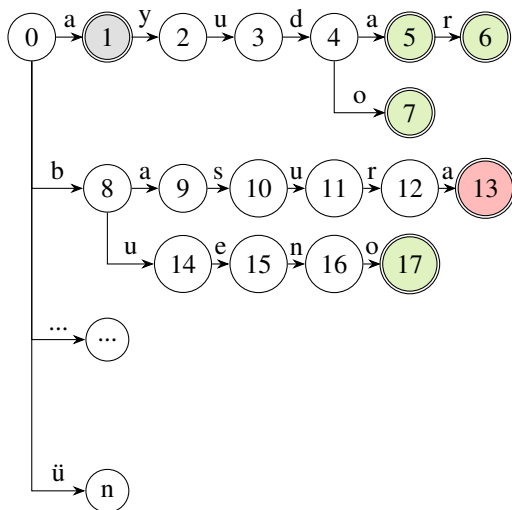


Figura 2: Grafo dirigido con tres nodos.

La idea es que todas las letras del abecedario tengan un único estado inicial. Esto permitirá que al recorrer la es-

tructura, solo se necesite una función, la función **mover**. Esta función tomará un carácter de entrada y, en función del nodo en el que se encuentre, se moverá a un nuevo nodo.

3.3. ESTRUCTURA DE LOS NODOS DEL AFD

Para la construcción del AFD, se definió la estructura **Nodo** junto con sus atributos correspondientes. En la Figura 3 podremos apreciar con más detalles la mencionada estructura.

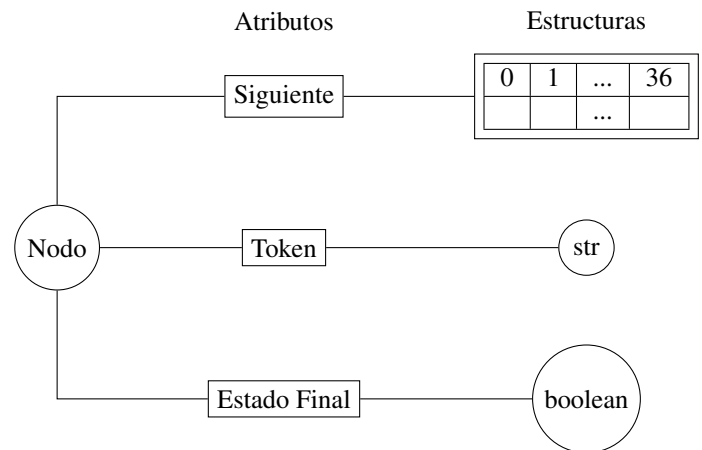


Figura 3: Modelo de un Nodo en el AFD.

El atributo **Siguiente** es un array con 37 elementos, donde cada celda contiene vacío o una referencia a otro nodo. El atributo **Token** es una cadena de texto de tipo string que describe el token al que pertenece el nodo. Por último, el atributo **Estado Final** es un valor booleano que indica si el nodo es un estado final o no. En la siguiente sección, se presentará la función hash que mapea los valores del array **Siguiente**.

3.4. FUNCIÓN HASH

En la sección anterior se presentó la estructura de un nodo, destacando que el atributo **Siguiente** es un array de 37 elementos. ¿Por qué se eligió esta estructura?. La decisión se basa en la necesidad de un acceso más rápido a los datos, donde cada posición del array representa un carácter del alfabeto, como se muestra en la Figura 4.

a	b	...	ñ	...	z	...	á	...	ú	ä	...	ü
0	1	...	14	...	26	...	27	...	31	32	...	36

Figura 4: Función hash para el array de 37 elementos.

Se ha investigado que el diccionario de la RAE contiene aproximadamente 93.000 entradas [1]. Se calculó que cada nodo ocupa aproximadamente 464 bytes. Con una media de 6 letras por palabras. Por lo tanto se tiene que

N : Cantidad aproximada de nodos

C : Capacidad requerida

$$N = 93.000 \text{ [palabras]} * 6 \left[\frac{\text{letras}}{\text{palabras}} \right]$$
$$= 558.000 \text{ [letras]}$$

como cada letra representa un nodo, entonces...

$$C = 558.000 \text{ [nodo]} * 464 \left[\frac{\text{bytes}}{\text{nodo}} \right]$$
$$= 258.912.000 \text{ [bytes]}$$

pasando a MB tenemos que ...

$$258.912.000 \text{ [bytes]} \equiv \frac{258.912.000}{2^{20}}$$
$$\approx 258,912 \text{ [MB]}$$

por lo tanto, la estructura propuesta es viable de implementar, ya que no requiere un espacio considerable. Aunque no se han tenido en cuenta las conjugaciones, hemos obtenido una aproximación que nos servirá como una cota.

