UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

IMPACTO DEL AUDIO PARA LA DESCRIPCIÓN AUTOMÁTICA DE VIDEOS (VIDEO TO TEXT)

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN TECNOLOGÍAS DE LA INFORMACIÓN

DIEGO FERNANDO NARANJO MOGOLLONES

PROFESOR GUÍA:
BENJAMIN BUSTOS CÁRDENAS

MIEMBROS DE LA COMISIÓN:
FELIPE BRAVO MÁRQUEZ
JOSÉ MANUEL SAAVEDRA RONDO
CLAUDIO LOBOS YAÑEZ

SANTIAGO DE CHILE
2020

Resumen

En esta tesis se abordó el desafío *video-to-text*: *match and ranking* organizado por *National Institute of Standar and Techonology (NIST)* donde los desafíos y resultados se agrupan en la rama conocida como TREC *Video Retrieval Evaluation* (TRECVID). Este problema consiste en que dado un conjunto de frases F y un conjunto de videos V, ambos del mismo tamaño (#F = #V) se deben ordenar las frases f_i de F de forma que el orden represente el nivel de correspondencia o similitud de cada frase f_i al video v_k. De esta forma cada video tendrá un *ranking* de frases que mejor lo describen. Se poseen datos etiquetados para entrenar y probar la solución. La base de datos para desarrollo y validación es provista por TRECVID, donde se usaron los datos etiquetados de los desafíos de *video-to-text* del 2016, 2017 y 2018.

En este trabajo se resolvió el problema usando el audio de los videos, se utilizó una red neuronal como descriptor para el audio esta fue la red pre-entrenada SoundNet [Aytar et al., 2016]. Como descriptor para las frases se utilizó un *Bag of Words* (BoW) y *word2vec* promediado. Para comparar los audios y las frases en un espacio común se utilizó una Red Neuronal Siamesa. Se realizaron varios modelos para evaluar estas componentes. La validación numéricamente se obtuvo promediando la posición o *rank* de la frase correspondiente al audio del video obtenida al ordenar las predicciones del modelo entre el audio y todas las frases de validación, esto para cada audio de video.

La posición promedio obtenida del *match* asociado al mejor modelo fue *ranking* fue 1,282. Si bien es alejado al mejor resultado posible (1), es mejor que usar un modelo de ordenamiento aleatorio uniforme para crear los *ranking*. Se observó que muchos de los audio de video era nulo o eran sonidos no relacionados al contenido de los cuadros de los videos que son los que describen las frases, esto sería un factor a explicar el desempeño regular usando el audio de los videos obtenido en este desafío particular. Además, el descriptor usado en las frases tiene la pérdida de orden palabras en los vectores resultantes. Los resultados, sin embargo, sugieren que el audio podría ser un aporte, por ejemplo, para ser usado como complemento a un método que analice los cuadros de los videos.

I just wondered,
how things where put together.
Claude Shannon

Sometimes it is the people who no one imagines anything
of who do the things that no one can imagine
Alan Turing

Toda la energía
sigue al pensamiento.
Anónimo

Si no puedes controlarte a ti mismo
no eres libre.
Pitágoras

Agradecimientos

Infinito en tiempo y en espacio, estudiamos y nos desarrollamos como letras que componen una oda al orden y al caos.

Gracias al DCC por todo lo enseñado, en especial a mi profesor guía un caballero, el señor Benjamín Bustos que confío en mí y me apoyó en mi camino. Muchas gracias a los miembros de la comisión por ayudarme a enriquecer mi trabajo. Muchas gracias a las y los tesistas aliados, sin ellos hubiese sido imposible lograrlo.

Mi familia como siempre apoyándome está en mi corazón por la eternidad.

Agradezco a las personas que aparecen citadas en esta tesis por su aporte a las ciencias y al saber de la humanidad.

Quiero agradecer a quien me inició en la computación inteligente, al señor Felipe San Martín y a mi amigo Guillermo Ekdhal compañero en el desarrollo personal.

Diego Fernando Naranjo Mogollones

Tabla de Contenido

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

> Los días los pasaban caminando y caminando, con un caminar silente pero constante, el sudor recorría la frente de todo el clan. Comían cáscaras que eran restos de los frutos que habían encontrado hace meses, la esperanza de encontrar un refugio era como el fuego que encendían por la noche, la cuidaban entre todos... El clan ya había perdido la mitad de sus miembros, fue cuando vieron una roca con seis puntas y tras ella una cueva. Ansiosos y temerosos de encontrar lo que estaban esperando, entraron algunos con algunas lanzas que eran de huesos tallados. El primero llevaba una madera con fuego que poco alumbraba.
>
> Húmeda y segura fue la sensación que se sintió en el ambiente mientras se adentraban, habían piquetes en el suelo como si alguien hace mucho tiempo hubiese estado allí. Pudieron ver en las paredes dibujos de lo que parecía una buena época pasada, la alegría se empezó a apoderar del ambiente. De repente, un miembro del clan emitió un ruido y movió sus manos haciendo señas, lo que atrajo la atención de todos al dibujo que estaba en frente. Era la roca de seis puntas con la bestia de color amarillento y manchas negras que estaba en las pesadillas de todo el clan. Los que estaban, rápidamente, agarraron unas rocas que habían en el suelo y corrieron para salir a defender su esperanza.

Esa fue la primera vez que la información fue usada por el humano.

1.1. Antecedentes

Desde el último par de décadas, las bases de datos multimedia se han hecho cada vez más presentes en el diario vivir, al interactuar con redes sociales, dejar el auto en un estacionamiento, pasar por un semáforo, ver un video, saber quién toca la música que está sonado, recibir recomendaciones de compras por internet, reconocimiento de rostro, patrones de huellas digitales para validar

identidad, predicciones en la bolsa de activos, saber que publicidad colocarte, subtitular canciones, encontrar lo que buscas por internet, y mucho más. El potencial de esta tecnología es muy grande y su combinación con metodologías de otras áreas de las ciencias de la computación (por ejemplo, Inteligencia Artificial) han logrado avances que antes ni se imaginaban. Por eso, es importante ir aplicando los conocimientos que se están generando en el entorno de estas disciplinas para resolver problemas cada vez más difíciles.

Una base de datos como un conjunto de datos organizado será multimedia cuando los datos que la componen son multimedia o sea que usan múltiples medios para ser representados. Volviendo al cuento presentado en el comienzo de la introducción, ese conjunto de dibujos en los muros eran datos multimedia. En la actualidad, un conjunto de imágenes, videos, audios, series de tiempo pueden ser consideradas como datos multimedia.

Para entender el trabajo con bases de datos multimedia se debe comprender el concepto de similitud que es fundamental para poder entender cuándo dos objetos de bases de datos multimedia se parecen ya que dicho concepto es inherentemente subjetivo. Para evitar esto se trata de estandarizar la similitud con diferentes enfoques. Una forma de tipo general es ver las características que componen al objeto en cuestión o abordarlo como un problema de optimización viendo cuánto cuesta transformar a un objeto para obtener otro. Otra forma de abordar la similitud es utilizando el concepto de distancia entre dos objetos para ser analizada, esto permite formalizarlo matemáticamente con espacios vectoriales y, de forma más general, con espacios métricos. Con esto, un objeto de una base de datos multimedia es inducido por la función distancia a transformarse a otro espacio donde la comparación es más efectiva.

En bases de datos multimedia, surge un problema conocido como *video-to-text* que en su versión más general consiste en que dado una base de datos de varios videos multimedia, se requiere un algoritmo que empareje estos videos con descripciones textuales de lo que ocurre en el video. De acuerdo a lo expuesto, un primer paso para poder resolverlo es comenzar controlando ciertas variables del problema, de forma que ciertos parámetros sean acotados y conocidos para poder enfocarse en implementaciones específicas, y medir su funcionalidad. De esta forma, el caso controlado a tratar, corresponde a una base de datos de videos dada, con una cierta cantidad de oraciones a usar como consultas también dadas y se sabe qué oraciones se asocian a cada video, de esta forma se espera poder comprobar si la implementación usada para resolver el problema es efectiva.

Existe dos enfoques para abordar el problema multimedia de coincidencia con texto. Una forma es utilizando los cuadros que componen el video para hacer un análisis secuencial de imágenes con estos cuadros. Otra forma es utilizar el audio del video para poder encontrar la relación con las consultas. En este trabajo se utilizará el audio para implementar la solución final.

El audio es una representación del sonido, una señal sonora que el humano puede escuchar está entre los 20 y $20,000$ hz y al igual que los ojos con la secuencia de imágenes (o colores), el cerebro humano a través del procesamiento de la señal sonora del oído (y el tímpano) es capaz de asociar distintos sonidos a referencias que ha aprendido en toda la vida.

Poder solucionar el problema abordado en este trabajo, en su forma más general, será de ayuda en distintas áreas de la vida cotidiana. Por ejemplo, en seguridad se puede usar para buscar sospechas en cámaras de seguridad buscando la vestimenta del sospechoso. En el sentida, ayudaría a entender mejor videos a discapacitados. Por último, destacarían aplicaciones en marketing al poder clasificar a personas usando el contenido de los distintos videos subidos en redes sociales o mediante acceso a la cámara y micrófono de celulares. Esto último, permitiría poder clasificar las personas por lo que hacen y hacer las predicciones de comportamiento cada vez más precisas pudiendo encontrar tendencias políticas, problemas de salud y casos límite de comportamiento contra las normas establecidas como sociedad.

Desde el 2016, el National Institute of Standards and Technology (NIST) ha organizado el desafío *video-to-text* en su conferencia de procesamiento de video TRECVID. Las consultas de *video-to-text* que se tratan en esta tesis tiene como objetivo corresponder miles de pares videos-oraciones de la red social Vine de Twitter con oraciones predefinidas. La dificultad está en el pequeño conjunto de datos disponible para entrenamiento. Hasta el momento, no ha habido soluciones relevantes que consideren el audio. Se espera usar avances significativos en el área, por ejemplo, la creación de redes neuronales que clasifican audio y poder comprobar como se comportan en este contexto. El uso de estas herramientas ha sido usado efectivamente para la resolución del problema con imágenes. Para cada video o audio se espera construir un *ranking* con las oraciones de acuerdo a su similitud y de tal forma que la oración que le corresponda al video (*match*) esté en primer lugar.

1.1.1. TRECVID

El objetivo principal de la Evaluación de recuperación de video o *Video Retrieval Evaluation* de TREC (TRECVID) es promover el progreso en el análisis basado en contenido y la recuperación de video digital a través de una evaluación abierta basada en métricas. TRECVID es una evaluación estilo de laboratorio que intenta modelar situaciones del mundo real o tareas importantes involucradas en tales situaciones. Tiene más de diez años de existencia aunque su relevancia ha sido cada vez mayor los últimos años.

Dentro de sus principales desafíos está:

- Búsqueda de video *ad-hoc*
- Actividades en video extendido
- Búsqueda de instancias
- *video-to-text*
- *Streaming Multimedia Knowledge*
- Vinculación de narraciones de video en redes sociales

1.1.2. *video-to-text*

El desafío a abordar en este trabajo es el de *video-to-text*. La anotación automática de videos usando descripciones de texto en lenguaje natural ha sido un objetivo remarcable de la visión por computadora. La tarea implica la comprensión de muchos conceptos tales como objetos, acciones, escenas, relaciones persona-objeto, orden temporal de eventos y muchos otros. En los últimos años ha habido avances importantes en las técnicas de visión por computadora que permitieron a los investigadores comenzar a trabajar en la práctica para resolver este problema. Muchos escenarios de aplicación de casos de uso pueden beneficiarse enormemente de dicha tecnología, como el resumen de video en forma de lenguaje natural, facilitando la búsqueda y exploración de archivos de video usando tales descripciones, describiendo videos para ciegos, etc. Además, aprendiendo la interpretación de videos y las relaciones temporales de los eventos en el video probablemente contribuirán a otras tareas de visión por computadora, como la predicción de eventos futuros a partir del video. En este trabajo se espera romper con este paradigma del desafio *video-to-text* utilizando el audio de los videos para aportar al problema. Existe una versión de *Matching and Ranking* o Emparejamiento

[0]https://www-nlpir.nist.gov/projects/tv2019/tv2019.html

y Ordenamiento y una versión de *Description Generation*. En este trabajo se aborda el *Matching and Ranking*.

1.1.3. Base de Datos

Los datos que se usaron corresponden a los datos utilizados para el desafío el 2016, 2017 y 2018, donde se toman videos de la red Twitter Vine, donde cada uno tiene una duración total de aproximadamente 6 segundos.Cada video es descrito Y veces; dado un video cada descripción es hecha por un anotador diferente; por lo tanto, Y representa: 1) la cantidad de anotaciones que tiene un video y 2) la cantidad de anotadores que participaron en la descripción de ese video.

Para datos del 2018, NIST recopiló un conjunto de datos de más de $50k$. Para el *video-to-text* se seleccionó aleatoriamente un subconjunto de aproximadamente $2,000$ videos de Vine. Como se dijo, cada video fue etiquetado Y veces donde $Y \leq 5$.

Para datos del 2017, se recopilaron un conjunto de datos de más de $50k$. Se seleccionó aleatoriamente un subconjunto de 1880 videos. Cada video tiene Y descripciones ($Y \leq 4$) cada una hecha por un anotador diferente.

Para datos del 2016, se recopilaron un conjunto de datos de más de $30k$. Se seleccionaron aleatoriamente un subconjunto de $2,000$. Cada video fue anotado 2 veces ($Y = 2$), esto hecho por dos anotadores. La metodología utilizada fue proporcionar 4 conjuntos de 500 videos no superpuestos a 8 anotadores para generar un total de $4,000$ descripciones de texto. Esas $4,000$ descripciones de texto se dividieron en 2 conjuntos correspondientes a los videos originales de $2,000$.

En cada uno de los años de los datos tomados, se les pidió a los anotadores que incluyan y combinen en 1 oración, si corresponde y está disponible, cuatro facetas del video que están describiendo:

- Quién: describir los actores del video, como por ejemplo objetos y seres concretos (tipos de personas, animales, cosas)
- Qué: describir qué están haciendo los objetos y los seres (acciones genéricas, condiciones/estado o eventos)
- Dónde: describir la locación en la que está pasando tales como sitio, lugar, geográfico, arquitectónico

- Cuándo: decribir la temporalidad en que ocurre tal como la hora del día, temporada o estación

1.1.4. Problema a resolver: *Matching and Ranking*

Dado un conjunto de URL de videos Vine y conjuntos de descripciones de texto, se pide:

1. Devolver para cada URL de video una lista clasificada de la descripción de texto más probable que corresponda (se anotó) al video de cada uno de los conjuntos Y.

2. Los resultados de puntuación serán automáticos contra el *ground truth* utilizando la posición promedio en el *ranking* en la que se encuentra el elemento anotado o equivalente. En este punto se cambia la validación con respecto al *video-to-text* original donde se utiliza el inverso de la posición media en el *ranking*.

1.2. Objetivos

1.2.1. Objetivo General

Evaluar el impacto del audio en la descripción de videos en el problema de *video-to-text*.

1.2.2. Objetivos Específicos

Para lograr el objetivo general se definen los siguientes objetivos específicos:

1. Definir una arquitectura basada en redes neuronales para *video-to-text* utilizando el audio de un video, que permita realizar el *matching* (o emparejamiento) entre videos cortos y oraciones predeterminadas.

2. Realizar una evaluación experimental de la arquitectura implementada, utilizado benchmarks estándar para *video-to-text*.

3. Implementar variantes para robustecer sustentar resultados.

4. Analizar los resultados obtenidos y concluir sobre el potencial uso del audio para *video-to-text*.

1.3. Hipótesis de trabajo

Este trabajo de tesis busca probar la siguiente hipótesis:

El audio de un video contiene información para resolver la tarea de *Matching* y *Ranking* en *video-text*.

El interés de estudio surge debido a la escacez de trabajos similares en el estado del arte, es decir, trabajos en que se ocupe audio para el *Matching and Ranking* en videos. De esta manera, es razonable preguntarse si el audio puede ser una fuente útil de información para resolver el problema. No es directa la utilidad del audio debido la gran cantidad de videos donde el audio es un adorno y no se relaciona con el contenido.

Capítulo 2

Marco Teórico

Primero, se debe entender qué es una base de datos. Una base de datos es una colección organizada de datos, de este término surge el concepto de base de datos relacional que se restringe a una colección de esquemas, tablas, consultas (*queries*), reportes vistas y otros elementos. Los diseñadores de bases de datos, típicamente, organizan los datos para poder modelar aspectos de la realidad de forma de poder dar soporte a procesos que requieran información. Por consiguiente, una base de datos multimedia (MMDB, *multimedia database*) es una colección de datos multimedia [Yu and Brandenburg, 2011]. Los datos multimedia incluyen uno o más tipos de datos multimedia de tipo primario como texto, imágenes, objetos gráficos (dibujos, ilustraciones, esquemas), secuencias de animaciones, audio y videos. Para poder manejar diferentes tipos de datos se usan Sistema de Manejo de Bases de Datos Multimedia (MMDBMS, *Multimedia Database Management System*) que es un *framework* capaz de manejar datos en una gran cantidad de formatos distintos, dentro de un gran arreglo de fuentes multimedia. Da soporte para tipos de datos multimedia, y facilita la creación, almacenaje, acceso, consulta (*query*) y control de una base de datos multimedia [Subrahmanian, 1998].

En métodos de bases de datos multimedia, los descriptores juegan un papel relevante en el procesamiento de éstas. Un descriptor representa las características del dato multimedia, esto permite tener una idea de distintos atributos del dato. Esto admite, por ejemplo, la medición de la similitud entre distintos elementos. El proceso de transformación de espacio de un dato multimedia a un descriptor es llamado como representación o incrustación.

