Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Descripción automática de imágenes con comprensión lectora: hacia máquinas con percepción visual mejorada e interpretación de imágenes con texto.

PRESENTA:

Rafael Gallardo García

Tesis para obtener el grado de

Licenciado en Ingeniería en Ciencias de la Computación

Enero de 2022

Directora de Tesis: *Dra.* Beatriz Beltrán Martínez

Asesor de Tesis: *M.C* Rodolfo Martínez Torres

Resumen

El problema de la descripción automática de imágenes con comprensión lectora consiste en obtener descripciones textuales dada una imagen de entrada, con la condición de que dichas descripciones deben tomar en cuenta el texto presente en la escena visual. Para resolver este problema, es necesario que los sistemas sean capaces de leer, comprender y utilizar el texto leído, además de comprender la escena, los actores y los objetos involucrados en la misma, por último, el sistema debe ser capaz de integrar todo y describirlo en lenguaje natural.

Dada la complejidad del problema, los sistemas y propuestas disponibles en la literatura hacen uso de técnicas y modelos que resultan sumamente costosos, tanto en términos de memoria como de procesamiento. En este trabajo se proponen dos arquitecturas de aprendizaje profundo (L-M4C y L-CNMT) que son capaces de resolver el problema, obteniendo puntajes cercanos al estado-del-arte y superando a los diversos métodos disponibles en la literatura, todo esto mientras se reduce el uso de memoria (del módulo de lectura) hasta en un 94 %. La principal aportación se encuentra en el reemplazo del modelo de representación de texto, originalmente se utilizaban vectores FastText, mientras que las propuestas de este trabajo utilizan vectores GloVe.

La segunda aportación de este trabajo consiste en presentar la primera arquitectura bilingüe (ML M4C) para el problema de la descripción automática de imágenes con comprensión lectora. Dicha arquitectura es capaz de alcanzar puntajes cercanos al estado-del-arte en el idioma inglés, y, además, es capaz de resolver el problema para el idioma español. ML M4C está diseñada para resolver el problema hasta en 102 idiomas distintos, sin necesidad de realizar cambios mayores al diseño.

Dedicatoria

A Bibiana García y Rafael Gallardo, por ser unos excelentes padres y nunca dudar de mí. Han sido y serán siempre un soporte fundamental para completar mis metas.

¡Gracias por tanto!

A mi hermano, Diego Gallardo, por ser mi mejor amigo y un ejemplo de perseverancia y disciplina.

Nunca te rindas.

A Sofía Jarquín, porque sé que siempre puedo contar contigo, en las buenas y en las malas.

¡Te quiero!

A mis abuelos, Rosalba Legaspi y Carlos García, por enseñarme que ni nuestro origen ni las adversidades determinan nuestra calidad como personas.

¡Gracias!

A Mike, por ser, indudablemente, uno de los mejores amigos que he tenido. Su amor por la ciencia resultó sumamente contagioso, y lo agradezco.

Nos volveremos a ver.

Agradecimientos

A lo largo de mi vida académica y personal, he recibido una generosa cantidad de apoyo, asistencia y enseñanza. Quiero agradecer a todos aquellos que, con su guía y ejemplo, me enseñaron reglas de vida que van más allá de lo académico.

Quiero agradecer a mi asesora, Dra. Beatriz Beltrán, por ser mi guía durante el proceso de formación y aprendizaje. Le agradezco por brindarme su tiempo y conocimiento durante la mayor parte de mi licenciatura, también por ayudarme a preparar mi camino en la investigación, y por brindarme oportunidades a las que, de otra manera, no habría tenido acceso.

También agradezco al Dr. Carlos Hernández Gracidas y al Ing. Rodolfo Martínez Torres, así como a todos los docentes e investigadores que me apoyaron durante mi proceso de aprendizaje, retroalimentando y fortaleciendo mi trabajo de investigación y por ende, mis habilidades.

Al Laboratorio de Ingeniería del Lenguaje y del Conocimiento, por todas las oportunidades que se me otorgaron.

Gracias a la Benemérita Universidad Autónoma de Puebla por la oportunidad de llevar a cabo mis estudios de licenciatura en una institución de prestigio y excelencia.

Hay enseñanzas que toman años para comprender, pero que marcan significativamente el carácter de las personas. Quiero agradecer a Roberto Ruíz, por demostrarme que no hay límites para aquellos que están dispuestos a esforzarse y trabajar. Y al profesor Adolfo Beltrán, porque me enseñó que también hay que aprender a aprender.

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

En los humanos, la percepción es la forma en la que el cerebro interpreta las señales que recibe a través de los sentidos. En [Efron, 1969] se define como la forma primaria del ser humano para el contacto cognitivo con el mundo que le rodea. La importancia de la percepción como parte de la conciencia y el razonamiento ha sido resaltada en doctrinas filosóficas como el empirismo de Bacon o el aforismo de Locke: "No existe nada en la mente que no haya pasado antes por los sentidos" [Russell and Norvig, 2002]. En las máquinas funciona de forma similar, la percepción (dada a través de sensores) les proporciona información sobre el mundo que les rodea, la percepción computacional implica dotar a las máquinas de la capacidad de recibir estímulos de su entorno, de forma que sea capaz de procesarlos, entenderlos y, en algunos casos, de razonar con ellos.