3.5. CONSIDERACIONES PARA LA PUNTUACIÓN

Para obtener las evaluaciones, se tuvieron en cuenta las siguientes consideraciones:

– $total$: Cantidad de palabras reconocidas en la entrada.

– $total_{buenas}$: Cantidad de palabras buenas encontradas.

Si se encontraron un saludo y una despedida, cada uno suma un punto en este apartado.

– $total_{malas}$: Cantidad de palabras malas encontradas.

Si no se encontraron un saludo y una despedida, cada uno suma un punto en este apartado.

– $total_{malas}$: Cantidad de palabras malas encontradas.

$$buenas_{norm} = \frac{total_{buenas}}{total}$$
$$malas_{norm} = \frac{total_{malas}}{total}$$
$$balance = buenas_{norm} - malas_{norm},$$

teniendo en cuenta lo anterior la puntuación final estará dada por $1 + p(x)$, siendo $p(x)$ definida por,

$$p(x) = \begin{cases} 0 & \text{si } balance \leq 0, \\ balance * 4 & \text{si } balance > 0 \end{cases}$$

con esto, obtendremos una puntuación final que oscilará entre 1 y 5.

4. CÓDIGO FUENTE

A continuación, se presentan las partes más importantes del código fuente:

```
1 def fibonacci(n):
2     """Return the n-th Fibonacci number."""
3     if n <= 0:
4         return 0
5     elif n == 1:
6         return 1
7     else:
8         return fibonacci(n-1) + fibonacci(n-2)
```

Listing 1: Ejemplo de código en Python

5. CASO DE PRUEBA

A continuación, se detallan los pasos necesarios para procesar el caso de prueba:

■ Entrada – Atención al Cliente

Buenos días, le saluda Joanna de atención al cliente, ¿en qué puedo ayudarle? Puedo darle esa información. Por favor, ¿me facilita su número de documento? Gracias, una pregunta de seguridad, ¿puede darme el año de su fecha de nacimiento? Gracias. Verifico que su saldo es de doscientos treinta mil guaraníes a la fecha de ayer. ¿Algo más que pueda hacer por usted? Gracias por llamar al servicio de atención al cliente. Que tenga un buen día.

■ Entrada – Cliente

Buen día. Desde ayer que no puedo consultar mi saldo. Claro, tres millones doscientos sesenta mil cero sesenta y ocho. En año de mi nacimiento es mil novecientos noventa y seis. No, muchas gracias. Usted ha sido muy amable. Hasta luego.

5.1. PROCESAMIENTO DE LA INTERACCIÓN DEL ATC

A continuación se detallan los pasos a seguir para procesar la interacción del personal de atención al cliente.

5.1.1. INGRESO DE LOS DATOS

Primero, se ingresa el texto en el apartado correspondiente al Personal de Atención al Cliente (ATC), tal como se muestra en la Figura 5.

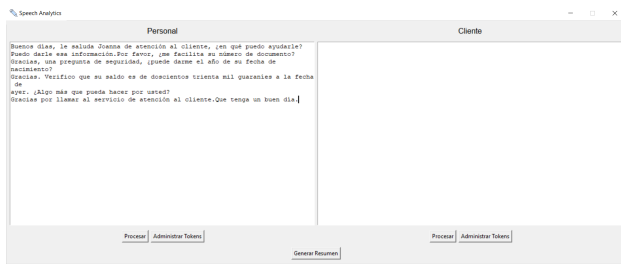


Figura 5: Ingreso de datos del personal

5.1.2. PROCESAMOS EL TEXTO

A continuación, se presiona el botón **Procesar** correspondiente al personal, tal como se muestra en la Figura 6.