2.1. Audio

En particular, se espera poder trabajar con audio, donde un descriptor clásico para audio son los Coeficientes Cepstrales en las Frecuencias de Mel o MFCC (*Mel Frecuency Cepstral Coefficients*)

que son coeficientes para la representación del timbre del habla basados en la percepción auditiva humana [Knees and Schedl, 2016]. En el área del reconocimiento de audio automático, estos coeficientes surgen de la necesidad de extraer características de las componentes de una señal de audio que sean adecuadas para la identificación de contenido relevante en el reconocimiento natural del lenguaje, así como para obviar todas aquellas que posean información poco valiosa como el ruido de fondo, emociones, volumen, tono, etc. y que no aportan al problema contexto en que surgieron estos coeficientes.

Para obtener estos coeficientes de una señal de audio se separa la señal en pequeños tramos, a cada tramo se le aplica la Transformada de Fourier discreta y se obtiene la potencia espectral de la señal. Luego, se aplica el banco de filtros correspondientes a la Escala Mel al espectro obtenido y se suma las energías en cada uno de ellos [Logan et al., 2000]. Por último, se toma el logaritmo de todas las energías de cada frecuencia Mel, para, finalmente, aplicarle la transformada de coseno discreta a estos logaritmos. Este proceso permitiría obtener los coeficientes buscados. Un gran avance en el procesamiento de señales de audio es la invención de herramientas que resuelven el problema habla-a-texto (*speech-to-text*) que corresponde a dado un audio en el que hay un discurso se busca obtener un texto que corresponda a la secuencia de palabras del audio [Klatt, 1987]. Estas herramientas son efectivas cuando la señal no posee ruido, o ha tenido un procesamiento particular que genera que el discurso posea una alta inteligibilidad.

En paralelo a bases de datos multimedia se ha investigado y avanzado en el tópico de la Inteligencia Artificial (IA, *Artificial Inteligence*) asociado a una máquina "inteligente" como un agente racional, flexible que percibe su entorno y lleva a cabo acciones que maximicen sus posibilidades de éxito en algún objetivo o tarea. La IA es una rama de las ciencias computacionales encargada de estudiar modelos de cómputo capaces de realizar actividades propias de los seres humanos en base a dos de sus características primordiales: el razonamiento y la conducta. Esta se apoya en cuatro pilares: Búsqueda del estado requerido en el conjunto de los estados producidos por las acciones posibles, Algoritmos Genéticos, Redes Neuronales Artificiales y Razonamiento mediante Lógica Formal.

2.2. Redes Neuronales

Las redes neuronales son un modelo computacional basado en una gran cantidad de capas de unidades neuronales simples, cada neurona está conectada con otras a través de enlaces. En estos,

el valor de salida de la neurona anterior es multiplicado por el valor de un peso, controlando la acción de las neuronas adyacentes [Hagan et al., 1996]. Estos sistemas aprenden y se forman a sí mismos y sobresalen en las áreas donde la detección de las características es difícil de expresar con la programación convencional. En el entrenamiento, los valores de los pesos de las neuronas se van actualizando de forma que se minimice alguna función de pérdida que evalúa la red en su total, esto se hace mediante la propagación hacia atrás o *backpropagation* [Werbos et al., 1990]. Dentro de los tipos de redes neuronales, se encuentran las redes neuronales convolucionales [Krizhevsky et al., 2012] usadas para problemas con datos multimedia de alta complejidad, como imágenes. Estas redes se desarrollan en la práctica en proyectos, donde se destaca: LeNet-5 [LeCun et al., 1995] usada para reconocer dígitos en cheques bancarios, AlexNet [Krizhevsky et al., 2012] que cumple con el desafío de conocimiento visual a gran escala, VGGNet [Simonyan and Zisserman, 2014] permite reconocer escenas en imágenes, ResNet [He et al., 2016] que es el último gran proyecto de detección de imágenes a la fecha.

En audio, destaca un proyecto conocido como SoundNet [Aytar et al., 2016]. En este proyecto, se aprendieron representaciones de sonido natural con una gran cantidad de videos sin etiquetar donde se usa la sincronización entre visión y sonido para vincular la representación acústica. SoundNet es entrenada con Places365 y con el modelo VGG16 . De esta forma, SoundNet permite clasificar las escenas y objetos del audio.
Con la aparición del incentivo a la investigación con el desafío DECASE (*Detection and Classification of Acoustic Scenes and Events*) organizado por varias universidades en conjunto el 2013 y del 2016 en adelante, aparecen proyectos de audio que avanzan en el estado del arte a de la solución del problema clasificar audios de la vida real en objetos y escenas.

Otro proyecto destacado es AENet que propone una nueva red profunda para reconocimiento de eventos de audio [Takahashi et al., 2017].

El uso de redes neuronales como representación para datos multimedia es cada vez mayor, donde la entrada de la red es el dato multimedia a estudiar y la salida es el vector característico que se usará como representación numérica de dicho dato.

2.2.1. SoundNet

SoundNet es una red neuronal desarrollada el 2016, la cual fue hecha a partir de una gran cantidad de datos de sonido sin etiquetas recolectados de fuentes públicas. Esta red está basada

en la sincronización natural entre video y sonido para aprender a tener representaciones acústicas usando dos millones de videos sin etiquetar [Aytar et al., 2016]. En este proyecto, se propone un procedimiento de entrenamiento estudiante-profesor que transfiere el conocimiento visual discriminatorio de modelos visuales establecidos a la modalidad del sonido usando videos sin etiquetar como puente. Esto significa obtener importantes características del sonido que permiten acercarnos a una representación más útil de él. Otros estudios se enfocan en características como espectrogramas y coeficientes MFCC, por otro lado, SoundNet se enfoca en el sonido natural.

La figura 2.1 muestra la arquitectura de SoundNet que resulta de la sincronización entre visión y sonido para aprender representaciones acústicas de videos sin etiquetar. La ventaja de videos sin etiquetar es la facilidad con que es adquirido a escalas masivas. De esta forma, SoundNet está compuesta por una red convolucional profunda que aprende directamente de ondas de audio sin procesar. Aunque esta red es entrenada con supervisión visual, esta no presenta dependencia en la visión durante la inferencia [Aytar et al., 2016].

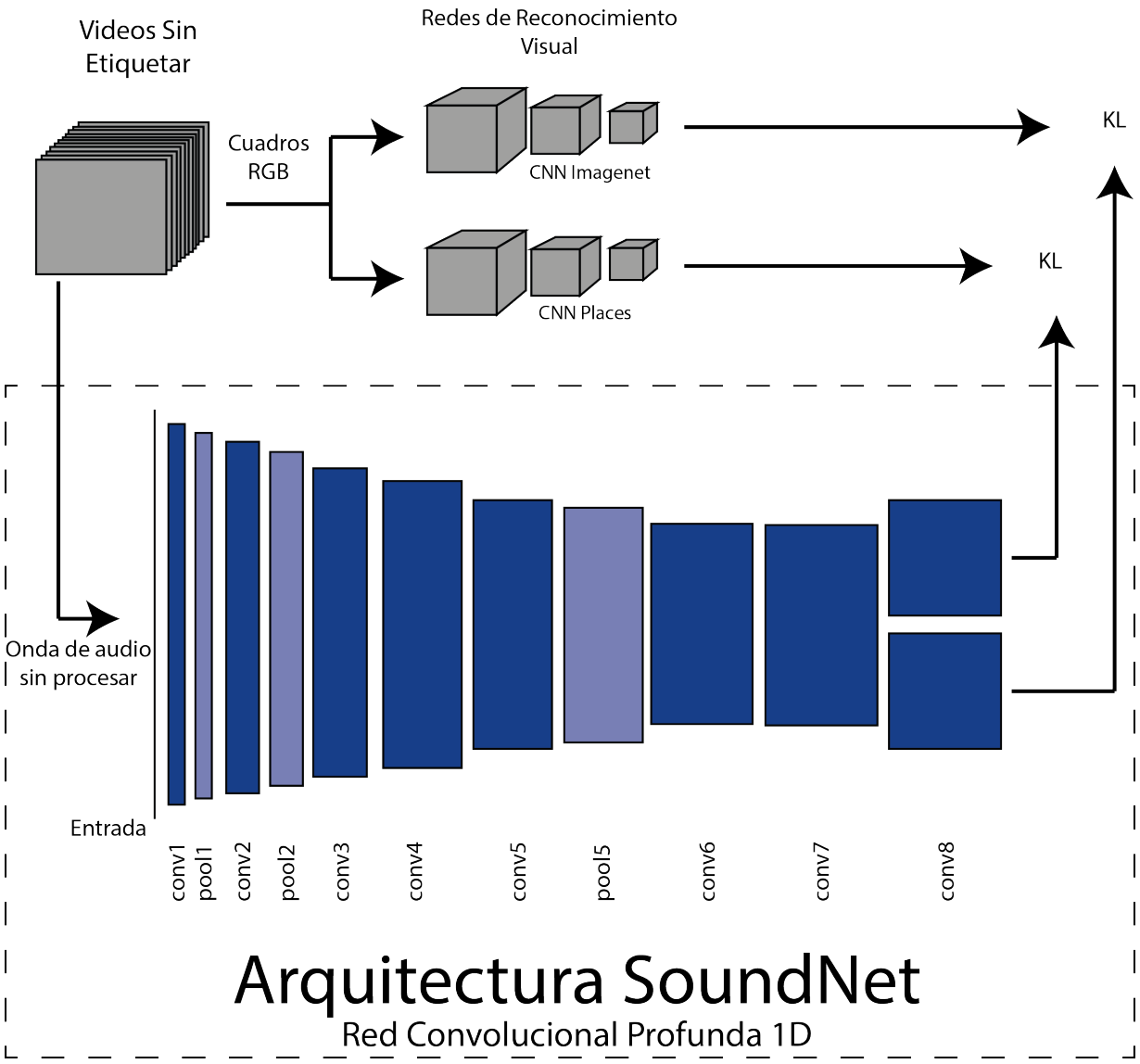

Figura 2.1: Muestra la arquitectura de SoundNet, compuesta por una red convolucional profunda para un reconocimiento natural del sonido. Se entrena la red transfiriendo el conocimiento discriminativo de reconocimiento visual a redes de sonido, esto se basa en la sincronización entre visión y sonido del video. Basado en [Aytar et al., 2016].

Con respecto al estado del arte al punto del desarrollo de SoundNet, que es muy similar al actual, el entendimiento del audio a escala amplia había sido estudiado extensivamente en el contexto de la música [Bertin-Mahieux et al., 2011, Van den Oord et al., 2013] y reconocimiento de voz [Hannun et al., 2014]. La clasificación de escenas basado en la acústica, clasifica el sonido, extrayendo de la acústica la escena/el objeto había sido predominantemente basado en aplicar una variedad de clasificadores generales (SVM, GMMs, etc.) a características del sonidos generadas manualmente (MFCC, espectogramas, etc.) [Gupta et al., 2016, Rakotomamonjy and Gasso, 2015, Roma et al., 2013, Stowell et al., 2015]. Incluso aunque hay métodos de aprendizaje profundo sin supervisión [Lee

et al., 2009] y supervisados [Piczak, 2015b, McLoughlin et al., 2015, Cakir et al., 2015, Hertel et al., 2016] aplicados a la clasificación del sonido, estos modelos son limitados por la cantidad de datos de sonido natural etiquetado. Sin embargo, la mayoría de los trabajos no se enfocan en el entendimiento del sonido natural, esto no se hacía debido a que colectar datos etiquetados de sonido es costoso y ambiguo. Para manejar el problema de las etiquetas para estos tipos de sonidos, en SoundNet los autores aprovecharon la relación natural entre imágenes y sonidos en videos usando un procedimiento de estudiante-profesor.

SoundNet es entrenada como una red convolucional profunda con una base de datos de larga escala ($2M$ de videos), principalmente, de Flickr con videos de toda clase destacando temáticas de Playas, Clases, Construcción, Ríos, *Club*, Bosques, *Hockey*, Sala de Juegos, Motores y Vegetación. La forma de operar de SoundNet es con ondas de sonido sin procesar, por lo que de los videos el único post-procesamiento hecho fue convertir el sonido de estos a mp3, reducir la tasa de muestra a 22kHz y convertirlos a audio de un sólo canal. También, se escaló la amplitud de la onda en el rango [-256, 256] donde no hubo sustracción de la media, bajo argumento que la onda estaba naturalmente cercana a estar centrada en 0.

Dentro de las principales características de SoundNet se destaca:

- Una red convolucional sin supervisión la cual aprende directamente de la onda de audio bruta, sin preprocesamiento de los datos de entrada.
- Una red convolucional para audio que es entrenada usando el video sin etiquetar como puente.
- Sin etiquetas, de sonido

La red convolucional presentada para la arquitectura de aprendizaje de representaciones de sonido, corresponde a una serie de convoluciones de una dimensión seguidas de operaciones no lineales (en otras palabras, capas ReLU) usado para procesar el sonido. También, posee capas de *pooling* que permite controlar el largo de las entradas. De esta forma la arquitectura de SoundNet se caracteriza:

1. Red totalmente convolucional: Capa convolucional de 1D + capa de activación ReLU. El uso de capas convolucionales en sonido se justifica en la propiedad de invariante a la translación reduciendo el número de parámetros y que ellas permiten apilar capas, lo que es útil para detectar conceptos de alto nivel.
2. Capas de *pooling*: que permiten disminuir el largo variable de las entradas.

Debido a que el sonido puede variar el largo temporal, se usa una red totalmente convolucional, debido a que las capas convolucionales son invariantes a la posición, se puede convolucionar cada capa dependiendo del largo del input, de esta forma, la arquitectura sólo usa capas convolucionales y de *pooling*. Para diseñar capas de salida que funcionen con entradas de distinto tamaño, se usa una capa de salida convolucional para producir una salida sobre múltiples pasos de tiempo en video.

La profundidad de la red es alta debido a la cantidad de datos disponibles para entrenar, aunque los experimentos hechos en [Aytar et al., 2016] son con redes de 5 y 8 capas. En este reporte, debido a la mejor efectividad, se usa la de 8 capas (figura 2.1). La arquitectura consiste en 8 capas convolucionales y 3 capas de *max-pooling* (figura 2.1).

Capa	**conv1**	**pool1**	**conv2**	**pool2**	**conv3**	**conv4**	**conv5**	**pool5**	**conv6**	**conv7**	**conv8**
Dim.	200,050	27,506	13,782	1,722	862	432	217	54	28	15	4
# de Filtros	16	16	32	32	64	128	256	256	512	1024	1401
Tamaño Filtro	62	8	32	8	16	8	4	4	4	4	8
Stride	2	1	2	1	2	2	2	1	2	2	2
Padding	32	0	16	0	8	4	2	0	2	2	0

Tabla 2.1: Muestra la configuración de las capas para las 8 capas de SoundNet. [Aytar et al., 2016]

Aunque se entrena SoundNet para clasificar categorías visuales, las categorías que gustarían que reconocieran pueden no aparecer en modelos visuales (por ejemplo, estornudos). Esto se subsana ignorando la capa de salida de la red y usando la representación interna como característica para entrenar clasificadores, usando pequeñas cantidades de datos de sonidos etiquetados y entrenar un clasificador de *Support Vector Machine (SVM)* con temas de interés.

El optimizador usado es *Adam* (tasa de aprendizaje de 0.001 y el término de momentum de 0.9). Para clasificación de multiclases se usa la estrategia de uno-contra-todos, además, usa validación cruzada para elegir el hiperparámetro de regularización marginal. Para mayor robustez, se sigue un procedimiento de aumento de datos estándar en el que cada muestra de entrenamiento se divide en extractos sonoros de longitud fija que se superponen, en los que se calcula las características y se utilizan para el entrenamiento. Durante la inferencia, se promedian las predicciones en todas las ventanas.

La implementación es hecha en Torch7. Los experimentos hechos en [Aytar et al., 2016] consistieron en validar la arquitectura de SoundNet con respecto a otros clasificadores donde SoundNet fue la mejor. Contra características «hechas a mano» SoundNet mejora en un 10 % el *accuracy* sobre

los datos dividiéndolos en datos de entrenamiento y validación (ver tabla 2.2). Usando varios métodos sobre bases de datos de sonidos ambientales (ESC-50 y ESC-10) mejora el comportamiento de varias lineas base (ver tabla 2.3). Finalmente, se sugiere que el uso de sonido para reconocer escenas y objetos, en comparación a usar las secuencias de imágenes es peor dado un experimento de reconocimiento multimodal, pero, el uso en conjunto mejora cada una de las efectividades mostradas independientes (medido con *accuracy*).

Método	***Accuracy***
RG	69 %
LTT	72 %
RNH	77 %
Ensemble	78 %
SoundNet	88 %

Tabla 2.2: Muestra la evaluación de la clasificación en la base de datos de validación. SoundNet, generalmente, mejora en un 10 % la efectividad. Se compara con respecto a RG [Rakotomamonjy and Gasso, 2015], LTT [David Li and Touh, 2013], RNH [Roma et al., 2013] y Ensemble [Stowell et al., 2015]. [Aytar et al., 2016]

	Accuracy en	
Método	**ESC-50**	**ESC-10**
SVM-MFCC	39.6 %	67.5 %
Convolutional Autoencoder	39.9 %	74.3 %
Random Forest	44.3 %	72.7 %
Piczak ConvNet	64.5 %	81.0 %
SoundNet	74.2 %	92.2 %
Actuación Humana	81.3 %	95.7 %

Tabla 2.3: Muestra la evaluación de la clasificación en las bases de datos ESC. Los resultados sugieren que SoundNet mejora las lineas base. Se compara con respecto a SVM, Convolutional Autoencoder, Random Forest [Piczak, 2015b], Piczak ConvNet [Piczak, 2015a] y Actuación Humana [Piczak, 2015b]. [Aytar et al., 2016]

Para la matemática del entrenamiento es importante tener en cuenta que en la fase de entrenamiento fue usado video, pero el objetivo es reconocer sonidos. Por esta razón, en el modelo compilado no hay videos como entrada.
El problema ha sido modelado de una perspectiva estudiante-profesor. En este caso, redes bien establecidas para visión enseñarán a la red para sonido a reconocer escenas y objetos.