Las máquinas pueden emular sentidos como el oído, el tacto y la visión, pero además, tienen acceso a formas de percepción que no son naturales para los humanos: ondas de radio, infrarrojos, posicionamiento global y señales inalámbricas. El enfoque principal de este trabajo se encuentra en la intersección, entre en la percepción visual (o visión por computadora) y el procesamiento del lenguaje natural.

Es claro que se han realizado avances significativos en diversas áreas de la

visión por computadora, sin embargo, la descripción de imágenes con lenguaje natural (conocido como "*image captioning*" en inglés) continúa siendo un problema extremadamente complejo, pues su solución está compuesta por una combinación de técnicas de visión computacional y procesamiento del lenguaje natural. Además, esta área cuenta con una gran variedad de problemas pendientes de resolver (e.g. adición de comprensión lectora, procesamiento de idiomas distintos al inglés, eliminación de sesgos en el texto generado, uso de relaciones complejas entre objetos, entre otros.)[Kumar and Goel, 2017]. La integración de comprensión lectora a los sistemas de subtitulado automático de imágenes se puede considerar dentro de esos problemas.

La descripción automática de imágenes tradicional (DAI) es un problema de la inteligencia artificial que se enfoca en generar descripciones en lenguaje natural, de prácticamente cualquier escena visual o imagen dada como entrada al modelo. Por otra parte, la descripción automática de imágenes con comprensión lectora (DAICL) es un reto relativamente reciente, pues el primer conjunto de datos enfocado a este problema vio la luz en el año 2020 [Sidorov et al., 2020]. La DAICL consiste en generar arquitecturas, sistemas o modelos inteligentes capaces de describir, en lenguaje natural, cualquier escena visual dada, con la complejidad añadida de que dicho sistema tiene que ser capaz de leer el texto presente en la escena, y, además, integrarlo de forma coherente en la descripción final. Una definición más completa del problema se puede encontrar a continuación.

1.1. Definición del problema

El problema se puede definir de manera formal como sigue: Encontrar una función f que, dada una imagen de entrada I_n sea capaz de generar una descripción en lenguaje natural de I_n, esto es, generar un conjunto de palabras $\{p_1, p_2, ..., p_m\}$ que describan objetos, acciones y características de la imagen de entrada, de forma coherente y respetando las reglas gramaticales del lenguaje objetivo (inglés y español para este trabajo). Lo anterior se puede formalizar como sigue:

$$f(I_n) = \{p_1, p_2, ..., p_m\}, \tag{1.1}$$

en donde la secuencia de palabras debe cumplir lo siguiente:

- El conjunto de palabras $\{p_1, p_2, ..., p_m\}$ debe ser coherente y respetar las reglas gramaticales del idioma objetivo, además de describir objetos, acciones, detalles o texto escrito, presentes en I_n.
- La función f será una arquitectura de aprendizaje profundo capaz de detectar entidades, acciones y texto en una escena, y describirlos en términos de lenguaje natural.

Como ya se ha mencionado, la adición de la capacidad de leer a los sistemas tradicionales de subtitulado de imagen implica retos técnicos y características importantes que la mayoría de propuestas en el área de la DAI tradicional no poseen, en [Sidorov et al., 2020] se plantean los siguientes:

- Determinar las relaciones entre los diferentes *tokens* obtenidos mediante Reconocimiento Óptico de Caracteres (OCR, por las siglas en inglés), así como las relaciones entre dichos *tokens* y el contexto visual de la imagen.
- Determinar que *tokens* OCR pueden añadirse juntos a la descripción, tomando en cuenta: semántica, relación espacial, apariencia visual y contexto.
- Alternar repetidamente entre las palabras del vocabulario y las palabras obtenidas a partir del OCR.
- Parafrasear e inferir sobre la utilidad y el significado de los *tokens* OCR en la descripción final.
- Aprender procesar palabras OCR nunca antes vistas (*zero-shot learning*).

La Figura 1.1 muestra tres ejemplos de como los modelos tradicionales de descripción de imagen carecen de la habilidad de leer e integrar el texto presente en escenas visuales, mientras que los humanos son perfectamente capaces de enriquecer la descripción utilizando el texto leído. Como se mencionó anteriormente, un modelo de descripción automática de imágenes con comprensión lectora debe ser capaz de leer y utilizar el texto leído, de forma que las descripciones generadas sean similares

Modelo tradicional: A box and a book are laying on a table.

Humano: A book about CEOs is sitting on top of a box.

Modelo tradicional: A group of birds in a sunset with the number 10 on it.

Humano: The music book cover with Sibelius Symphonies from Minnesota Orchestra.

Modelo tradicional: A big door at the entrance of the supermarket.

Humano: A set of grey double doors are under a sign that says Motor Kars Tire & Auto Care.

Figura 1.1: Tres escenas visuales y las descripciones generadas por un modelo tradicional comparadas con las generadas por un humano.

a aquellas generadas por los humanos. Lo anterior supone una gran variedad de retos técnicos adicionales.