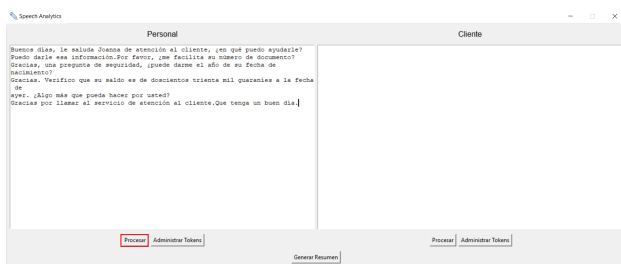


Figura 6: Procesamiento del texto ingresado

5.1.3. CATEGORIZACIÓN DE LOS LEXEMAS

Al principio, es necesario entrenar el programa, ya que aún no tiene cargada ninguna información. Por lo tanto, debemos indicarle palabra por palabra a qué token pertenecen, tal y como se observa en la Figura 7.

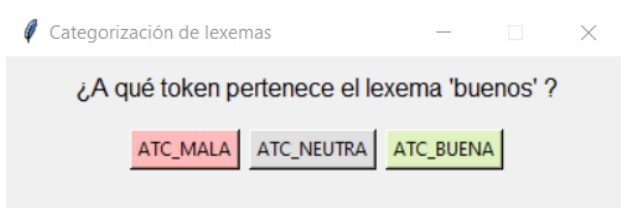


Figura 7: Categorización de los lexemas

5.1.4. RESULTADOS PRELIMINARES

Una vez completado el paso anterior, podremos observar los resultados generados para este análisis, los cuales se ajustarán a los criterios establecidos previamente. En la Figura 8, podremos observar esto. Estos resultados nos proporcionan información sobre el desempeño del personal de ATC y nos permiten identificar si es necesario realizar mejoras en su capacitación.

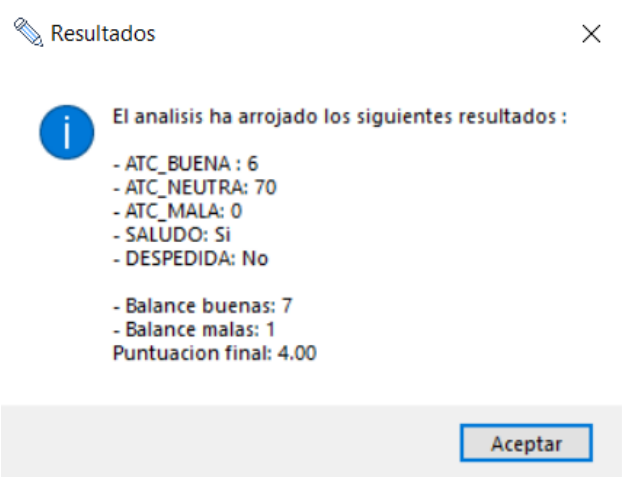


Figura 8: Resultados del análisis

En la pantalla principal también podremos visualizar los resultados de manera gráfica de la categorización, tal y como se muestra en la Figura 9. En donde el color de resalta-do indica el token al que pertenece la palabra.

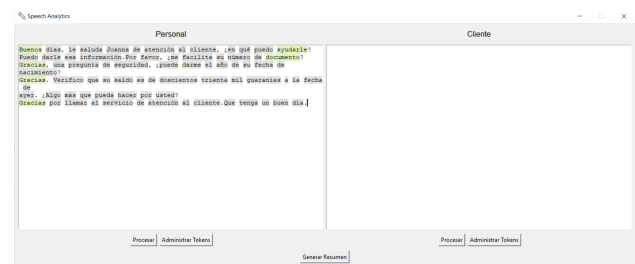


Figura 9: Resultado de la categorización

5.1.5. ADMINISTRACIÓN DE TOKENS

Una vez procesado el texto, también podemos gestionar los tokens de las palabras reconocidas en el texto. Es decir, tenemos la opción de cambiar los tokens ya establecidos para que el tokenizador pueda actualizar su base de datos y aprender conforme vayamos estableciendo los parámetros. Para acceder a esta función, simplemente hacemos clic en el botón **Administrar Tokens**, como se muestra en la Figura 10.