Las entradas del modelo son ondas de audio sin procesar $x_i \in \mathbb{R}^D$ y video $y_i \in \mathbb{R}^{3xTxWxH}$ para cada i con $1 < i < N$, donde W, H y T son el ancho, la altura y el número de cuadros de muestreo en el video, respectivamente.

Durante el aprendizaje el foco es obtener las probabilidades $g_k(y_i)$ de la red de visión profesora para poder entrenar al a la red de audio estudiante $f_k(x_i)$, k enumera el número de conceptos transferidos del profesor al estudiante.

De esta forma, ellos definen una función de pérdida,

$$L(g_k, f_k) = min_\theta \sum_{k=1}^{K} \sum_{i=1}^{N} D_{KL}(g_k(y_i)||f_k(x_i;\theta))$$

donde D es la divergencia de Kullback-Leibler

$$D_{KL}(P||Q) = \sum_{j=1} P_j \times log\frac{P_j}{Q_j}$$

Esta es una medida de como una probabilidad de distribución es diferente de una segunda. Los autores eligieron la divergencia KL porque es diferenciable, luego es posible de optimizarla usando Propagación hacia atrás y en descenso de gradiente estocástico.

2.3. *Representación de Oración*

La representación de oraciones es un conjunto de técnicas en el procesamiento de lenguaje natural donde las oraciones se asignan a vectores de números reales.

Dentro del estado del arte actual destacan varias metodologías para tratar este problema por ejemplo: *Sentence Embedding* [Le and Mikolov, 2014, Arora et al., 2016, Wieting et al., 2015] con el BoW [Mikolov et al., 2013a, Mikolov et al., 2013b] de amplio uso o el *FastText* [Bojanowski et al., 2017, Joulin et al., 2016]. El uso de arquitecturas más complejas basada en redes neuronales, por ejemplo, los *SkipThought vectors* [Kiros et al., 2015] proponen una función objetivo que se adapta al modelo de *skip-gram* para palabras [Mikolov et al., 2013b] en el nivel de oración, codificando una oración para predecir la oración en el entorno de esta, y usando características en un modelo lineal, fueron capaces de demostrar buen rendimiento en 8 tareas de transferencia; uso redes neuronales recurrentes o *RNN, Recurrent Neural Networks* tipo GRU *Gated Recurrent Units* [Cho et al., 2014, Chung et al., 2014] o LSTM *Long Short-Term Memory* [Hochreiter and Schmidhuber, 1997, Conneau et al., 2017] son usadas en arquitecturas complejas para resolver el problema de descripción de oraciones.

2.3.1. *Bag of Words*

El modelo de *Bag of Words* o BoW ha sido usado ampliamente en representación de oraciones desde su primeras formulaciones [Salton et al., 1975]. Se basa en que el texto (una oración o un documento) es representado como una bolsa (conjunto) de sus palabras sin importar el orden. La bolsa-de-palabras es comúnmente usada en métodos de clasificación de documentos donde la (frecuencia de) ocurrencia de cada palabra es una como una característica para el entrenamiento del clasificador. Este modelo involucra:

1. Un vocabulario de palabras conocido
2. Una medida de la presencia de las palabras conocidas

Implementación: ejemplo

A continuación se enmarcará un ejemplo de implementación

1. Recolección de data

   Se toma una serie de oraciones:

   It was the best of times
   it was the worst of times
   it was the age of wisdom

   it was the age of foolishness

2. Designación de vocabulario

   Se toma una lista de todas las palabras en el modelo de vocabulario. Las palabras únicas ignorando mayúsculas y signo de puntuación son:

   - "it"
   - "was"
   - "the"
   - "best"
   - "of"
   - "times"
   - "worst"
   - "age"

- “wisdom”
- “foolishness”

Esto es un vocabulario de 10 palabras en un *corpus* de 24 palabras.

3. Creación de los vectores de cada oración

   El siguiente paso es darle puntaje a cada oración. El objetivo es transformar cada oración o documento en un vector que pueda ser usado como entrada para algún algoritmo de *machine learning*.

   Como sabemos que el vocabulario es de 10 palabras, se fija un vector de largo de 10 dimensiones donde cada componente en él será el puntaje de la presencia o no de una palabra en la oración. El método más sencillo para puntuar es asignar un valor booleano, 0 para la ausencia de la palabra en la oración y 1 para la presencia.

   Usando un orden arbitrario de las palabras, podemos pasar por la primera oración (“It was the best of times“) y convertirla en un vector binario. La puntuación sería la siguiente:

   - “it” = 1
   - “was” = 1
   - “the” = 1
   - “best” = 1
   - “of” = 1
   - “times” = 1
   - “worst” = 0
   - “age” = 0
   - “wisdom” = 0
   - “foolishness” = 0

   Y el vector binario se vería:

   $$[1, 1, 1, 1, 1, 1, 0, 0, 0, 0]$$

   Los otros tres se verían así:

   $$\text{"}it\ was\ the\ worst\ of\ times\text{"} = [1,\ 1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0]$$

   $$\text{"}it\ was\ the\ age\ of\ wisdom\text{"} = [1,\ 1,\ 1,\ 0,\ 1,\ 0,\ 0,\ 1,\ 1,\ 0]$$

$$\text{"}it\ was\ the\ age\ of\ foolishness\text{"} = [1,\ 1,\ 1,\ 0,\ 1,\ 0,\ 0,\ 1,\ 0,\ 1]$$

Nuevas oraciones o documentos que tengan palabras en común con el vocabulario puede ser modeladas, incluso si tienen palabras fuera del vocabulario donde solo la presencia de palabras conocidas del vocabulario son puntuadas, el resto se ignoran.

A medida que el vocabulario crece, también lo hace el vector de representación y esto complica tanto en memoria como en complejidad de aprendizaje del algoritmo posterior a usar. Existen métodos para evitar esto como ignorar palabras frecuentes conocidas como *stop words* (en inglés, por ejemplo, «a, of , ...»), reducir lo verbos conjugados o tomar las palabras más relevantes para caracterizar.

Una mirada más sofisticada es agrupar las palabras continuas de forma de ir capturando más el significado del documento. En este abordaje cada palabra o *token* es llamado *gram*. Crear un vocabulario de pares de palabras es llamado *bigram*. Un ejemplo de como se vería el vocabulario en un bigrama en la oración "It was the best of times" sería:

- "it was"
- "was the"
- "the best"
- "best of"
- "of times"

2.3.2. *word2vec*

word2vec es un software desarrollado por un equipo liderado por Tomas Mikolov [Mikolov et al., 2013b].

Modelo *skip-gram*

El punto de partida de la teoría tras el software es el modelo *skip-gram*. En este modelo dado un *corpus* de palabras w y su contexto c. Se considera la probabilidad condicional $p(c|w)$ y un *corpus text*, y la meta es configurar los parámetros θ de $p(c|w;\theta)$ para maximizar la probabilidad del *corpus*:

$$arg\ \max_{\theta} \prod_{w\ \epsilon\ text} \left[\prod_{c\ \epsilon\ C(w)} p(c|w:\theta) \right] \tag{2.1}$$

en esta ecuación, $C(w)$ es el conjunto de conjutos de contextos de palabras w. Alternativamente:

$$arg\ \max_{\theta} \prod_{(w,c)\ \epsilon\ D} p(c|w;\theta) \tag{2.2}$$

aquí D es el conjunto de todos los pares de palabras y contextos que se extraen del texto.

Parametrización del modelo *skip-gram*

Un acercamiento para parametrizar el modelo *skip-gram* sigue los modelos de redes neuronales de lenguaje (literatura), y modelos condicionales de probabilidad $p(c|w;\theta)$ usando *soft-max*:

$$p(c|w;\theta) = \frac{e^{v_c \cdot v_w}}{\sum_{c'\epsilon\ C} e^{v_{c'} \cdot v_w}} \tag{2.3}$$

donde v_c y $v_w\ \epsilon R^d$ son representaciones vectoriales para c y w, respectivamente, y C es el conjunto de todos los contextos disponibles. Los parametros θ y v_{c_i}, v_{w_i} para $w \epsilon V$, $c \epsilon C$, $i \epsilon 1, ..., d$ (un total de $|C| \times |V| \times d$ parámetros). Se trata de asignar parámetros tal que el producto de 2.2 sea maximizado.

Para cambiar de producto a suma se aplica la función logaritmo:

$$arg\ \max_{\theta} \sum_{(w,c)\ \epsilon D} log\ p(c|w) = \sum_{(w,c)\ \epsilon D} \left(log\ e^{v_c \cdot v_w} - log \sum_{c'} e^{v_{c'} \cdot v_w} \right) \tag{2.4}$$

Una suposición del proceso de incrustación es que maximizando 2.4 resultará en buenas incrustaciones $v_w\ \forall w\ \epsilon\ V$, en ese sentido palabras similares tendrán vectores similares.

Mientras el objetivo 2.4 puede ser calculado, es computacionalmente costoso porque el término $p(c|w;\theta)$ es muy costoso para calcular debido a la suma $\sum_{c'\ \epsilon\ C} e^{v_{c'} \cdot v_w}$ sobre todos los contextos c' (pueden ser cientos o miles de ellos). Una forma de hacer la computación más abordable es reemplazar la función *softmax* con una función *hierarchical softmax*.

Muestreo Negativo

En desarrollo hecho por Mikolov, se presenta el abordaje por muestreo negativo como una forma más eficiente de derivar alguna representación de palabras. Mientras que el muestreo negativo es basado en el modelo de *skip-gram*, es de hecho optimizando un diferente objetivo. Lo que seguirá es la derivación del objetivo del muestreo negativo.

Considere un par (w, c) de palabra y contexto. La interrogante a notar es si este par proviene de los datos de entrenamiento. Denotaremos por $p(D = 1|w, c)$ la probabilidad de que (w, c) venga de

los datos del *corpus*. Correspondientemente, $p(D = 0|w, c) = 1 - p(D = 1|w, c)$ será la probabilidad de que (w, c) no provenga del *corpus* de los datos. Como antes, se asumirá que hay parámetros θ controlando la distribución de probabilidad: $p(D = 1|w, c; \theta)$. El objetivo es ahora encontrar parámetros para maximizar las probabilidades de que todas las observaciones, en efecto, vengan de los datos:

$$\begin{aligned} & arg\ \max_{\theta} \prod_{(w,c)\ \epsilon\ D} p(D = 1|w, c; \theta) \\ &= arg\ \max_{\theta} log \prod_{(w,c)\ \epsilon\ D} p(D = 1|w, c; \theta) \\ &= arg\ \max_{\theta} \sum_{(w,c)\ \epsilon\ D} log\, p(D = 1|w, c; \theta) \end{aligned}$$

La cantidad $p(D = 1|c, w; \theta)$ puede ser definida usando la función *softwax*:

$$p(D = 1|w, c; \theta) = \frac{1}{1 + e^{-v_c \cdot v_w}}$$

Esto lleva al objetivo:

$$arg\ \max_{\theta} \sum_{(w,c)\ \epsilon\ D} log\ \frac{1}{1 + e^{-v_c \cdot v_w}}$$

Este objetivo tiene una solución trivial si se asigna θ como $p(D = 1|w, c; \theta) = 1$ para cada par (w, c). Este puede ser fácilmente calculado definiendo un θ tal que $v_c = v_w$ y $v_c \cdot v_w = K$ para todo v_c, v_w, donde K es un número lo suficientemente grande (en la práctica, se obtiene una probabilidad cercana a 1 cuando $K \approx 40$).

Se necesita un mecanismo que prevenga que todos los vectores tengan el mismo valor, esto, rechazando algunas combinaciones (w, c). Una forma para hacerlo, es presentar un modelo con algunos pares (w, c) para los cuales $p(D = 1|w, c; \theta)$ deba ser bajo, es decir, pares que no estén en los datos. Este es logrado generando un conjunto D' de datos aleatorios de pares (w, c), asumiendo que ellos están todos incorrectos (el nombre «muestreo negativo» es debido a este conjunto D' de ejemplos negativos aleatoriamente generados). La optimización objetivo ahora queda:

$$\begin{aligned} & arg\ \max_{\theta} \prod_{(w,c)\ \epsilon\ D} p(D = 1|w, c; \theta) \prod_{(w,c)\ \epsilon\ D'} p(D = 0|w, c; \theta) \\ &= arg\ \max_{\theta} \prod_{(w,c)\ \epsilon\ D} p(D = 1|w, c; \theta) \prod_{(w,c)\ \epsilon\ D'} (1 - p(D = 1|w, c; \theta)) \end{aligned}$$

$$= arg\ \max_{\theta} \sum_{(w,c)\ \epsilon\ D} log\ p(D=1|w,c;\theta) \sum_{(w,c)\ \epsilon\ D'} log\ (1-p(D=1|w,c;\theta))$$

$$= arg\ \max_{\theta} \sum_{(w,c)\ \epsilon\ D} log\ \frac{1}{1+e^{-v_c \cdot v_w}} \sum_{(w,c)\ \epsilon\ D'} log\ \left(1-\frac{1}{1+e^{-v_c \cdot v_w}}\right)$$

$$= arg\ \max_{\theta} \sum_{(w,c)\ \epsilon\ D} log\ \frac{1}{1+e^{-v_c \cdot v_w}} \sum_{(w,c)\ \epsilon\ D'} log\ \frac{1}{1+e^{v_c \cdot v_w}}$$

Si definimos

$$\sigma = \frac{1}{1+e^{-x}}$$

obtenemos:

$$arg\ \max_{\theta} \sum_{(w,c)\ \epsilon\ D} log\ \frac{1}{1+e^{-v_c \cdot v_w}} \sum_{(w,c)\ \epsilon\ D'} log\ \frac{1}{1+e^{v_c \cdot v_w}}$$

$$= arg\ \max_{\theta} \sum_{(w,c)\ \epsilon\ D} log\ \sigma(v_c \cdot v_w) \sum_{(w,c)\ \epsilon\ D'} log\ \sigma(-v_c \cdot v_w)$$

Este desarrollo del objetivo expuesto en estas es lineas [Goldberg and Levy, 2014] es una explicación del desarrollo hecho por Mikolov con respecto a *word2vec*. El objetivo presentado acá es para todo el *corpus* $D \cup D'$, a diferencia del desarrollo de Mikolov que es presentada para cada ejemplo $(w,c)\ \epsilon\ D$ y k ejemplos $(w,c_j)\ \epsilon\ D'$, siguiente la forma particular de construir D'.

Específicamente, con el muestreo negativo k se construye D' k veces más grande que D y para cada $(w,c)\ \epsilon\ D$ se construyen k muestras (w,c_1), ... , (w,c_k), donde cada c_j es tomado de acuerdo a una distribución unigramas elevado a la potencia de $3/4$. Este es equivalente a tomar las muestras (w,c) en D' de la distribución $(w,c) \sim p_{words}(w)\frac{p_{contexts}(c)^{3/4}}{Z}$, donde $p_{words}(w)$ y $p_{contexts}(c)$ son de la distribuciones unigramas de palabras y contextos, respectivamente, y Z es una constante de normalización. Cada contexto es una palabra (y todas las palabras aparecen en un contexto), y, por lo tanto $p_{context}(x) = p_{words}(x) = \frac{count(x)}{|Text|}$ [Mikolov et al., 2013b].

Todo esto se basa en la hipótesis de que palabras en similares contextos tienen similares significados.

2.4. Red Siamesa

Una Red Siamesa es una red de aprendizaje semi-supervisada que produce una representación de características para las entradas. Introduciendo entradas de canales múltiples en la red y una

función apropiada de pérdida (*loss function*), la Red Siamesa es capaz de representar entradas similares con características semejantes de representación y proyectar diferentes entradas con diferentes vectores [Guo et al., 2017, Dong and Shen, 2018]).

Usualmente, el vector de características es un vector de alta dimensión. La similitud se representa por una función de distancia Euclidiana en un espacio de alta dimensionalidad.

La figura 2.2 muestra una Red Siamesa con dos canales de entrada. Las dos ramas de redes idénticas, que en este caso son una Red Neuronal Convolucional (CNN), comparten los mismos pesos. En adición a CNN, la arquitectura puede ser cualquier red neuronal. Debe notarse que dos redes hermanas pueden ser de la misma o diferente arquitectura. Incluso si las dos redes hermanas poseen la misma arquitectura, estas no tienen que compartir los pesos, pueden ser independientes una de la otra. Usualmente, si las entradas son de diferente «tipo», las redes hermanas usualmente, serán de distinta arquitectura o usarán distintos pesos para la misma arquitectura.