Al momento de escribir el presente trabajo, únicamente se ha encontrado un conjunto de datos que fue específicamente diseñado para resolver el problema de la DAICL, y es llamado TextCaps (nombrado así por las palabras en inglés *Textual Captions*), mismo que fue presentado en [Sidorov et al., 2020] en el año 2020, como parte del trabajo de investigación de la división de inteligencia artificial de *Facebook Research*. Los detalles sobre este conjunto de datos serán profundizados en la sección 4.

Uno de los enfoques de este trabajo de tesis hace énfasis en el tamaño en memoria de los modelos diseñados para la DAICL, la mayoría de las arquitecturas del estado-del-arte no se enfocan en presentar alternativas que reduzcan de alguna manera el uso de memoria. Este trabajo de tesis tiene como objetivo presentar alternativas que reduzcan la cantidad de memoria utilizada por los modelos en las fases de entrenamiento e inferencia.

Aunque las arquitecturas del estado-del-arte que resuelven el problema de DAICL han reportado resultados cuantitativos impresionantes, todas están específicamente diseñadas para resolver el problema en el idioma inglés. Uno de los objetivos de este trabajo consiste en desarrollar propuestas que sean capaces de resolver el problema

de la DAICL desde un enfoque multi-lenguaje, además de que sus desempeños en el idioma inglés deberán ser comparables, al menos, con las métricas de la base reportada por los autores de TextCaps.

1.2. Objetivo general y objetivos específicos de la investigación

A continuación se pueden encontrar los objetivos del presente trabajo. Primero, se enuncia el objetivo general y, posteriormente, se plantean los objetivos específicos.

Objetivo general: Proponer arquitecturas de DAICL que tengan un desempeño comparable o superior a la base para el conjunto TextCaps, que sean reducidas en tamaño (cantidad de parámetros y uso de memoria) y, además, sean viables para ser utilizadas en idiomas distintos al inglés.

Objetivos específicos:

1. Realizar una exploración del estado del arte, principalmente de trabajos que hagan uso de aprendizaje profundo para solucionar el problema del subtitulado de imagen.

2. Analizar las arquitecturas del estado-del-arte para detectar los componentes que tienen incidencia en el objetivo general del trabajo.

3. Presentar la primera versión (sintética) en español del conjunto de datos TextCaps.

4. Proponer arquitecturas que, con mínimas variaciones en su diseño, puedan resolver el problema para los idiomas inglés y español.

5. Proponer arquitecturas reducidas en cantidad de parámetros y en uso de memoria.

6. Evaluar los productos de este trabajo con las métricas utilizadas en la literatura.

7. Comparar de manera cuantitativa los resultados reportados en la literatura y los resultados obtenidos en este trabajo.

1.3. Preguntas de investigación

Dado que las arquitecturas de DAICL actuales tienen limitantes, se establecen las siguientes preguntas de investigación:

1. ¿Es posible diseñar arquitecturas de DAICL, que obtengan resultados cuantitativamente similares o superiores al modelo base de TextCaps, y que además, sean menores en cantidad de parámetros entrenables?

2. ¿Es posible diseñar arquitecturas de DAICL, que obtengan resultados cuantitativamente similares o superiores al modelo base de TextCaps, y que además, puedan realizar entrenamiento e inferencia haciendo uso de una menor cantidad de memoria cuando se las compare con los métodos actuales?

3. ¿Es posible diseñar arquitecturas de DAICL, que obtengan resultados cuantitativamente similares o superiores al modelo base de TextCaps, y que además, sean viables para su uso en distintos idiomas?

1.4. Hipótesis

Con base en las preguntas de investigación propuestas anteriormente, se establecen las hipótesis siguientes:

1. Es posible diseñar arquitecturas de DAICL que obtengan un rendimiento comparable con la base de TextCaps mientras son menores en cantidad de parámetros, si se reemplazan aquellos módulos del diseño para los que se proponga un sustituto con igual o mejor rendimiento pero con menor cantidad de parámetros.

2. Es posible diseñar arquitecturas de DAICL que obtengan un rendimiento comparable con la base de TextCaps mientras son más ligeros en uso de memoria, si se reemplazan aquellos módulos del diseño para los que se proponga un sustituto con igual o mejor rendimiento pero siendo más ligero en uso de memoria.

3. Es posible diseñar arquitecturas de DAICL que obtengan un rendimiento comparable con la base de TextCaps (y que sean viables para su adaptación a nuevos idiomas), si se proponen componentes alternativos que extiendan la capacidad del diseño hacia nuevos idiomas, haciendo uso de técnicas como la transferencia de aprendizaje.

1.5. Justificación

Los humanos son capaces de entender completamente las escenas visuales que se les presentan y, frecuentemente, es necesario leer el texto asociado y comprenderlo en el contexto de la escena visual [Sidorov et al., 2020], de modo que ambas fuentes de información se complementen. Es importante desarrollar sistemas de subtitulado automático de imágenes más modernos, que tengan la capacidad de leer e integrar el texto leído en sus salidas, pues la adición de estas habilidades podría incrementar el rango de aplicación de este tipo de sistemas, además, esto podría permitir que los modelos de visión computacional tengan un entendimiento más profundo y completo de sus entornos.