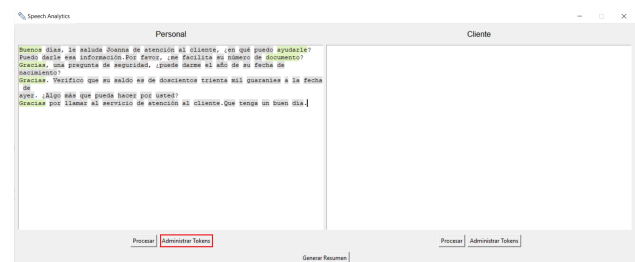
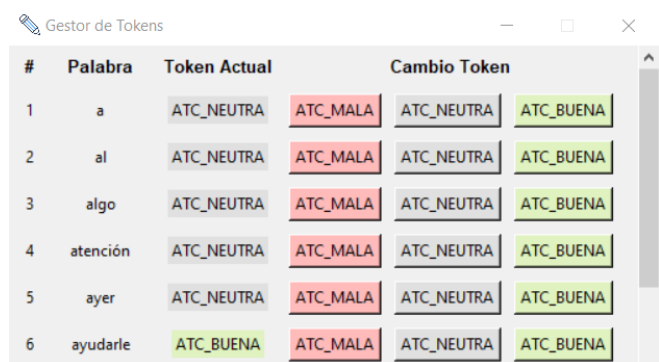


Figura 10: Administración de tokens

A continuación, se desplegará la ventana de administración de tokens, donde tendremos la capacidad de modificar el

token asociado al lexema reconocido, tal y como se puede observar en la Figura 11.



#	Palabra	Token Actual	Cambio Token
1	a	ATC_NEUTRA	ATC_MALA
2	al	ATC_NEUTRA	ATC_MALA
3	algo	ATC_NEUTRA	ATC_MALA
4	atención	ATC_NEUTRA	ATC_MALA
5	ayer	ATC_NEUTRA	ATC_MALA
6	ayudarle	ATC_BUENA	ATC_MALA

Figura 11: Ventana del administrador de tokens

5.2. PROCESAMIENTO DE LA INTERACCIÓN DEL CLIENTE

Todos los pasos descritos anteriormente para la interacción del personal de atención al cliente (ATC) se repiten para el cliente. El resultado final se puede observar en la Figura 12. Esta información es útil para determinar la calidad del servicio ofrecido.

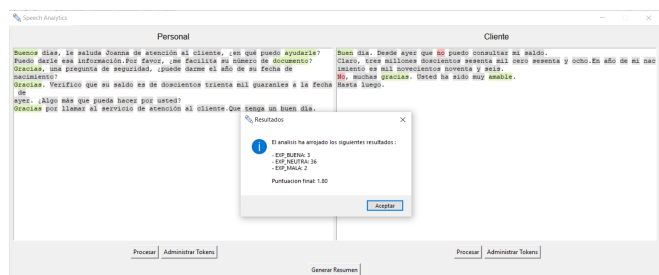


Figura 12: Ventana del administrador de tokens

5.3. RESULTADO Y PUNTUACIÓN GENERAL DE LA INTERACCIÓN

A continuación, para generar el resumen general de la interacción, debemos presionar el botón **Generar Resumen**, como se muestra en la Figura 13.

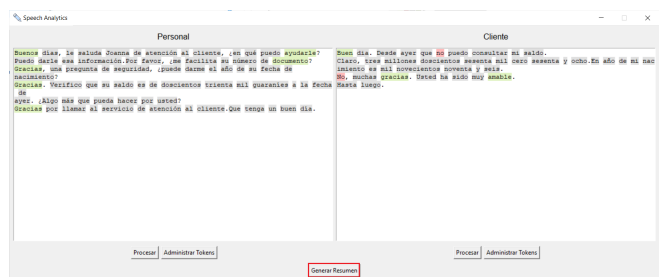


Figura 13: Generación del resumen general

Esto nos generará un resumen de la interacción de la llamada entre el ATC y el cliente. Esta puntuación es un prome-

dio de ambas interacciones y nos proporciona una noción de la puntuación general de la llamada realizada. El resultado se puede observar en la Figura 14.

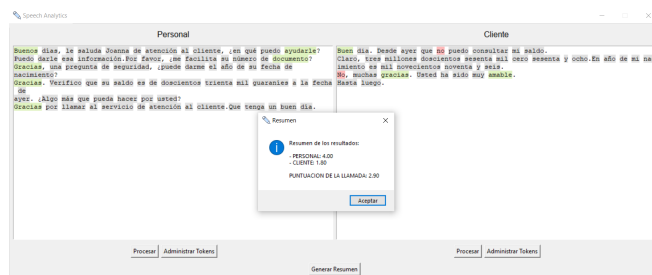


Figura 14: Generación del resumen general

REFERENCIAS

- [1] Real Academia Española. “La Academia entrega la 23.ª edición del DRAE, que se publicará en octubre”. En: *Diccionario de la lengua española* (2014). Consultado el 27 de junio de 2024. URL: <https://www.rae.es>.