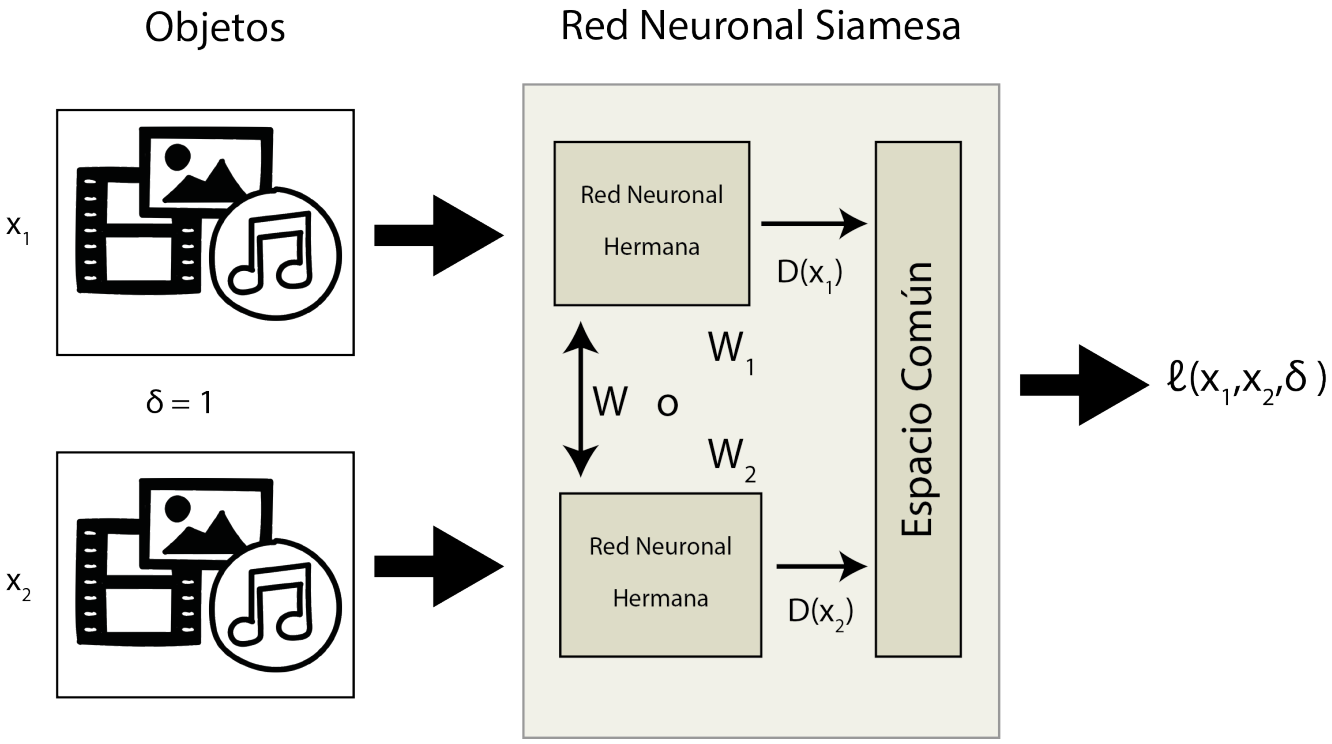

Figura 2.2: Muestra un esquema de una Red Neuronal Siamesa.

La distancia calculada al final de la red (generalmente, distancia Euclidiana) de las salidas de dos canales es calculada y sujeta a la minimización de la *loss function* $l(x_1, x_2, \delta)$. Una función muy común usada es la llamada función de triplete o *Triplet Loss* [Chechik et al., 2010]. Esta es una

función de pérdida, al igual que las demás utilizadas en redes siamesas, basada en distancia (opuesto a función de pérdida de predicción basada en error como las usada en clasificación como por ejemplo la *Logistic loss* o *Hinge loss*).
La función de pérdida de triplete es una función de pérdida para redes neuronales artificiales donde una entrada de línea de base (ancla o *anchor*) se compara con una entrada positiva (verdadera o *truthy*) y una entrada negativa (falsa *falsy*). La distancia desde la entrada de línea de base (ancla) a la entrada positiva (verdad) se minimiza, y la distancia desde la entrada de línea de base (ancla) a la entrada negativa (falsa) se maximiza. La función de pérdida se puede describir usando la función de distancia Euclideana:

$$\mathcal{L}(A, P, N) = max\left(||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha, 0\right)$$

donde A es la entrada ancla, P es la entrada positiva de la misma clase que A, N es la entrada negativa de una diferente clase de A, α es el margen entre los pares positivos y negativos. Esta función ha sido utilizada en la solución de problemas de ranking ([Chechik et al., 2010]).

También, encontramos la función de pérdida contrastiva o *contrastive loss* ([Hadsell et al., 2006]) o la función de pérdida central o *Centre Loss*. Más usada es la función contrastiva que es utilizada, generalmente, para tarea de recuperación de imágenes para aprender características discriminativas de imágenes. Durante el entrenamiento, un par de imágenes alimentan el modelo con su relación de verdad y (1 si son de la misma clase, 0 si no). La función de pérdida para un solo par es:

$$yd^2 + (1 - y)max(\alpha - d, 0)^2$$

donde d es una distancia Euclidiana entre dos características de imágenes (si sus características fueran f_1 y f_2): $d = ||f_1 - f_2||_2$.. α es el término usado para tener como margen.

2.5. Validación Cruzada

Estimar la precisión de un clasificador inducido por algoritmos de aprendizaje supervisado es importante no sólo para predecir la precisión futura, sino también para poder elegir un clasificador de un conjunto dado (selección de modelo), o combinar clasificadores ([Wolpert, 1992]). La validación cruzada o *cross validation* ha sido ampliamente estudiada (por ejemplo en [Kohavi et al., 1995]) y utilizada para probar la efectividad de modelos. Esto enmarcado en que al entrenar un modelo de forma directa no se puede asumir que va a funcionar bien en datos que no han sido vistos antes. Para evaluar el rendimiento de un algoritmo de inteligencia de maquinas se debe probar en

datos que no han sido vistos, en base a esto podemos analizar si nuestro modelo está en respuesta *over-fitting* - *under-fitting* o está bien generalizado. Para realizar la validación cruzada tenemos que tener aparte una muestra que será una porción de los datos que no se usará para entrenar al modelo para después usar este modelo para pruebas y validación.

El procedimiento general es mostrado en las figuras 2.3 y 2.4 y es detallado a continuación:

1. Barajar el conjunto de datos aleatoriamente
2. Dividir el conjunto de datos en k conjuntos
3. Para cada conjunto único:
   - a) Tomar el conjunto como un conjunto de datos de prueba o espera
   - b) Tomar los grupos restando como datos de entrenamiento
   - c) Ajustar el modelo con los datos de entrenamiento y evaluarlo con los datos de prueba
   - d) Mantener la puntuación de la evaluación
4. Resumir las atributos del modelo usando el conjunto de las puntuaciones de evaluación

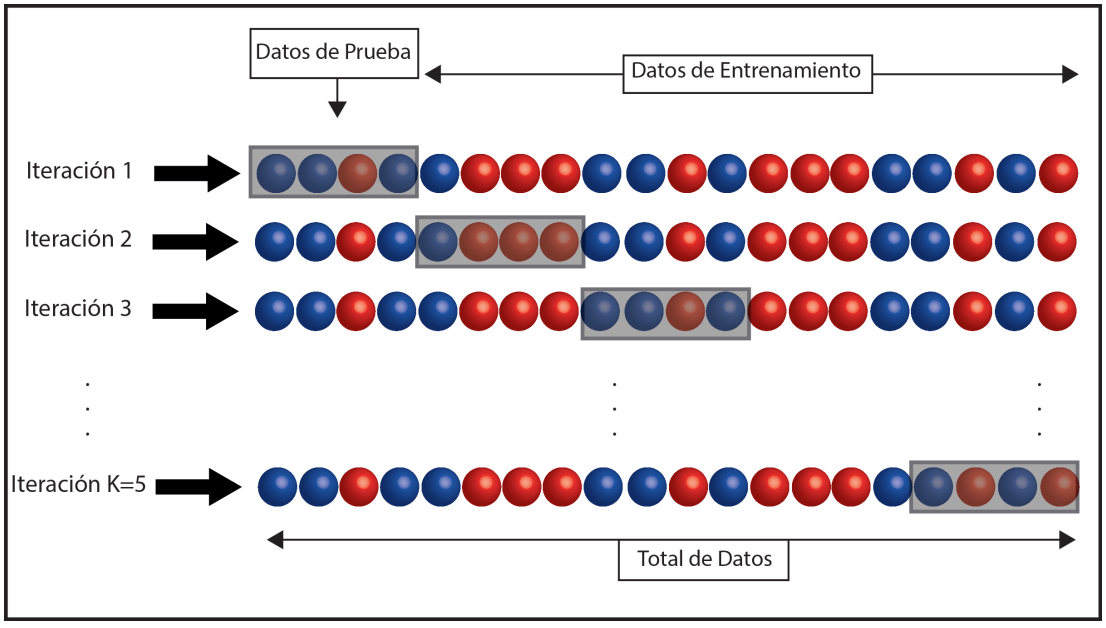

Figura 2.3: Muestra la metodología de validación cruzada. Se ejemplifica con K=5.

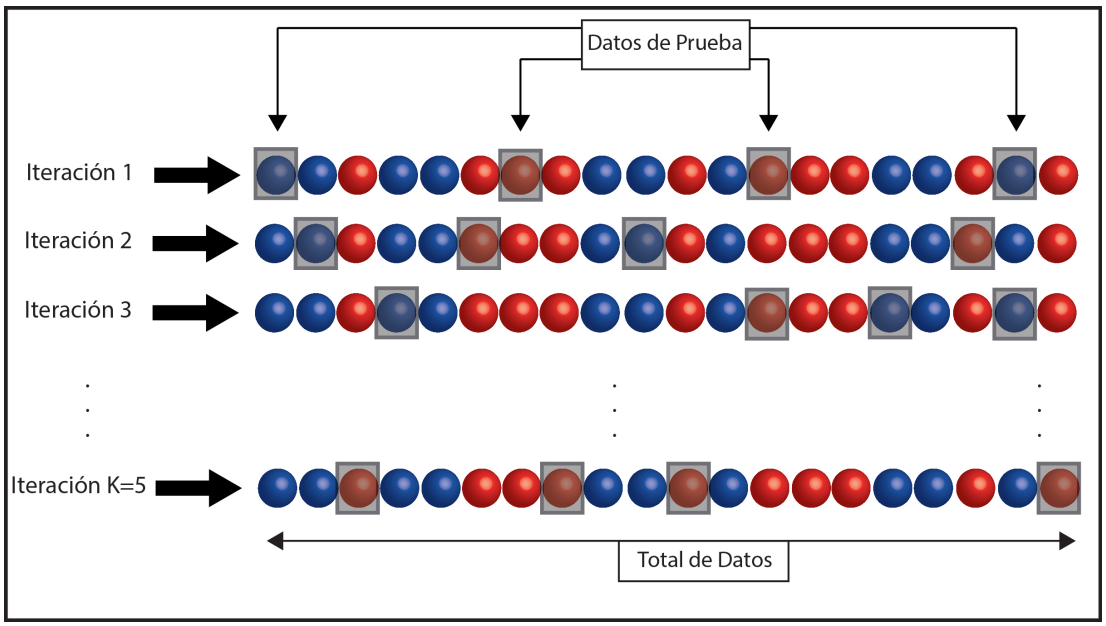

Figura 2.4: Muestra la metodología de validación cruzada eligiendo los elementos de los conjuntos aleatoriamente. Se ejemplifica con K=5.

Capítulo 3

Metodología

La metodología de resolución del problema fue abordada de tal forma que se pueda medir o tener una primera luz de qué tanto influye el audio para resolver el *video-to-text*.

Se espera aplicar el conocimiento adquirido en la resolución de este problema adaptando la arquitectura de [Dong, 2017]. Para ello se usa una arquitectura que aproveche la red neuronal pre-entrenada de SoundNet, usando su salida como descriptor de características. Por otro lado, se usará un *Bag of Words* (BoW) y *word2vec* para describir las oraciones de cada video. Así, se comparará ambas descripciones utilizando una Red Neuronal Siamesa, un modelo es ejemplificado en la figura 3.1. La Red Neuronal Siamesa está compuesta por dos redes hermanas de arquitectura equivalente conformada por dos capas simples cada una con una capa de normalización a la salida de éstas. Finalmente, se obtienen vectores en un espacio común a ser comparados usando la distancia entre estos vectores. Se realizan varios modelos con variaciones de estos componentes para evaluar cómo se comportan.

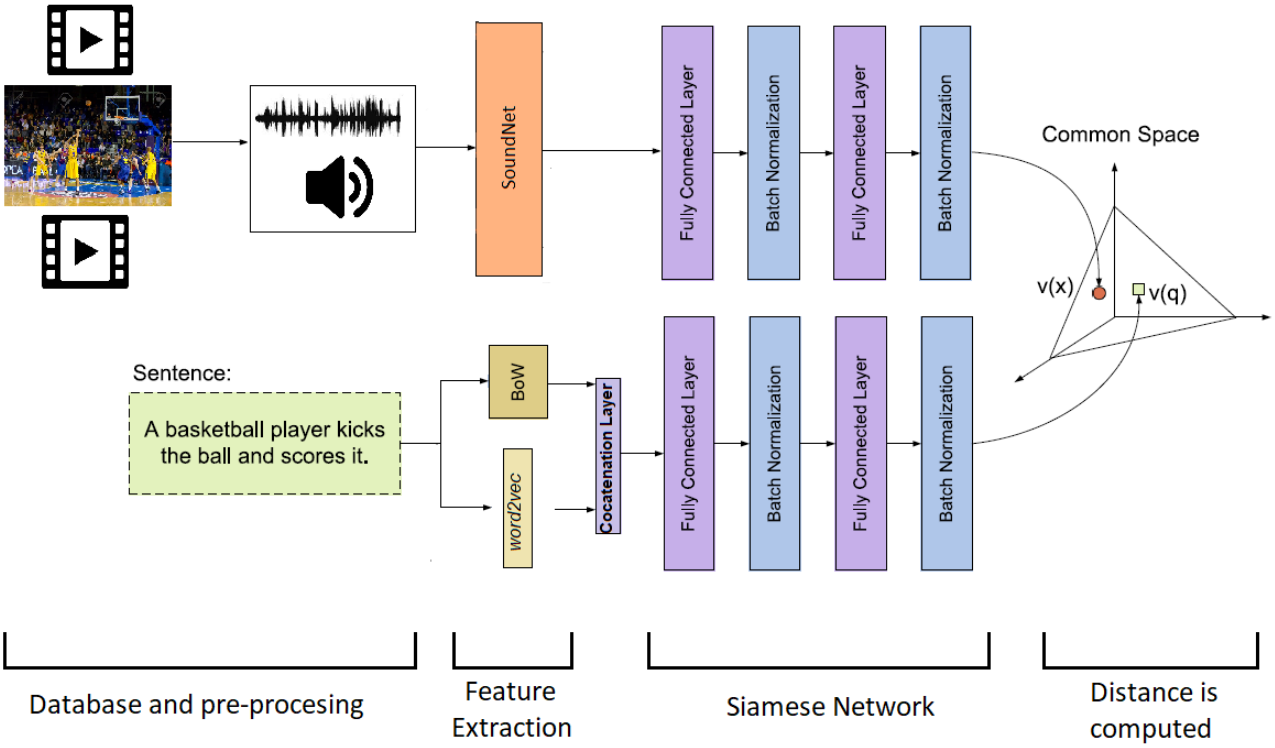

Figura 3.1: Arquitectura de que ejemplifica una solución problema. Basado en [Dong, 2017].

3.1. Base de Datos

La descarga de los videos fue hecha tomando los enlaces directamente de lo publicado en el sitio oficial de TRECVID 2016, 2017 y 2018. Se utilizo un bot a través de Python que automáticamente descarga los videos de los enlaces publicados en la base de datos. Estos videos poseen de una hasta cinco oraciones de descripción hechas por anotadores diferentes que serán usadas para desarrollar y validar este trabajo. Luego de descargados los videos se extrajo el contenido de audio y fue almacenado usando la librería FFmpeg contenida en la implementación de la librería MoviePY de Python.

Para el uso de los datos se considera lo siguiente, sea $x_{1,k}$ un audio multimedia el cual posee m descripciones $x_{2,k,1}$, ... , $x_{2,k,i}$, ... , $x_{2,k,m}$ se formaron m tuplas (audio,descripción) de la forma: $(x_{1,k}; x_{2,k,1})$, ... ,$(x_{1,k}; x_{2,k,i})$, ... , $(x_{1,k}; x_{2,k,m})$, que serán usadas para el desarrollo de la solución computacional al problema.

El conjunto de audios descargados tiene una cantidad de $5,431$ datos independientes, pero teniendo en cuenta el párrafo anterior la tupla de audio-descripción creció a $14,987$ datos, que serán

usados para desarrollar y validar la solución al problema (tabla 3.1 y 3.2).

Nombres	**Cantidad**
Audios	5,431
Descripciones	14,987
Ratio Descripciones/Audio	2.76

Tabla 3.1: Resumen de datos a utilizar.

Base de Datos	**Cantidad de Audios de Video**
2016	1,763
2017	1,773
2018	1,894

Tabla 3.2: Resume la cantidad de audios obtenidos por cada año.

Como pre-procesamiento, las oraciones de descripción de los videos/audios fueron convertidas de tal forma que sólo tengan minúsculas y se eliminaron caracteres especiales.

Si bien se usó el audio de los videos, para entender el *dataset* se muestran algunos *frame* o cuadros del videos asociado a la tupla audio-descripciones, la forma de la onda de audio asociada y la descripción (figura 3.2).

Figura 3.2: Muestra de base de datos: cuadro de video, forma de la onda de audio y oración de descripción para cuatro videos distintos.