Aunque las primeras soluciones para el subtitulado automático de imagen se publicaron la década pasada [Farhadi et al., 2010], la adición de la capacidad de leer y de comprender el texto leído es un problema nuevo: el primer conjunto de datos específico se publicó en el año 2020 [Sidorov et al., 2020], abriendo un panorama de oportunidades para nuevas investigaciones, tanto en ciencia básica como en ingeniería y aplicación.

Es importante mencionar que el subtitulado automático de imágenes "tradicional" ya era considerado como un problema intelectualmente desafiante [Amirian et al., 2019]. Aunque los modelos tradicionales han mejorado su desempeño y calidad en los últimos años, incluso los mejores métodos para subtitulado automático de imágenes fallan en reconocer e integrar o parafrasear el texto presente en la escena [Anderson et al., 2018, Chen et al., 2019, Huang et al., 2019, Sidorov et al., 2020], lo que se atribuye principalmente a que dichos modelos únicamente se enfocan en los objetos visuales al generar las descripciones.

La aplicación de los sistemas de descripción automática de imágenes (DAI) abarca una gran variedad de problemas como: indexado automático de imágenes, sistemas de visión por computadora de propósito general, descripción de escenas visuales o apoyo para personas con capacidades visuales diferentes. Además, las áreas de aplicación incluyen bio-medicina, comercio, tecnología militar, educación, búsqueda web y robótica, entre otras [Amirian et al., 2019, Hossain et al., 2019].

Los objetivos de este trabajo plantean las bases para el desarrollo de tecnologías multimodales (que trabajen tanto con visión como con lenguaje), que requieran de menos recursos computacionales y que puedan ser utilizadas en varios idiomas. El cumplimiento de dichos objetivos es sumamente necesario si se desea que este tipo de modelos resuelvan problemas en el mundo real.

1.6. Límites de la tesis

Los objetivos de la tesis buscan igualar o superar las métricas obtenidas por el modelo base de TextCaps, es posible que aun cumpliendo estos objetivos no se logre igualar a los otros métodos disponibles en la literatura. La meta de este trabajo se encuentra en innovar en áreas distintas de la DAICL, siendo los resultados cuantitativos un objetivo secundario.

Otra posible limitante del trabajo se encuentra en la disponibilidad de datos para el idioma español. El conjunto TextCaps fue únicamente publicado para el idioma inglés, y dado que la traducción manual del conjunto no es una opción viable, se tiene que recurrir a la traducción automática del conjunto, y aunque se buscará la mejor opción disponible para esta tarea, es altamente probable que el modelo de DAICL herede los errores del modelo de traducción automática.

Además, de la literatura revisada, el presente trabajo de tesis es el primero que busca atacar el problema en un idioma distinto al inglés, razón por la cual no existe una evaluación base contra la que se puedan comparar los resultados obtenidos por un modelo en español.

1.7. Estructura de la tesis

A continuación se describe la estructura general de este trabajo de investigación:

En el Capítulo 1 (Introducción) se da una definición formal y gráfica del problema y se introduce el objetivo general y los objetivos específicos. Posteriormente, se describen las preguntas de investigación y las hipótesis planteadas. Por último, se incluye la justificación y las limitantes de este trabajo de tesis, así como la presente descripción de la estructura.

El Capítulo 2 contiene el marco teórico de este trabajo de tesis. Dicho marco teórico contiene toda la información necesaria para comprender la investigación y los resultados obtenidos, así como las aportaciones del trabajo.

En el Capítulo 3 se describe la exploración de la literatura, así como todas las propuestas disponibles para resolver la DAICL, de igual manera, este capítulo contiene una tabla comparativa con los resultados cuantitativos obtenidos por los modelos del estado-del-arte.

Una descripción del conjunto de datos, las especificaciones de la metodología, el diseño de las arquitecturas organizadas según el objetivo a cumplir, el diseño de los experimentos y los detalles de los mismos se encuentran en el Capítulo 4.

Los resultados, su respectivo análisis y su comparación con las técnicas estado-del-arte se encuentran en el Capítulo 5. Por último, el Capítulo 6 contiene las conclusiones, aportaciones y el trabajo a futuro.

CAPÍTULO 2

Marco teórico

En esta sección se presenta la información necesaria para comprender el trabajo de investigación realizado. Se abordan desde conceptos generales del aprendizaje automático, hasta conceptos avanzados del aprendizaje profundo y sus aplicaciones a la visión computacional y el procesamiento del lenguaje natural.

2.1. Panorama general sobre el aprendizaje automático

Como se describe en [Goodfellow et al., 2016], un algoritmo de aprendizaje automático es un algoritmo con la capacidad de aprender de los datos. Otra definición es aquella dada por [Mitchell, 1997]: "Se dice que un programa de computadora aprende de la experiencia E con respecto a una cierta clase de tareas T y una medida de rendimiento P, si su rendimiento en las tareas T, medido por P, mejora con la experiencia E".