Dentro de las observaciones con respecto a la base de datos, algunas direcciones de videos en la red Vine estaban rotos, es decir, no existían videos vinculados. De los videos descargados algunos no tenían audio, también, en algunos videos el audio no tiene relación con las imágenes del video, por ejemplo, una canción que acompaña a la secuencia de cuadros del video. Las oraciones describen el

contenido del video de los cuadros de imágenes no de los audios.

3.2. Representación

Por un lado, entonces, se tiene el audio proveniente de los video, por otro lado, se tiene las descripciones en la que cada una es una oración que describe al contenido del video. A continuación se explicará el tratamiento utilizado para obtener un vector de característica de ambos lados, este vector de características se refiere a un vector numérico que permita describir el objeto de interés, el proceso de creación es conocido como incrustación o representación.

3.2.1. Audio: SoundNet

SoundNet puede ser encontrada en el sitio oficial creado por sus desarrolladores para liberar su uso y su teoría). En su sitio se puede descargar el modelo pre-entrenados y se puede acceder al enlace que lleva a la nube de almacenamiento especializada en código computacional GitHub donde se encuentra una implementación en LUA. En este trabajo se extendió la implementación realizada por la Universidad de IMT Atlantique utilizando la librería Keras de Python agregándole la totalidad de la pesos de la red *SoundNet* a la carga parcial implementada por ellos.

Como se vio en la sección teórica el pre-procesamiento para el audio es mínimo, tan sólo configurar a un sólo canal (señal mono) y que la tasa de muestreo sea de $22,050$ hz.

La salida de la red de 8 capas SoundNet posee un largo de $1,401$. Esta salida está relacionada con ciertos objetos y categorías que se usaron para hacer pruebas. Esta red fue utilizada como una representación de el audio en un vector numérico.

Existen ciertas consideraciones propias en la arquitectura de SoundNet que generan peculiaridades, por ejemplo, que el vector de entrada no tenga que tener un largo específico o que se realicen convoluciones 1D. Esto determina que en la práctica al momento de predecir el vector salida por la red, éste dependa del largo del audio de entrada. De la siguiente forma, cuanto mayor sea la duración del audio o del vector de entrada, la red predecirá más vectores de salida esto vinculado al cálculo convolucional propio de la red descrito en el Marco Teórico en SoundNet. La forma de abordar esto fue promediar las salidas de los vectores componente a componente de forma que las k salidas de vectores de $1,401$ componentes son reducidas a 1 salida promediada.

[0]http://soundnet.csail.mit.edu
[0]https://github.com/cvondrick/soundnet

Otro aspecto a tener en mente, es que se necesita un largo mínimo de aproximadamente 5 segundos de audio para poder obtener una salida, si no posee este largo no arrojará ninguna salida. Muchos audios de los videos bordean esta característica necesaria para obtener una salida por lo que se debió idear una solución para evitar que audios de videos no tengan predicción. La solución utilizada fue calcular el vector salida por la red y en caso de no poder por no cumplir el largo mínimo, se clona el audio y se concatena o une al original de forma que se duplique el largo del audio sin cambiar su contenido sustancialmente. En caso de no ser suficiente se triplica, se cuadruplica o quintuplica, de forma que el cálculo arroje una salida. De esta forma cada audio de video tendrá su representación vectorial.

3.2.2. Oraciones: *Bag of Word* y *word2vec*

Se implementó el *Bag of Word* (BoW), para ello cada oración (que fue pre-procesada asegurando que esté en minúsculas y sin caracteres especiales) fue transformada a una lista de palabras, de esta lista de palabras fueron eliminadas las *stop words* o palabras vacías. Palabras vacías es el nombre que reciben las palabras sin significado como artículos, pronombres, preposiciones, etc. Luego, se obtiene el conjunto de palabras que componen el *corpus* que correspondería a todas las oraciones sin las palabras vacías. Este conjunto de palabras del *corpus* es ordenado asociado a la frecuencia con que aparecen, luego se toman las primeras $3,000$ palabras que se asociaran al vector que describirá las oraciones, donde tendrá valor 1 si dentro de las palabras de la oración a describir están o 0 si no están.
De esta forma, para el BoW se obtiene un vector de largo $3,000$ que tendrá valores de 0 o 1 en cada componente.

El uso de *word2vec* se hace usando la librería Gensim de Python, además, se usa un modelo ya pre-entrenado que para calcular palabras. El modelo utilizado es desarrollado por Google filial del conglomerado Alphabet Inc este modelo es entrenado para poder modelar 3 millones de palabras y oraciones . Al modelar una palabra, ésta es transformada a un espacio vectorial de 300 componentes. Se definió que el vector que representará a la oración será el promedio del vector transformado por el modelo de Google de cada palabra que compone la oración.

De esta manera, si se usan ambas descripciones concatenadas (BoW y *word2vec*). Estos dos vectores unidos generan vector combinado de $3,300$ componentes que se usaría como descriptor de

[0]https://code.google.com/archive/p/word2vec/

cada oración.

3.3. Red Siamesa

Las entradas para la Red Siamesa utilizada para la solución son los vectores de características obtenidos de la representación del audio y de la representación de las oraciones explicado en la sección Representación.

La función de pérdida utilizada es la *Triplet Loss* donde se usaron distintos valores del margen α (0.001, 0.01, 0.1, 0.2, 0.5, 2, 10, 100) para encontrar el que mejor se desempeña dado un mismo conjunto de entrenamiento y validación, para esto se usa un espacio común fijo de 175. Como las entradas de la red son de naturalezas distintas, las redes hermanas que computarían entradas que no serían la misma, éstas tuvieron distintos pesos para esta prueba de parámetros y aunque tienen la misma cantidad de capas, los tamaños de las capas, a excepción de la última, son diferentes (tablas 3.3 y 3.4). Notar que después de cada capa interna en ambas redes hermanas existe una normalización o *batch normalization*, para la dimensionalidad del espacio común final se emitirán diferentes resultados para diferentes dimensiones (50, 100, 175 y 250).

El modelo en que se testeará todo lo descrite es el modelo de red siamesa sin compartir pesos con *BoW* y *word2vec* para describir texto y la salida de *SoundNet* para describir el audio

Capa	**Dimensión**
Capa Simple	600
Capa Simple	400
Capa Simple de Salida (Espacio Común)	[50, 100, 175, 250]

Tabla 3.3: Esta tabla resume las capas de la red que procesa el audio. El listado en la salida representa los distintas arquitecturas en las cuales se presentarán resultados.

Capa	**Dimensión**
Capa Simple	1,000
Capa Simple	500
Capa Simple de Salida (Espacio Común)	[50, 100, 175, 250]

Tabla 3.4: Esta tabla resume las capas de la red que procesa las oraciones. El listado en la salida representa los distintas arquitecturas en las cuales se presentarán resultados.

3.4. Entrenamiento

3.4.1. Ancla, Ejemplo positivo y Ejemplo Negativo

Resumiendo, hasta el momento por lo que se ha explicado se tiene un vector característico de audio y su correspondiente vector característico de la oración que describe al video que posee el audio estudiado. Según se explicó en el capítulo anterior de Marco Teórico, para poder optimizar los pesos de la red siamesa con la función de pérdida *Triplet Loss*, se requiere que por cada par de datos positivos (que es lo que se tiene hasta el momento) se entregue una contraparte que no corresponda (ejemplo negativo).

En este punto, es prudente definir que el audio será el punto ancla que será comparado con la oración que lo describe (ejemplo positivo) y con una oración que no lo describa (ejemplo negativo). La oración que no lo describe o el ejemplo negativo será escogido dentro de los mismos datos. Aleatoriamente, se elegirá una oración del *dataset* teniendo cuidado que no sea una descripción del ancla a comparar, esto es similar al muestreo negativo de la teoría del *word2vec*. De esta forma, se obtiene un 3-tupla compuesta por el audio, la oración que lo describe y la oración que no lo describe.

3.4.2. Datos de entrenamiento y validación

Del total de datos se usarán unos $11,982$ datos que corresponden a un $80\,\%$ del total de datos para desarrollar el modelo y unos $2,996$ datos que corresponde al $20\,\%$ del total de datos para poder validarlo.

3.4.3. Configuración de Parámetros de entrenamiento

Como parámetros para entrenar la red que minimizará la función de pérdida *Triplet Loss*, 512 será el *batch size* o tamaño de conjunto antes de que el modelo es actualizado, se entrenará por 60 épocas para probar parámetros y por 200 épocas para la validación final y el algoritmo de optimización será *Adam*.

3.5. Validación

La validación se hace calculando la distancia entre el audio y las oraciones positivas de todo el conjunto de validación, para cada audio. Estas oraciones se ordenan con respecto a la distancia predicha en orden ascendente, esperando que el *match* o la oración correspondiente al audio tomado quede entre los primeros lugares sino el primero. De esta forma para cada audio de video se tendrá un *ranking* en el que estarán todas las oraciones. En dicho *ranking* en alguna posición está el

match del audio que en el caso idea, como se dijo, debe estar en la primera posición. La posición del *match* en el *ranking* es usada para validación, tomando el promedio de esta posición para la lista de posiciones del *match* obtenida de confeccionar el *ranking* para cada uno de los audio de video disponibles en el conjunto de validación (en el desafío de TRECVID se usa el inverso del la posición del *match*, a diferencia de este reporte).

Además, se usa la metodología de validación cruzada para esto la base de datos es dividida en 5 partes iguales usando en cada iteración un conjunto distinto de validación, esto para entender el desempeño más general de la solución.

Como línea base se usa un ordenamiento aleatorio de las oraciones para entender cómo se desempeña el modelo con respecto a un método aleatorio uniforme de *ranking*, se usan 5 secuencias aleatorias.

3.6. Modelos

Para validar la metodología tal que cumpla el objetivo planteado en este reporte, se entrenan varios modelos similares en arquitectura, variando componentes tales como el compartir o no pesos de la red siamesa y usar BoW en conjunto o no con*word2vec* para describir oraciones.

3.6.1. Modelo red siamesa sin compartir pesos usando BoW y *word2vec*

Primero, se prueba el modelo mostrado en la figura 3.3, donde en la red siamesa no se comparten pesos y la oraciones son descritas con la concatenación de BoW y *word2vec*.

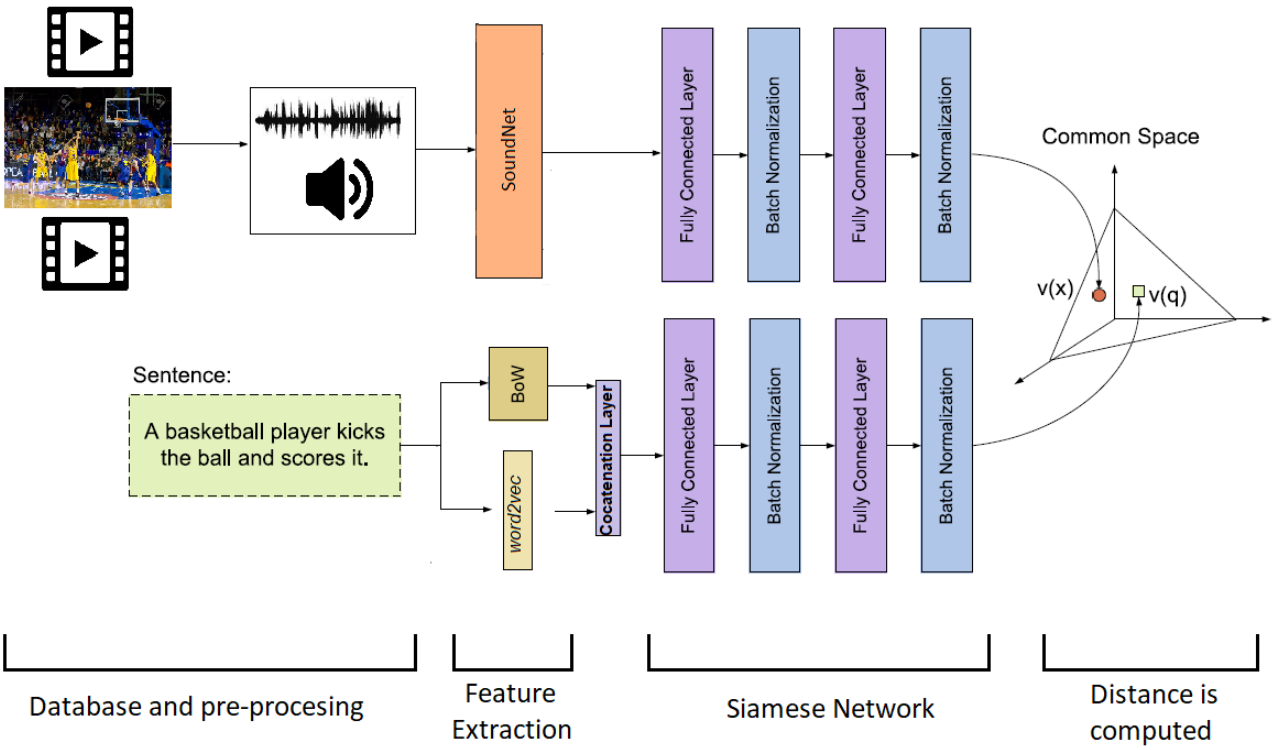

Figura 3.3: Arquitectura del modelo de red siamesa sin compartir pesos usando *BoW* y *word2vec*. Basado en [Dong, 2017].

Sobre este modelo se realizan las pruebas de parámetros de la arquitectura.

3.6.2. Modelo red siamesa compartiendo pesos usando *BoW* y *word2vec*

En este modelo alternativo se repite el mismo procedimiento para la representación de vectores del modelo anterior. Lo que diferencia es que en esta variación se comparten pesos en la red siamesa.

Para hacer esto posible, debido a la arquitectura de las componentes la red siamesa estructurados por capas simples, la dimensión de los vectores de entrada de audio y texto se ajusta para permitir compatibilidad con la única capa de entrada. La descripción del audio, dado por la salida de SoundNet, es de dimensión $1,401$. La descripción de las frases por otro lado está dada por el modelo de *word2vec* de dimensión 300 el que se une al vector de *Bag of Word* de dimensión $3,000$ asociado a la cantidad de palabras seleccionadas. Se reduce la cantidad de palabras del *Bag of Word* a $1,101$. De esta forma ambos vectores de representación poseen dimensión de $1,401$.

Como los vectores tienen la misma dimensión de la salida de SoundNet, la arquitectura de la red siamesa que comparte pesos será la misma que la elegida luego de hacer el procedimiento descrito

en este capitulo en la sección de Red Siamesa 3.3, la arquitectura será la misma usada para la rama de audio (tabla 3.3). Y los parámetros de la red serán los asociados a mejores resultados del modelo alternativo anterior.

3.6.3. Modelo red siamesa sin compartir pesos usando *BoW*

En este modelo no se comparten pesos en la red siamesa y se utiliza sólo *BoW* para describir la oración. En vector asociado al audio será procesado tal como se definió en secciones anteriores (3.3), la arquitectura para la red hermana que procesa vectores asociado a texto se presenta en la tabla 3.5. Y los parámetros de la red serán las asociadas a mejores resultados del primer modelo.

Capa	Dimensión
Capa Simple	1,000
Capa Simple	500
Capa Simple de Salida (Espacio Común)	resultado de la evaluación de parámetros

Tabla 3.5: Esta tabla resume las capas de la red que procesa las oraciones en el modelo sin compartir pesos usando *BoW*. La capa de salida será determinada en los resultados.

3.6.4. Modelo red siamesa sin compartir pesos usando *word2vec*

En este modelo no se comparten pesos en la red siamesa y se utiliza sólo *word2vec* para describir la oración. En vector asociado al audio será procesado tal como se definió en secciones anteriores (3.3), la arquitectura para la red hermana que procesa vectores asociado a texto se presenta en la tabla 3.6. Y los parámetros de la red serán las asociadas a mejores resultados del primer modelo.

Capa	Dimensión
Capa Simple	250
Capa Simple	200
Capa Simple de Salida (Espacio Común)	resultado de la evaluación de parámetros

Tabla 3.6: Esta tabla resume las capas de la red que procesa las oraciones en el modelo sin compartir pesos usando *word2vec*. La capa de salida será determinada en los resultados.

3.7. Variación en la descripción de oración de *word2vec*

Para entender el ruido que puede ocasionar hacer el promedio de las palabras de la oración a través del modelo de *word2vec*, se describe la oración seleccionando dos palabras centrales y promediandolas. Luego, se realiza el modelo descrito en la sección anterior cuando sólo se usa *word2vec* como descripción de oración (subsección 3.6.4).

3.8. Validación con datos de entrenamiento

Para poder vislumbrar la presencia de algún *bias* en los datos también se evalúa el el rendimiento con los mismos datos de entrenamiento para el modelo de red siamesa sin compartir pesos usando sólo *word2vec* para describir las oraciones.

La limitación en esta sección fue el tiempo de inversión y validación (36 horas), por lo que fue posible sólo realizar una validación.

Capítulo 4

Resultados

4.1. SoundNet: Uso

Para garantizar un correcto procesamiento de la información para solucionar el problema fue pertinente realizar el análisis de la Red Neuronal SoundNet, especialmente, teniendo en cuenta que el uso de esta fue programada en LUA y en este reporte se extendió un programa en Python (Keras) para su uso.

Para corroborar el uso de la red se procesó a través de ésta un audio de 20 s de duración. Desafortunadamente, no puede adjuntarse audio (aún) en los trabajos escritos, por lo que el audio fue descrito en detalle por el autor, las anotaciones están a continuación:

> El audio en la primera mitad tiene un sonido grave de campanas como señal de alarma, típicas en estaciones de tren que he visto en las películas cuando se acerca el tren. Superpuesto a esto, se escucha de fondo el sonido del viento que podría asociarse al acercamiento del tren. Este sonido va aumentando en volumen hasta el final del video donde se escucha el tren pasar. Esto se deduce por el sonido del viento y los sonidos de las ruedas pasando por las vías.

Ahora, el audio es procesado por la red obteniendo en la última capa un vector de $1,401$ componentes. Se obtuvieron 4 vectores de predicciones debido la duración del audio.
Se separó la última capa de la forma natural por la cual fue entrenada, 401 componentes de PlaceNet y $1,000$ componentes de ImageNet, luego en cada uno de los cuatro vectores se tomaron los 3 mayores valores y se vio que etiqueta tienen asociadas dichas componentes, además, se graficaron todas las componentes de los vectores para mejor visualización (tablas 4.1 y 4.2 y figuras 4.1 y 4.2).

Predicción	Etiquetas de los 3 máximos valores
1	1) railroad track 2) train station-platform 3) train railway
2	1) railroad track 2) train station-platform 3) train railway
3	1) railroad track 2) train station-platform 3) train railway
4	1) train station-platform 2) railroad track 3) train railway

Tabla 4.1: Muestras la etiquetas de Placenet de los tres máximo del vector.

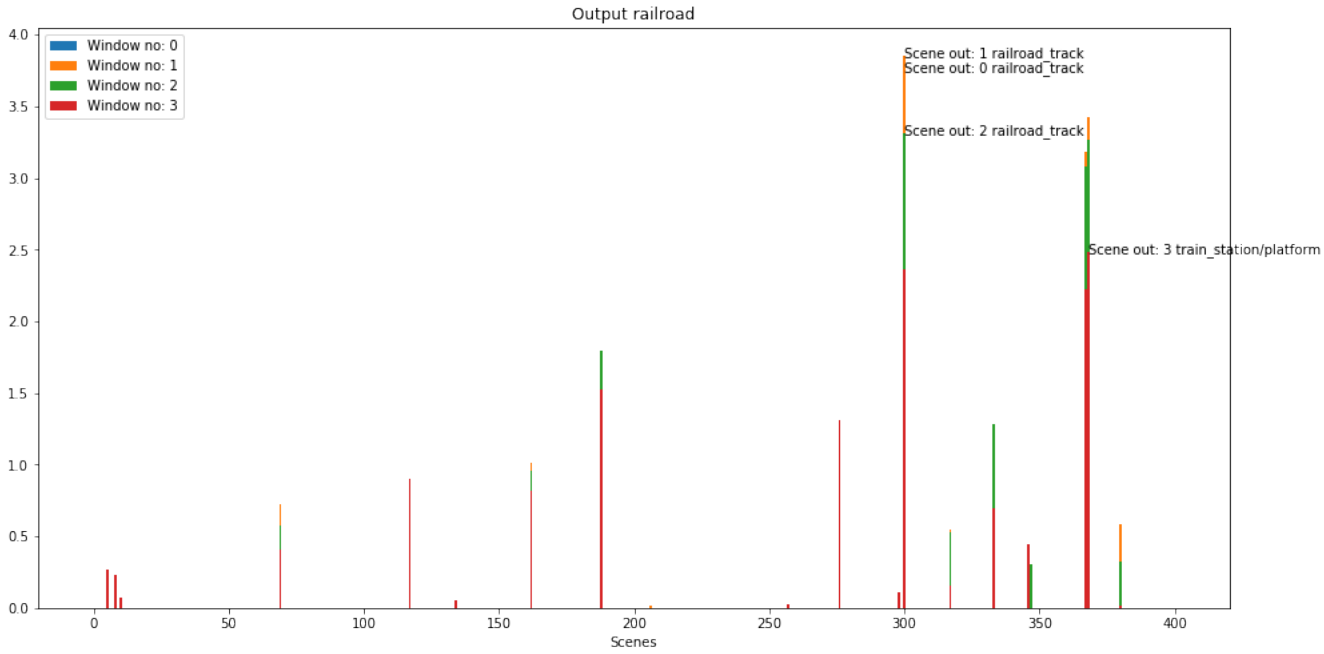

Figura 4.1: Muestra los valores del vector obtenido con la etiqueta en los valores máximos para cuatro vectores de predicción. Esto asociado al vector entrenado con Placenet.

Predicción	Etiquetas de los 3 máximos valores
1	1) traffic light, traffic signal, stoplight 2) freight car 3) streetcar, tram, tramcar, trolley, trolley car
2	1) passenger car, coach, carriage 2) electric locomotive 3) freight car
3	1) passenger car, coach, carriage 2) electric locomotive 3) freight car
4	1) passenger car, coach, carriage 2) bullet train, bullet 3) streetcar, tram, tramcar, trolley, trolley car

Tabla 4.2: Muestras la etiquetas de Imagenet de los tres máximo del vector.

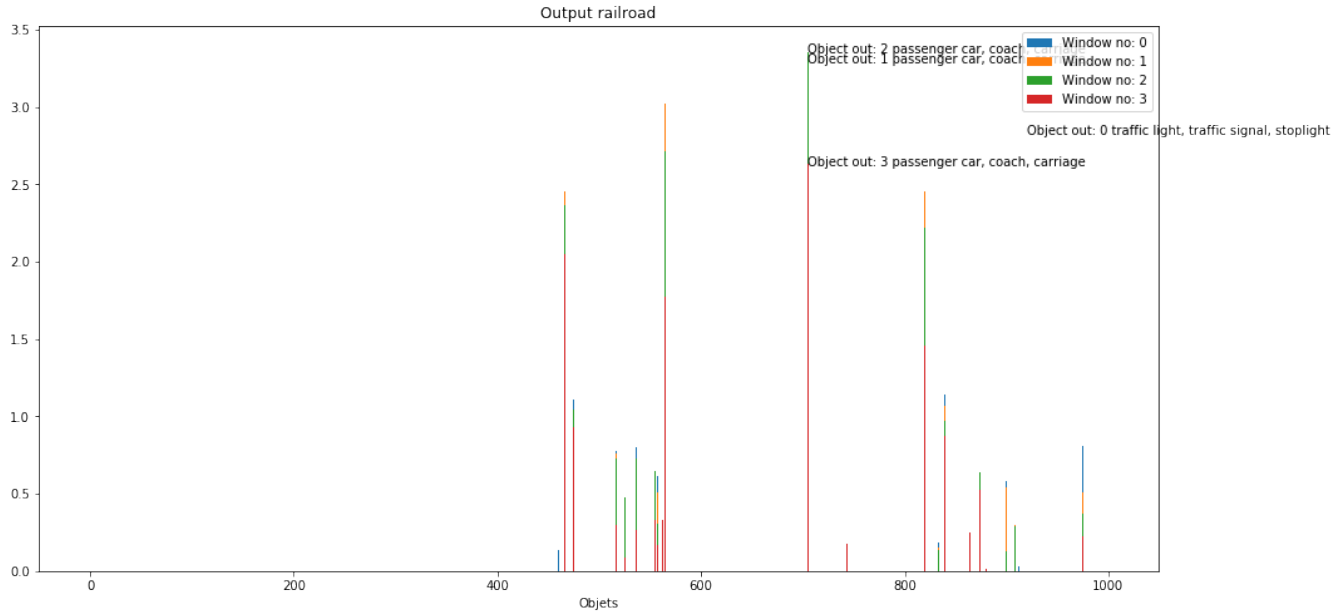

Figura 4.2: Muestra los valores del vector obtenido con la etiqueta en los valores máximos para cuatro vectores de predicción. Esto asociado al vector entrenado con Imagenet.

Se pudo probar que un audio de duración de 5 s (110, 250 muestras / 22, 050 muestras por segundos) no es capaz de generar una salida debido a su corta duración.

Estos resultados juntos con más pruebas hechas en diferentes audios controlados permitieron dar paso a programar la solución la cual se expondrá en las siguientes secciones.

4.2. Línea Base

Como línea base se confeccionaron *ranking* aleatorios y se obtuvo la posición promedio de los *match*. Esto se hizo 5 veces con 5 diferentes semillas de la secuencia aleatoria uniforme. La tabla 4.3 muestra los resultados y en promedio las frecuencias aleatorias resultaron en una posición promedio de 1, 499.

Ordenamiento Aleatorio	**Posición Promedio**
Secuencia Aleatoria 1	1,499
Secuencia Aleatoria 2	1,499
Secuencia Aleatoria 3	1,499
Secuencia Aleatoria 4	1,499
Secuencia Aleatoria 5	1,498

Tabla 4.3: Muestra los resultados obtenidos de validación para la posición del *match* en el *ranking* confeccionado por distintas secuencias aleatorias.

Como es un promedio en esta linea base aleatoria, distribuye como una distribución uniforme.

4.3. Modelo red siamesa sin compartir pesos usando *BoW* y *word2vec*

Este modelo de procesamiento de oraciones agregado es usado para evaluar diferentes parámetros.

4.3.1. Margen α *Triplet Loss*

La función de pérdida *Triplet Loss* posee un parámetro de margen α que da la distancia entre ejemplos positivos y negativos. Este parámetro es fijado por quien diseña la solución. Para elegirlo se tomó se fijo un conjunto de entrenamiento y validación y se desarrolló el modelo para distintos α midiendo la validación de cada uno. Recordemos la función de pérdida utilizada:

$$\mathcal{L}(A, P, N) = max\left(||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha, 0\right)$$

donde A es la entrada ancla, P es la entrada positiva de la misma clase que A, N es la entrada negativa de una diferente clase de A.

De esta forma, se prueba con 6 α distintos que son 0.0, 0.2, 0.4, 0.6, 0.8 y 1.0, la tabla 4.4 muestra la posición promedio de la oración *match* en los *ranking* para cada audio de video de validación. Finalmente, el margen escogido fue de 100.0.

Margen α	**Posición Promedio**
0.0001	1,381
0.001	1,369
0.1	1,367
0.2	1,363
0.5	1,382
1.0	1,363
2.0	1,364
10.0	1,368
100.0	1,335
500.0	1,359
1000.0	1,407
5000.0	1,392
10000.0	1,414
100000.0	1,424
1000000.0	1,428

Tabla 4.4: Muestra los resultados obtenidos de validación para la posición del *match* en el *ranking* para cada uno de los margen.

La figura 4.3 muestra la visualización de los datos obtenidos donde el eje X es transformado a un espacio logarítmico para ser visualizado mejor. La elección del parámetro α es basada en el

mínimo.

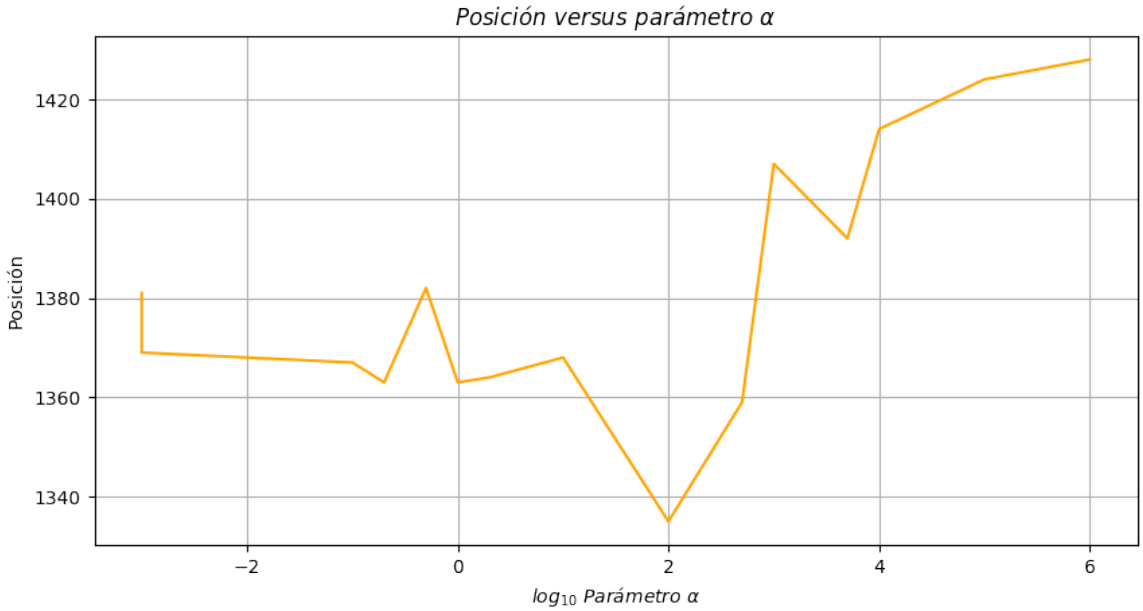

Figura 4.3: Muestra la posición promedio del modelo para distintos valores del parámetro alpha del modelo. Visualización del eje X en logaritmo en base 10.

4.3.2. Dimensión del Espacio Común Red Siamesa

Una vez computados el audio y las oraciones por el modelo a través de su transformación a un vector característico y luego su procesamiento por la red siamesa, las entradas son proyectadas a un espacio dimensional común de una misma dimensión donde es calculada la distancia Euclideana, por la que finalmente las oraciones ordenadas de acuerdo a la distancia y obtenido los *rankings*. Para un mismo conjunto de desarrollo y validación, el margen escogido en la sección anterior, se prueba con distintas dimensiones del espacio común que fueron 50, 100, 175 y 250. Esto se valida con la posición promedio de los *match* en los *rankings* de oraciones de los audios en el conjunto de validación. La tabla 4.5 muestra los resultados obtenidos y la dimensión escogida fue 175.

Dimensión	**Posición Promedio**
50	1,348
100	1,340
175	1,335
250	1,351

Tabla 4.5: Muestra los resultados obtenidos de validación para la posición del *match* en el *ranking* para cada uno de las dimensiones del espacio común probadas.

Los parámetros obtenidos en estas evaluaciones serán usados para esta y los siguientes modelos.

4.3.3. Validación Cruzada

Ya teniendo los parámetros de entrenamiento, se procede a obtener la posición final promedio de los *match* en los *rankings*. Son 5 conjuntos usados donde se desarrolla y entrena. Los resultados son mostrados en la tabla 4.6. La posición promedio de los 5 conjuntos es de $1,354$.

Validación Cruzada	Posición Promedio
Conjunto 1	1,343
Conjunto 2	1,335
Conjunto 3	1,365
Conjunto 4	1,371
Conjunto 5	1,369

Tabla 4.6: Muestra los resultados obtenidos de validación para la posición del *match* en el *ranking* para cada uno de los 5 conjuntos de validación cruzada.

Además, para entender la distribución de las posiciones del *match* se hizo un histograma de estas empaquetadas en un intervalo de 30 posiciones (1-30, 31-60, ...) mostrando la frecuencia en este intervalo (figura 4.4). En este histograma se puede observar de un patrón más bien uniforme, con la particularidad de tener un cúmulo de posiciones en los primeros intervalos.

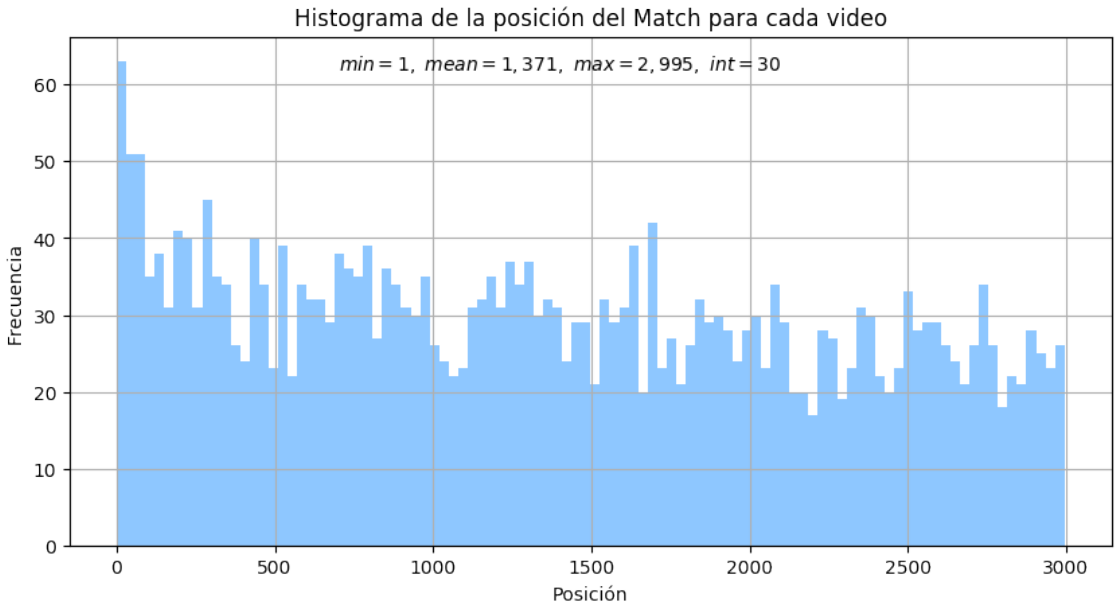

Figura 4.4: Muestra un histograma de las posiciones obtenidas para el Conjunto 4 de mayor *ranking* promedio.

4.4. Modelo red siamesa compartiendo pesos usando *BoW* y *word2vec*

Para este de modelo red siamesa que comparte pesos (vectores de entrada con misma dimensión). Como descriptor de oraciones se usa *BoW* y *word2vec* y usan los parámetros ya encontrados.