Los algoritmos de aprendizaje automático se pueden clasificar según distintos criterios, por ejemplo [Géron, 2019]:

1. Dependiendo de la forma en que aprenden, sea con supervisión humana o

sin ella, se clasifican en: supervisados, no-supervisados, semi-supervisados, autosupervisados o aprendizaje reforzado.

2. Dependiendo de su capacidad para aprender (o no) de forma incremental mientras funcionan: aprendizaje en línea o aprendizaje por lotes.

3. Dependiendo de su funcionamiento para trabajar con nuevas muestras, por ejemplo, si se compara la nueva muestra con las muestras conocidas o si el algoritmo busca patrones en los datos de entrenamiento y después construye un modelo predictivo: basados en instancias o basados en modelos.

En esta sección se incluyen los conceptos necesarios para comprender, de forma general, la mayoría de los términos utilizados en el área del aprendizaje automático.

2.1.1. Aprendizaje supervisado, no supervisado, semi supervisado, autosupervisado y reforzado

Esta es una de las clasificaciones más comunes de encontrar en la literatura, a continuación se definirá brevemente cada una de las categorías.

Aprendizaje supervisado: En este paradigma de aprendizaje, los modelos son "supervisados" a través de los datos, es decir, los datos de entrenamiento que se dan como entrada al algoritmo ya contienen las soluciones deseadas, comúnmente llamadas "etiquetas" [Géron, 2019]. Para entrenar este tipo de sistemas se necesita dar al modelo una cantidad suficiente de ejemplos, de modo que le sea posible predecir el comportamiento de los datos. Algunos ejemplos de algoritmos de aprendizaje supervisado se pueden encontrar a continuación:

- k-Vecinos más cercanos,
- regresión lineal,
- regresión logística,
- *support vector machines*,

- árboles de decisión y bosques aleatorios,

entre otros. Es importante mencionar que, dependiendo de su diseño, las redes neuronales se pueden utilizar en distintos paradigmas de aprendizaje, por lo que se pueden clasificar en todas las categorías antes mencionadas.

Aprendizaje no-supervisado: A diferencia del aprendizaje supervisado, en este caso los datos no contienen etiquetas, es decir, el algoritmo tiene que aprender o descubrir los patrones que caracterizan a los datos [Géron, 2019]. A continuación se incluyen algunos ejemplos de algoritmos que pertenecen a esta categoría:

- Agrupamiento: k-Means, Análisis Jerárquico de Grupos, Esperanza-Maximización, entre otros.
- Reducción de dimensionalidad: Análisis de Componentes Principales (ACP), ACP por kernel, embebido lineal local, entre otros.

Aprendizaje semi-supervisado: Los algoritmos semi-supervisados son capaces de trabajar con datos parcialmente etiquetados, es decir, que los datos de entrenamiento pueden contener muestras con etiquetas o sin ellas, comúnmente, la cantidad de datos sin etiquetar es mayor que la de los datos etiquetados [Géron, 2019]. El diseño de estos algoritmos normalmente consiste en combinaciones de algoritmos de aprendizaje supervisado y no-supervisado.

Aprendizaje autosupervisado: El aprendizaje autosupervisado es uno de los campos más prometedores para entrenar sistemas que resuelvan con alta efectividad los problemas del mundo real. La importancia de este concepto radica en su capacidad para entrenar con cantidades inmensas de datos no etiquetados, transformándolos en información utilizable según el propósito para el que se diseñen. Algunos de los resultados más impactantes en el área del procesamiento del lenguaje natural basado en aprendizaje profundo se han dado en el entorno del aprendizaje autosupervisado, tal como sucedió con *Bidirectional Encoder Representations from Transformers* (BERT) [Devlin et al., 2018], GPT-1,2,3 [Radford et al., 2018, Radford et al., 2019], o *RoBERTa: A Robustly Optimized BERT Pretraining Approach* [Liu et al., 2019].

Actualmente, una de las arquitecturas de aprendizaje profundo más conocidas y que realiza aprendizaje autosupervisado es llamada *Transformer* [Vaswani et al., 2017].

Desde un punto de vista, el aprendizaje autosupervisado puede considerarse aprendizaje no-supervisado, puesto que no depende de las etiquetas humanas para generar sus objetivos, en este caso, es el mismo algoritmo quien etiqueta, categoriza o analiza la información que recibe, según la tarea que se busque resolver. Sin embargo, el aprendizaje autosupervisado también podría considerarse una forma "autónoma" de aprendizaje supervisado, pues, aunque no necesita datos etiquetados, la tarea a resolver comúnmente implica generar o predecir una salida etiquetada por el propio algoritmo, dada una determinada entrada. Dadas las razones anteriores, para este trabajo, el aprendizaje supervisado se considera como una categoría aparte.

Aprendizaje reforzado: El aprendizaje por refuerzo funciona de manera muy distinta a todas las categorías mencionadas anteriormente. En este caso, el sistema de aprendizaje está compuesto, principalmente, por un *agente* y un *entorno* [Géron, 2019]. El agente tiene la capacidad de observar o actuar sobre el entorno y recibir recompensas positivas o negativas. El aprendizaje del agente radica en la búsqueda de la mejor estrategia, de forma que las recompensas se maximicen o minimicen, según sea el caso.

2.1.2. Aprendizaje en línea y aprendizaje por lotes

Este criterio de clasificación se basa en la capacidad de los algoritmos para aprender de forma incremental dado un flujo continuo de datos.

Aprendizaje por lotes: En el aprendizaje por lotes, el algoritmo es incapaz de aprender de forma incremental [Géron, 2019], lo que implica que el sistema tiene que ser entrenado utilizando todos los datos de entrenamiento disponibles. Una vez que el sistema es entrenado con todos los datos de entrenamiento, se puede proceder a la etapa conocida como "inferencia", en donde se pueden procesar puntos o muestras nunca antes vistos. Sin embargo, el sistema únicamente utilizará conocimiento ya aprendido, si se deseara incrementar el conocimiento, se necesitaría volver a entrenar

el sistema desde cero, incluyendo los datos antiguos y los nuevos. A este tipo de aprendizaje también se le conoce como *offline*.

Aprendizaje en línea: Para aquellos entornos en donde el aprendizaje por lotes no resulta eficiente, el aprendizaje en línea es una alternativa. En este tipo de sistemas, el entrenamiento se puede llevar a cabo de forma incremental, mediante el procesamiento secuencial de muestras, ya sea de forma individual o en pequeños grupos llamados "mini-lotes". A este tipo de aprendizaje también se le conoce como aprendizaje incremental.

2.1.3. Aprendizaje basado en instancias y aprendizaje basado en modelos

Para explicar este criterio de clasificación, es necesario primero definir el término de "generalización". El problema principal en el aprendizaje automático radica en el desempeño del algoritmo en muestras nunca antes vistas, la generalización es la habilidad de un algoritmo de tener un buen rendimiento en dichas muestras [Goodfellow et al., 2016]. Para clasificar algoritmos según este criterio, es necesario revisar la manera en que generaliza un modelo.

Aprendizaje basado en instancias: Los algoritmos que pertenecen a esta categoría procesan nuevas muestras mediante la comparación directa con muestras disponibles en los datos de entrenamiento, normalmente dicha comparación se lleva a cabo con alguna métrica de similitud o distancia. Algunos ejemplos de este tipo de algoritmos son:

- k-Vecinos más cercanos,
- Mapas autoorganizados,
- Aprendizaje pesado localmente,

entre otros.

Aprendizaje basado en modelos: Estos son los algoritmos que construyen modelos a partir de los datos de entrenamiento, es decir, el algoritmo generaliza a través de un modelo (también llamada etapa de entrenamiento), que después es utilizado para realizar predicciones (o etapa de inferencia). Un modelo es el resultado de ejecutar un algoritmo de aprendizaje sobre los datos de entrenamiento, el modelo representa el aprendizaje obtenido por el algoritmo.

Retomando la definición de [Mitchell, 1997], un modelo M puede entenderse como un contenedor, en donde se ven almacenados todos los resultados, reglas o patrones aprendidos por el algoritmo a partir de E, de modo que la tarea T se vea resuelta de la mejor manera posible para M según P. Una vez M se encuentra entrenado, es posible resolver T para muestras nunca antes vistas, y su rendimiento dependerá de la calidad de la generalización.

En este trabajo de investigación únicamente se hace uso de algoritmos con aprendizaje basado en modelos. A partir de ahora, todos los conceptos descritos son aplicables a esta categoría.

2.2. Conceptos generales sobre el aprendizaje profundo

Originalmente, el campo de la inteligencia artificial se enfocó en resolver problemas que eran intelectualmente complejos para los humanos (e.g. formulación o demostración de reglas matemáticas formales), y aun así, resultaron ser relativamente fáciles para las computadoras [Russell and Norvig, 2002]. Irónicamente, las tareas que, para los humanos resultan simples, han supuesto grandes dificultades para las computadoras. Aquellos problemas que resultan complejos de representar de manera formal son, generalmente, los más complejos para resolver, como por ejemplo, reconocer simples palabras, rostros o formas.

Era necesario trabajar con algoritmos que fueran capaces de resolver estos problemas "simples". El aprendizaje profundo es una forma de dotar a una computadora de la capacidad de aprender de la experiencia, y de entender el mundo de una manera en donde los conceptos se relacionan de forma jerárquica, siendo cada