Se realiza una validación cruzada de 80 % de datos de entrenamiento y 20 % de datos de validación (5 iteraciones) los resultados se muestran en la tabla 4.7 donde el ranking promedio del *match* es 1, 496.

Validación Cruzada	**Posición Promedio**
Conjunto 1	1,493
Conjunto 2	1,497
Conjunto 3	1,497
Conjunto 4	1,495
Conjunto 5	1,496

Tabla 4.7: Muestra los resultados obtenidos de validación para la posición del *match* en el *ranking* para cada uno de los 5 conjuntos de validación cruzada en el modelo alternativo.

Además se muestra como distribuye la posición del *match* para cada video en 4.5.

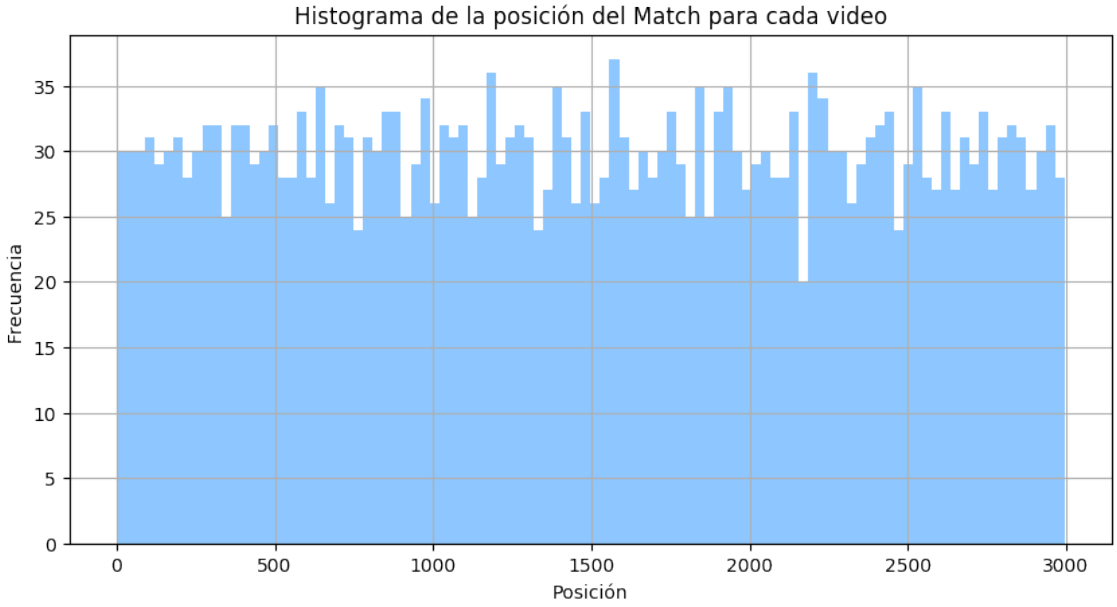

Figura 4.5: Muestra un histograma de las posiciones obtenidas para el Conjunto 5.

4.5. Modelo red siamesa sin compartir pesos usando *BoW*

Para este de modelo red siamesa que no comparte pesos. Las oraciones se describen solo con *BoW* y usan los parámetros ya encontrados.

Se realiza una validación cruzada de 80 % de datos de entrenamiento y 20 % de datos de validación (5 iteraciones) los resultados se muestran en la tabla 4.8 donde el ranking promedio del *match* es 1, 370.

Validación Cruzada	Posición Promedio
Conjunto 1	1,376
Conjunto 2	1,357
Conjunto 3	1,378
Conjunto 4	1,370
Conjunto 5	1,367

Tabla 4.8: Muestra los resultados obtenidos de validación para la posición del *match* en el *ranking* para cada uno de los 5 conjuntos de validación cruzada en el modelo usando *BoW*.

Además se muestra como distribuye la posición del *match* para cada video en 4.6.

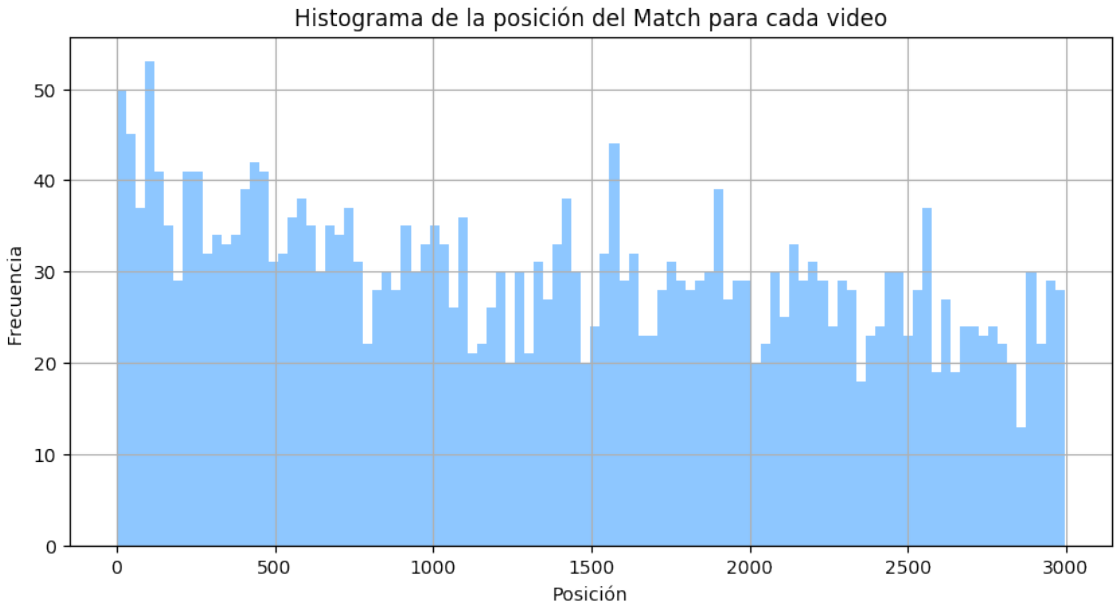

Figura 4.6: Muestra un histograma de las posiciones obtenidas para el Conjunto 1.

4.6. Modelo red siamesa sin compartir pesos usando *word2vec*

Para este de modelo red siamesa que no comparte pesos. Las oraciones se describen solo con *word2vec* y usan los parámetros ya encontrados.

Se realiza una validación cruzada de 80 % de datos de entrenamiento y 20 % de datos de validación (5 iteraciones) los resultados se muestran en la tabla 4.9 donde el ranking promedio del *match* es $1,282$.

Validación Cruzada	Posición Promedio
Conjunto 1	1,277
Conjunto 2	1,297
Conjunto 3	1,287
Conjunto 4	1,266
Conjunto 5	1,284

Tabla 4.9: Muestra los resultados obtenidos de validación para la posición del *match* en el *ranking* para cada uno de los 5 conjuntos de validación cruzada en el modelo usando *word2vec*.

Además se muestra como distribuye la posición del *match* para cada video en 4.7.

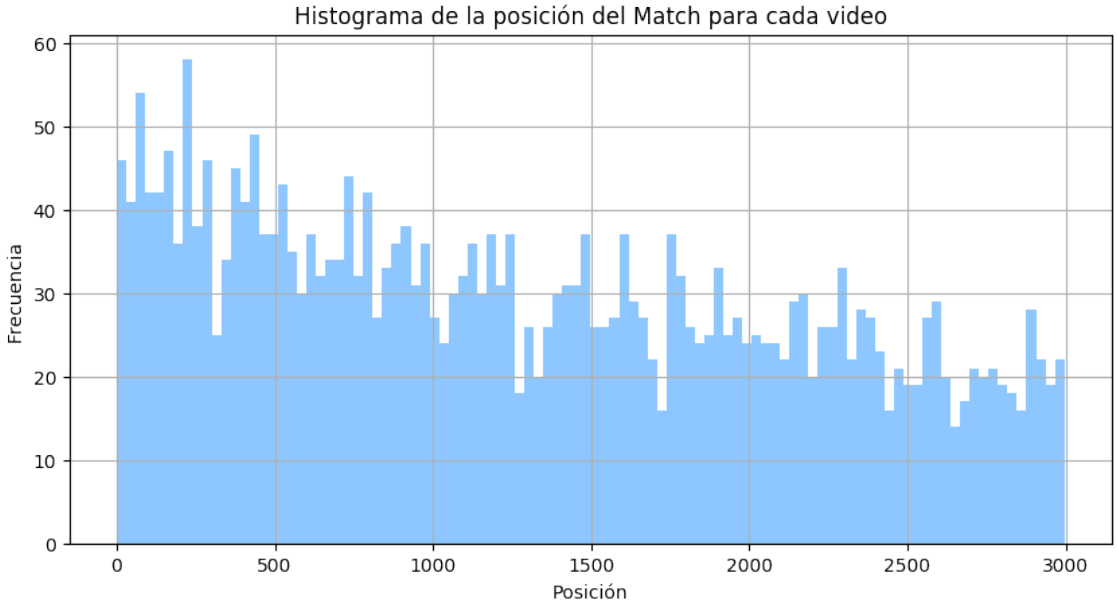

Figura 4.7: Muestra un histograma de las posiciones obtenidas para el Conjunto 5.

4.7. Modelo de oración sólo *word2vec* tomando palabras centrales

Para este de modelo red siamesa que no comparte pesos. Las oraciones se describen solo con *word2vec* tomando solo dos palabras centrales y usan los parámetros ya encontrados.

Se realiza una validación cruzada de 80 % de datos de entrenamiento y 20 % de datos de validación (5 iteraciones) los resultados se muestran en la tabla 4.10 donde el ranking promedio del *match* es $1,423$.

Validación Cruzada	Posición Promedio
Conjunto 1	1,435
Conjunto 2	1,422
Conjunto 3	1,408
Conjunto 4	1,428
Conjunto 5	1,415

Tabla 4.10: Muestra los resultados obtenidos de validación para la posición del *match* en el *ranking* para cada uno de los 5 conjuntos de validación cruzada en el modelo usando *word2vec* en palabras centrales.

Además se muestra como distribuye la posición del *match* para cada video en 4.8.

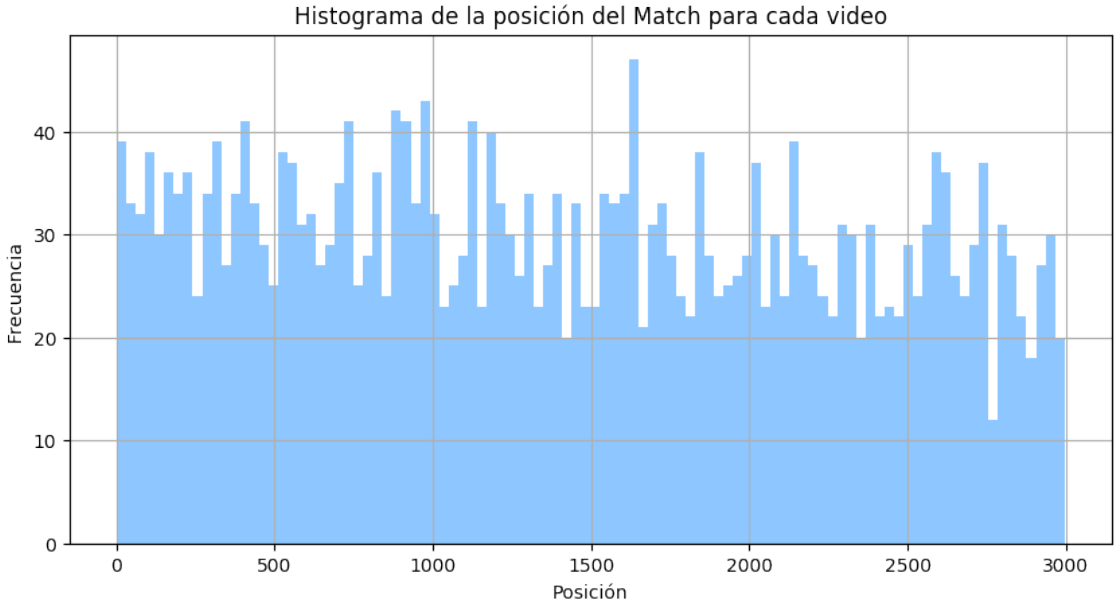

Figura 4.8: Muestra un histograma de las posiciones obtenidas para el Conjunto 2.

4.8. Validación con datos de entrenamiento

Para el Modelo sin compartir pesos usando sólo *word2vec* promediado palabra por palabra para describir las oraciones que fue el mas efectivo, se validan los datos de entrenamiento para entender si hay algún sesgo. Se hace sólo una validación con los datos de entrenamiento donde el ranking promedio de del match es $3,605$. Además se muestra como distribuye la posición del *match* para cada video en 4.9.

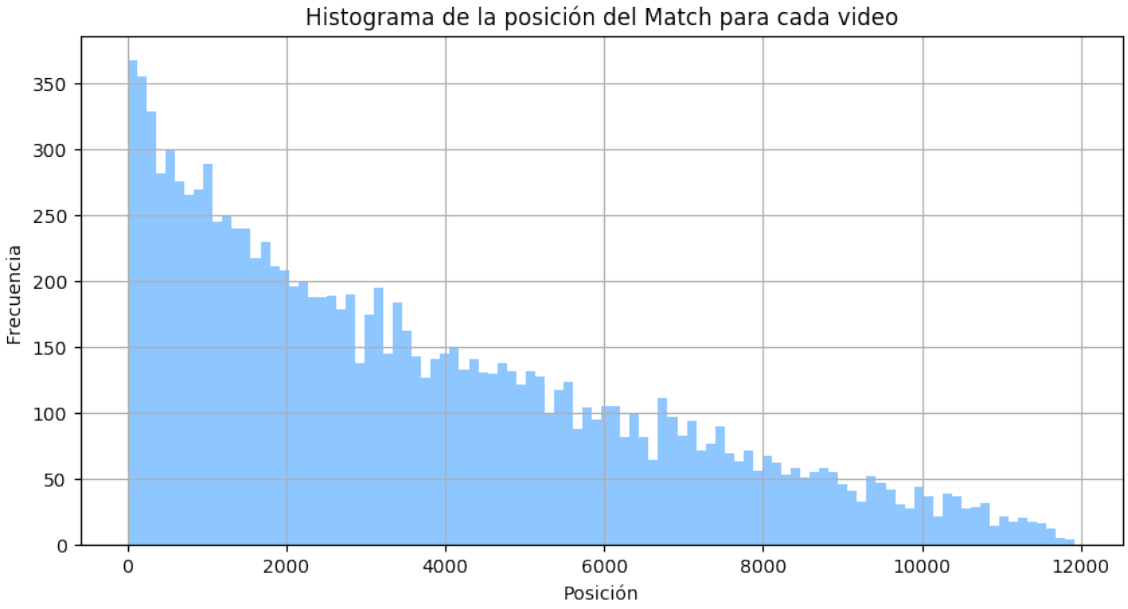

Figura 4.9: Muestra un histograma de las posiciones obtenidas para validación con datos de entrenamientos.

Capítulo 5

Discusiones

Antes que todo, es importante discutir una inquietud que puede haber surgido en el lector ingenioso: es lógico que el uso de sistemas de modelos pre-entrenados para obtener vectores característicos implica heredar también las debilidades de dichos modelos y pueden alejar a la implementación de resolver el problema. Este pensamiento sin duda tiene niveles de verdad, sin embargo, la metodología de este reporte fue basada en un ganador del certamen del 2016 [Dong, 2017], quien al detectar que se tienen un 1 % de datos necesarios para crear una red robusta, decide usar redes pre-entrenadas que le faciliten al modelo su aprendizaje en procesamiento de datos similares para que el uso de datos disponibles sea para que la red modele cuando video y frase se asemejan. Tener en cuenta los más de 9,000,000 de datos usados para entrenar *ResNet* (proyecto AI de imágenes) o los 2,000,000 de videos usados para entrenar *SoundNet* en contraste a los 5,431 audios distintos y 14,987 descripciones de este reporte. Sobre la misma dimensión discutida, resulta interesante en el audio poder realizar un entrenamiento sobre las capas de la red pre-entrenada, seguido de la arquitectura de proyección del vector al espacio común para comparar con las oraciones. Se esperaría, idealmente, que en la red pre-entrenada ocurriera un afinamiento fino del problema dedicado al audio mientras que la arquitectura siguiente concatenada se encargaría de modelar la función que proyectaría dicho audio al espacio común con las oraciones. Esto podría encapsular de cierta forma las funciones a modelar lo que podría simplificar la optimización de la inversión por lo que se espera que dicho procedimiento mejore los resultados. Dicho desarrollo e implementación se propone para trabajos futuros.

Una red neuronal, en una definición bastante amplia, es una función tipo transformada lineal entre dos espacios vectoriales. Donde, como norma general para datos multimedia, se busca reducir la cantidad de dimensiones en las que se encuentra el vector inicial, esto plasmado en la arquitectura de cada red neuronal. En términos simples, también, una red neuronal es una transformada de

reducción de dimensión. Como norma general, esa reducción de dimensión no es hecha con restricciones que impidan la pérdida de información. De ahí que el uso de redes neuronales pre-existentes tenga una pérdida de información, como regla general. Sin embargo, el uso de redes neuronales que se relacionan intuitivamente al problema (según un cerebro neuronal de un humano) es una forma de un mejor uso de los pocos datos disponibles, en contraste a usarlos para entrenar millones y millones de coeficientes desde cero.