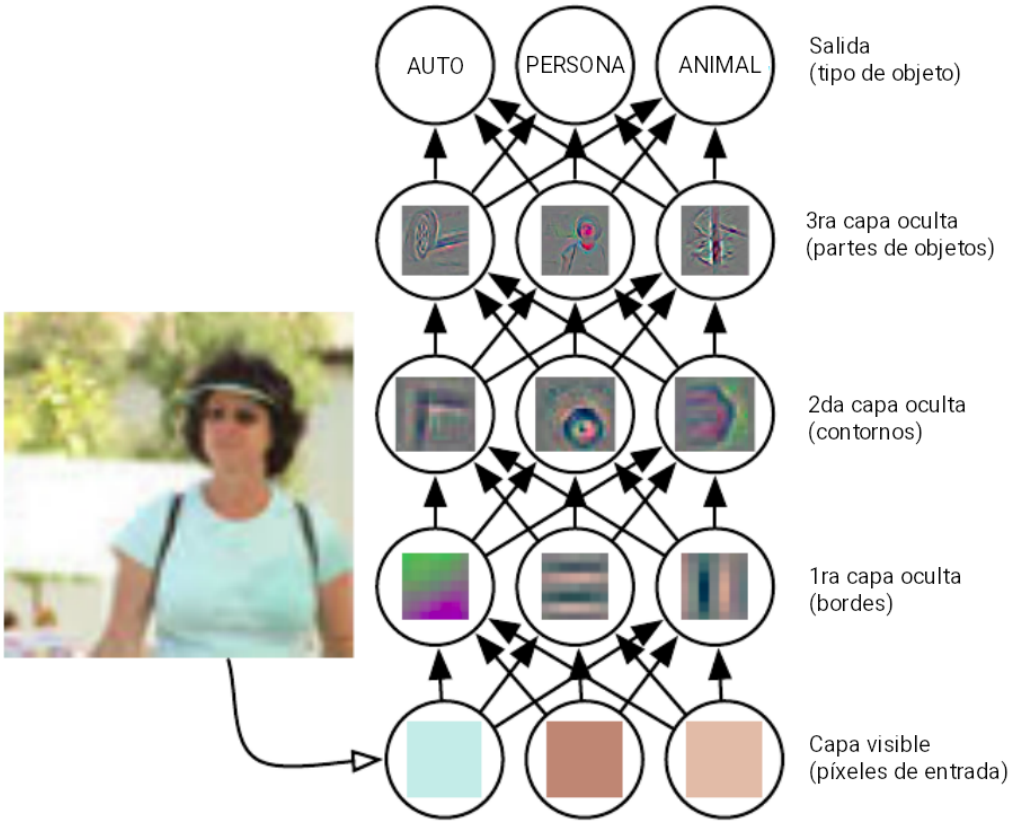


Figura 2.1: Ilustración de un modelo de aprendizaje profundo. Las capas más profundas reconstruyen las entradas con una representación más abstracta. Figura adaptada de [Goodfellow et al., 2016].

concepto "complejo" definido como una combinación o relación de conceptos más simples [Goodfellow et al., 2016]. Esta jerarquía permite a la computadora aprender conceptos complicados, construyéndolos a partir de alternativas más simples.

Como se menciona en [Goodfellow et al., 2016], el aprendizaje profundo busca mejores formas de representar el conocimiento que recibe como entrada, mediante la introducción de representaciones que se dan en términos de otras más simples. La Figura 2.1 muestra un ejemplo gráfico de lo descrito anteriormente, cuanto más profunda es la red (más capas tiene), más específicos o abstractos son los conceptos con los que construye la representación de la entrada.

Prácticamente, todos los algoritmos de aprendizaje profundo pueden ser descritos como una combinación de: un conjunto de datos, una función de coste, un procedimiento de optimización y un modelo [Goodfellow et al., 2016]. La configuración de los componentes descritos anteriormente puede variar enormemente entre un modelo

y otro, adicionalmente, incluso un modelo que usa los mismos datos, funciones de coste y optimización variará según los hiperparámetros con los que se ejecute.

2.2.1. Hiperparámetros

La mayoría de algoritmos de aprendizaje automático se pueden configurar de distintas maneras haciendo uso de variables que controlan el comportamiento de dicho algoritmo, a estas variables se les conoce como "hiperparámetros". Los hiperparámetros son variables configurados que no son ajustados por el algoritmo de aprendizaje (aunque existen técnicas como el *Auto Machine Learning* que realizan este tipo de ajustes).

Los hiperparámetros de un modelo pueden ajustar el procedimiento de optimización (e.g. tasa de aprendizaje, tamaño de lote, épocas de entrenamiento, etcétera) o, ajustar directamente las características del modelo (e.g. cantidad de capas, cantidad de neuronas, tipos de funciones, etcétera).

Hiperparámetros versus parámetros del modelo: Es importante mencionar que estos términos, aunque similares, hacen referencia a dos partes muy distintas del modelo. Los *hiperparámetros* son variables externas, que controlan el comportamiento del modelo, mientras que los *parámetros* son variables internas, que son ajustadas durante el entrenamiento y que tienen que ver directamente con la capacidad del modelo. En aprendizaje profundo es común llamar "pesos" a los parámetros de un modelo.

2.2.2. Conjuntos de entrenamiento, validación y prueba

Si se habla de procesamiento de datos, una de las mejores prácticas radica en la creación de subconjuntos, de modo que cada subconjunto pueda ayudar a diferenciar el impacto del procesamiento. En el área de aprendizaje profundo, es común encontrar los conjuntos de datos divididos en los siguientes tres subconjuntos:

Conjunto de entrenamiento: Es el subconjunto de datos que se usará para ajustar el modelo, este conjunto contiene todas las muestras que se usarán para

entrenar el modelo, en el aprendizaje profundo, este entrenamiento ajustará los parámetros del diseño. El modelo es capaz de ver y de aprender de esta fuente de datos.