Como análisis de los resultados, al no encontrar trabajos relevantes usando el audio de los videos en el desafío *video-to-text*, se creó una línea base con *rankings* aleatorios que debería ser muy fácil de «vencer» teniendo en cuenta que se están usando métodos inteligentes. Ahora, si bien la solución creada tiene mejores resultados que el sistema aleatorio, la solución propuesta es muy cercana a este resultado y muy lejana a un resultado que se pueda aproximar a una solución total del problema. Sumado a esto se discute sobre la posible resolución alternativa a los diferentes tamaños de los vectores de audio y texto con la aplicación de técnicas provenientes del desarrollo de redes convolucionales 2D. En dichas arquitecturas se suele usar capas GAP (*Global Average Pooling*) donde en cada mapa de características se toma un vector de forma de obtener un vector que reemplace la arquitectura de capas conectadas en redes neuronales del tipo convolucional. En una dimensión se aplicaría un promedio de todo el vector para así obtener una proyección en un espacio común. Esto, además, daría pie un abanico de posibles tratamientos directos de mapeo a los vectores que podrían variar los resultados obtenidos.

La lejanía de una resolución cabal como la obtenida en este trabajo está explicada por varios factores que se trataran de exponer a la luz en las siguientes líneas. Como se ha descrito en los capítulos anteriores, los audios de los videos de datos con que se desarrolló la solución no siempre tienen relación con el contenido del video, por ejemplo, puede ser una música como fondo de un video que es muy típico en videos cortos humorísticos de redes sociales como de la que se obtuvieron todos los videos. Otra característica de los audios, es que la señal de audio es nula para algunos videos. Además, en muchos casos las acciones efectuadas que se observan en los cuadro de los videos tiene un transfondo distinto a lo que se escucha en un nivel superficial, por ejemplo, dado un audio de una conversación en el video, si se vieran las imágenes se podría ver que realmente hay unos jóvenes haciendo morisquetas o podría ser una obra de teatro, cosas casi imposibles de deducir a través de la conversación escuchada. Todo esto que se ha expuesto, sumado a que los anotadores que describen los videos caracterizan el contenido de los videos basados en los cuadros de imágenes

más que el contenido de audio, explicarían los bajos resultados obtenidos resolviendo este problema usando sólo el audio de los videos.

Un factor en la construcción del modelo que puede explicar el resultado obtenido es la pérdida de orden de la estructura de la oración en los métodos de representación de oraciones BoW y *word2vec*, por la forma de realizar la representación se pierde el orden de las palabras de la oración que, por supuesto, podría influir en el significado de la oración; automáticamente se pierde muy cercano a una dimensión en el objeto oración que se está tratando de describir al perder el orden, sin duda, esto influye en la descripción generada. La descripción del audio, también, puede ser mejorada usando los conocimientos de inteligencia artificial en audio, incluso descriptores clásicos (ver la sección de Recomendaciones en el capítulo de Conclusiones). Prueba de esto es que el uso de *word2vec* sólo tiene mejores resultados que en conjunto con el *BoW*.Con respecto al uso de *word2vec* sobre otros descriptores contextuales similares como GloVe o FastText como se dijo, la idea de este reporte es replicar la solución ganadora del desafio del 2016([Dong, 2017]). En ese sentido es una posiblidad usar esas metodologías de descripción de palabras aunque su uso escapa los alcances de este reporte aplicado.

Con respecto a resultados obtenidos en soluciones del *video-to-text*, es posible contrastar y discutir algunos de los resultados obtenidos y a grandes rasgos la metodología usada en resoluciones del problema utilizando los cuadros de las imágenes. Se encuentra el uso vectores de representación extrayendo características útiles de ambos objetos resultando en que el *ranking* promedio del *match* sea 11 [Mark Marsden, 2016]. Se han usado Redes Neuronales complejas con procesamiento tipo GRU y CNN obtienendo resultados de *match* en posiciones 2 y 8 en el *ranking* en distintas ejecuciones [Niluthpol C. Mithun, 2017]. Tamibién, se han usado Redes Neuronales complejas con procesamiento tipo LSTM y se obtienen resultados de *match* en posiciones 3, 7, 10 y 552 en distintas ejecuciones [Youngsaeng Jin, 2018]. Por último, destaca, el uso Redes Neuronales complejas con procesamiento tipo GRU obteniendo resultados de *match* en posiciones 4 y 6 en distintas ejecuciones [Danny Francis, 2018].

Se puede apreciar los claros mejores resultados que se obtienen usando los cuadros de los videos. Era una evidencia de la alta probabilidad de este análisis en la descripción que da el NIST a este desafío introduciendo los avances que se podrían hacer en convertir a texto imágenes secuenciales, y es que el desafío fue construido para ellos, describiendo el contenido de las cuadros de los videos más

que de los audios, también, debido a los argumentos expuestos en más arriba en estas discusiones.

De la forma en que se realizó este trabajo, aunque los resultados están lejos de solucionar el problema, por sí solos, estos aportan algo de información para la solución. Observando la figura de los mejores resultados 5.1, que es un histograma que muestra la frecuencia de las posiciones del *match* para un intervalo de posiciones del *ranking*, se puede apreciar el cúmulo de altas posiciones para el primer intervalo. Por lo que el uso de la arquitectura utilizada para esta solución como complemento de una arquitectura que utilice las imágenes debería mejorar los resultados, especialmente, en los videos que está asociado a los primeros intervalos, en alguno de estos casos este complemento podría ser el voto de desición necesario para mejorar la calidad de la solución global. Resultados análogos se han obtenido al utilizar SoundNet como complemento para estudios con imágenes [Aytar et al., 2016].

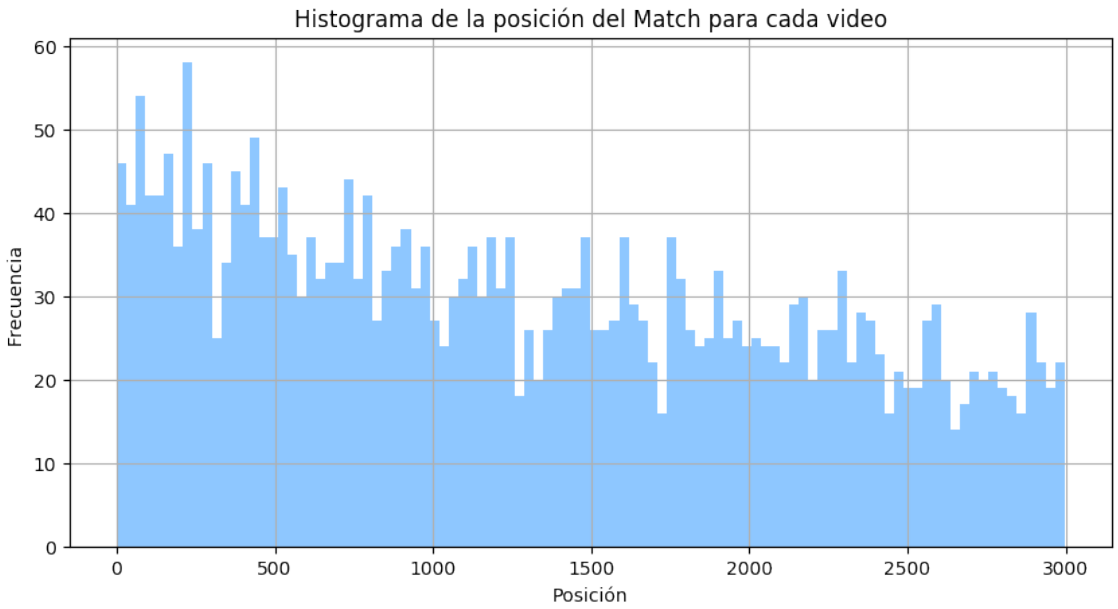

Figura 5.1: Muestra un la mejor validación obtenido en el modelo que no comparte pesos y usa sólo el promedio total de *word2vec* para describir oraciones.

Sobre el *overfitting* que pudo haber obtenido el modelo al momento de elección de los parámetros del modelo a utilizar, este es contrarrestado en la validación cruzada del modelo, por esta misma razón el histograma mostrado como referencia de la solución es hecho con el conjunto de mayor *ranking* promedio obtenido (peor rendimiento).

Con respecto al modelo en que en la red siamesa se comparte pesos, su resultado es cuasi-aleatorio lo que es esperable debido a que al compartir pesos en una red siamesa se está forzando

a que los vectores de entradas sean de la misma naturaleza, y en este caso no es así de hecho un subconjunto de la representación de las oraciones es *sparse*. Lo positivo de este resultado, es que de cierta forma valida el procedimiento puesto que se obtiene una respuesta esperada en el modelo (un peor resultado que tomando los pesos por separado).

En relación a los diferentes modelos que no comparten pesos, tienen un desempeño similar a excepción de la variación del *word2vec* en la oración que sólo toma palabras centrales. La validación con los datos de entrenamiento muestra que en el entrenamiento si hay un ajuste mejor esperado. Se destaca que se realizó un modelo con una extracción de las partes centrales de las oraciones para probar una cierta variación manual en la extracción de características que si bien no mejora la solución sólo se probó que dicha forma de realizarlo no funcionó. Se podrían probar otras ideaciones de este tipo: se propone que para un futuro se realice lectura a cada oración para poder ver el patrón humano en la descripción de los videos dadas las condiciones solicitadas, de esta forma se podría idear una metodología más personalizada que mejore el resultado. Con respecto al audio del video es ideal poder hacer pruebas similares, la limitante es que *SoundNet*, al realizar cálculos del tipo convolucional que reducen la dimensión del vector audio, este necesito un largo mínimo para poder generar una salida por lo que está fuera del alcance de este proyecto estudiar esta variante que podría mejorar el resultado final, donde se deben hacer modificaciones a la arquitectura de *SoundNet* cuidando no perder su efectividad.

Capítulo 6

Conclusiones

En este trabajo se pudo plantear una solución usando el audio de los videos al problema de VTT la cual obtuvo un desempeño mejor que una solución aleatoria pero muy lejos de resolver el problema. Se concluyó que:

1. Los resultados obtenidos sugieren que añadir análisis de audio constituye un aporte, sin embargo, es de marcada menor relevancia que los cuadros de imágenes de los videos. Esto contrastado con una línea base de *match and ranking* aleatorio.

2. Debido a las condiciones propias del desafío *video-to-text* de descripción de contenido de los cuadros de los videos y que no es necesariamente el mismo contenido del audio, por sí solo, el audio modelado en este trabajo no es capaz de resolver el problema planteado.

3. El regular desempeño abre la puerta a mejorar al modelo para tener mejores resultados.

4. Es posible comparar la similitud de dos objetos de naturaleza distinta utilizando Redes Neuronales Siamesas, herramienta de la rama de Inteligencia Artificial.

El aporte de trabajo es que si bien el audio, por si sólo, no es una buena característica a usar en la resolución del problema de VTT. Los resultados obtenidos sugieren que podría ser un buen complemento a usar en conjunto con las imágenes que podría ayudar a discernir entre una u otra descripción a asociar al video.

Recomendaciones

Dentro de recomendaciones para la mejora del modelo podemos encontrar.

1. Probar el funcionamiento de otra red neuronal para describir el audio, como la destacada AENet.

2. Probar el funcionamiento de descriptores clásicos de audio como MFCC para resolver el problema.

3. Probar el uso de métodos más complejos para representación de texto que no pierdan el orden de la oración, por ejemplo alguna red tipo GRU.

4. Probar complementar el modelo a una solución que use sólo los cuadros de las imágenes.

Bibliografia

[Arora et al., 2016] Arora, S., Liang, Y., and Ma, T. (2016). A simple but tough-to-beat baseline for sentence embeddings. In *5th International Conference on Learning Representations, ICLR 2017*.

[Aytar et al., 2016] Aytar, Y., Vondrick, C., and Torralba, A. (2016). Soundnet: Learning sound representations from unlabeled video. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 892–900.

[Bertin-Mahieux et al., 2011] Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. (2011). The million song dataset. In *In International Society for Music Information Retrieval Conference, 2011*.

[Bojanowski et al., 2017] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[Cakir et al., 2015] Cakir, E., Heittola, T., Huttunen, H., and Virtanen, T. (2015). Polyphonic sound event detection using multi label deep neural networks. In *2015 international joint conference on neural networks (IJCNN)*, pages 1–7. IEEE.

[Chechik et al., 2010] Chechik, G., Sharma, V., Shalit, U., and Bengio, S. (2010). Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(Mar):1109–1135.

[Cho et al., 2014] Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409 1259*.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412 3555*.

[Conneau et al., 2017] Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705 02364*.

[Danny Francis, 2018] Danny Francis, Benoit Huet, B. M. (2018). Eurecom participation in trecvid vtt 2018. In *VTT TRECVID Conference*.

[David Li and Touh, 2013] David Li, J. T. and Touh, D. (2013). Auditory scene classification using machine learning techniques. In *AASP Challenge 2013*.

[Dong and Shen, 2018] Dong, X. and Shen, J. (2018). Triplet loss in siamese network for object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 459–474.

[Dong, 2017] Dong, Huang, X. T. (2017). Video-to-text description. *TRECVid Conference, 13-15 Nov 2017*.

[Goldberg and Levy, 2014] Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al's negative-sampling word-embedding method. *arXiv preprint arXiv:1402 3722*.

[Guo et al., 2017] Guo, Q., Feng, W., Zhou, C., Huang, R., Wan, L., and Wang, S. (2017). Learning dynamic siamese network for visual object tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1763–1771.

[Gupta et al., 2016] Gupta, S., Hoffman, J., and Malik, J. (2016). Cross modal distillation for supervision transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2827–2836.

[Hadsell et al., 2006] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

[Hagan et al., 1996] Hagan, M., Demuth, H., Beale, M., and De Jesús, O. (1996). Neural network design vol 20: Pws pub.

[Hannun et al., 2014] Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., et al. (2014). Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412 5567*.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Hertel et al., 2016] Hertel, L., Phan, H., and Mertins, A. (2016). Comparing time and frequency domain for audio event recognition using deep learning. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3407–3411. IEEE.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Joulin et al., 2016] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607 01759*.

[Kiros et al., 2015] Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

[Klatt, 1987] Klatt, D. H. (1987). Review of text-to-speech conversion for english. *The Journal of the Acoustical Society of America*, 82(3):737–793.

[Knees and Schedl, 2016] Knees, P. and Schedl, M. (2016). *Music similarity and retrieval: An introduction to audio-and web-based strategies*, volume 36. Springer.

[Kohavi et al., 1995] Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Le and Mikolov, 2014] Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

[LeCun et al., 1995] LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Muller, U., Sackinger, E., et al. (1995). Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks*, volume 60, pages 53–60. Perth, Australia.

[Lee et al., 2009] Lee, H., Pham, P., Largman, Y., and Ng, A. Y. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104.

[Logan et al., 2000] Logan, B. et al. (2000). Mel frequency cepstral coefficients for music modeling. In *ISMIR*, volume 270, pages 1–11.

[Mark Marsden, 2016] Mark Marsden, Eva Mohedano, K. M. X. G.-i.-N. N. E. O. J. Z. L. A. T. D. B. D. M. H. H. A. J. D. D. G. W. L. A. W. A. F. S. (2016). Dublin city university and partners' participation in the ins and vtt tracks at trecvid 2016. In *VTT TRECVID*.

[McLoughlin et al., 2015] McLoughlin, I., Zhang, H., Xie, Z., Song, Y., and Xiao, W. (2015). Robust sound event classification using deep neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):540–552.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301 3781*.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Niluthpol C. Mithun, 2017] Niluthpol C. Mithun, Juncheng B Li, F. M.-A. K. R.-C. S. D. (2017). Cmu-ucr-bosch trecvid 2017: Video to text retrieval. In *Robert Bosch LLC, Research and Technology Center, USA. VTT TRECVID*.

[Piczak, 2015a] Piczak, K. J. (2015a). Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE.

[Piczak, 2015b] Piczak, K. J. (2015b). Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018. ACM.

[Rakotomamonjy and Gasso, 2015] Rakotomamonjy, A. and Gasso, G. (2015). Histogram of gradients of time–frequency representations for audio scene classification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(1):142–153.

[Roma et al., 2013] Roma, G., Nogueira, W., and Herrera, P. (2013). Recurrence quantification analysis features for environmental sound recognition. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4. IEEE.

[Salton et al., 1975] Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409 1556*.

[Stowell et al., 2015] Stowell, D., Giannoulis, D., Benetos, E., Lagrange, M., and Plumbley, M. D. (2015). Detection and classification of acoustic scenes and events. *IEEE Transactions on Multimedia*, 17(10):1733–1746.

[Subrahmanian, 1998] Subrahmanian, V. (1998). Principles of multimedia database systems. In *Morgan Kaufmann*, pages 53–98.

[Takahashi et al., 2017] Takahashi, N., Gygli, M., and Van Gool, L. (2017). Aenet: Learning deep audio features for video analysis. *IEEE Transactions on Multimedia*, 20(3):513–524.

[Van den Oord et al., 2013] Van den Oord, A., Dieleman, S., and Schrauwen, B. (2013). Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651.

[Werbos et al., 1990] Werbos, P. J. et al. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

[Wieting et al., 2015] Wieting, J., Bansal, M., Gimpel, K., and Livescu, K. (2015). Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511 08198*.

[Wolpert, 1992] Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, 5(2):241–259.

[Youngsaeng Jin, 2018] Youngsaeng Jin, Junggi Kwak, Y. L. J. Y.-H. K. (2018). Ku-ispl trecvid 2018 vtt model. In *Intelligent Signal Processing Laboratory, Korea University. VTT TRECVID*.

[Yu and Brandenburg, 2011] Yu, C. and Brandenburg, T. (2011). Multimedia database applications: Issues and concerns for classroom teaching. *arXiv preprint arXiv:1102 5769*.