Conjunto de validación: Es el subconjunto que se usará para realizar evaluaciones sobre el rendimiento del modelo, sobre todo para realizar el ajuste correcto de los hiperparámetros. Originalmente solo se separaba un conjunto de prueba, sin embargo, el ajuste iterativo de los hiperparámetros para reducir el error de prueba puede sesgar la evaluación. Una práctica más correcta consiste en ajustar los hiperparámetros del modelo haciendo uso del conjunto de validación, para posteriormente únicamente realizar la evaluación sobre el conjunto de prueba.

Conjunto de prueba: El conjunto de prueba contiene las muestras sobre las cuales se realizará la evaluación del rendimiento del modelo sobre datos nunca antes vistos. La evaluación sobre este conjunto se realiza únicamente al final, cuando el modelo y sus hiperparámetros ya hayan sido ajustados sobre el conjunto de validación.

2.2.3. Capacidad del modelo, sobre-ajuste y sub-ajuste

Como ya se mencionó anteriormente, la generalización de un modelo es el problema central a resolver en el aprendizaje automático y por ende, en el aprendizaje profundo. Es posible medir la cantidad de generalización de un modelo M sobre el conjunto de entrenamiento mediante la medición de P, que es, para fines prácticos, cualquier métrica que sirva para comparar las salidas actuales con las salidas deseadas. A partir de ahora, la métrica P será llamada "función de coste". Si la función de coste durante el entrenamiento mide algún tipo de error, entonces a la medición de ese error sobre el conjunto de entrenamiento se le llamará "error de entrenamiento"; el propósito del procedimiento de optimización seleccionado consiste en reducir dicho error de entrenamiento.

Si se desea medir la generalización del modelo, entonces es necesario medir, con la función de coste, el error que el modelo arroja cuando se le encarga predecir salidas para entradas nunca antes vistas, a este error se le llamará "error de prueba" o "error

de generalización". A diferencia de los problemas de otros problemas de optimización, en el aprendizaje profundo se busca que tanto el error de entrenamiento como el error de prueba se minimicen [Goodfellow et al., 2016]. Comúnmente, el error de generalización del modelo se mide evaluando su rendimiento sobre un conjunto de prueba, dicho conjunto de prueba consiste de muestras recolectadas de forma separada al conjunto de entrenamiento, o, de forma más práctica, son muestras que se separan del conjunto de entrenamiento, de modo que el modelo no tenga la capacidad de explorarlas mientras entrena.

En general, se puede resumir el rendimiento de un algoritmo según su habilidad para:

- Minimizar el error de entrenamiento.
- Minimizar la brecha que existe entre el error de entrenamiento y el error de prueba.

Cuando un modelo no es capaz de minimizar el error de entrenamiento de manera satisfactoria (según la función de coste seleccionada), se dice que ese algoritmo está sufriendo de *sub-ajuste*, en otras palabras, el diseño del algoritmo no es capaz de resolver el problema dado con los datos disponibles. En cambio, si el error en el entrenamiento es muy bajo, pero el error de prueba no es cercano al error de entrenamiento, se dice que el sistema está *sobre-ajustado*, es decir, aprendió demasiado bien el conjunto de entrenamiento, pero no logró generalizar para ejemplos nunca antes vistos. Ambos de los problemas mencionados arriba se controlan mediante la *capacidad* del modelo. Según [Goodfellow et al., 2016], la capacidad de un modelo es su habilidad para ajustar una amplia variedad de funciones. Los modelos con poca capacidad pueden tener problemas para ajustar el conjunto de entrenamiento, mientras que los modelos con mucha capacidad pueden caer en la memorización de los datos de entrenamiento.

La capacidad de un modelo puede ajustarse de muchas maneras distintas, algunos ejemplos se listan a continuación:

- Alterando la cantidad de muestras en los datos de entrenamiento (e.g. recolección de datos, aumentado sintético de datos).

- Añadiendo regularización a los modelos (e.g. dropout, penalizaciones).
- Alterando directamente el tamaño del modelo (e.g. cantidad de neuronas por capa, cantidad de capas).
- Modificando los hiperparámetros (cada modelo de aprendizaje profundo tiene distintos hiperparámetros).

2.2.4. Transferencia de aprendizaje

Sin duda, una de las prácticas que ha acelerado el desarrollo de modelos de aprendizaje profundo cada vez más capaces y complejos es la transferencia de aprendizaje. El término de transferencia de aprendizaje hace referencia a la situación donde lo aprendido en una configuración, por ejemplo, una distribución P_1, se explota para mejorar la generalización en otro entorno, por ejemplo, una distribución P_2 [Goodfellow et al., 2016]. En un entorno de aprendizaje supervisado, la transferencia de aprendizaje supone asumir que algunos comportamientos de P_1 son relevantes para comprender el comportamiento de P_2, incluso cuando P_2 tenga una naturaleza ligeramente distinta. En otras palabras, transferir el aprendizaje de un modelo M_1 a un modelo M_2, significa tomar el modelo M_1 (puede tomarse solo una parte del modelo) entrenado sobre el conjunto D_1 y utilizar los pesos de dicho modelo para inicializar el modelo M_2, de modo que el entrenamiento de M_2 sobre un conjunto D_2 contenga algunas "intuiciones" sobre el comportamiento de los datos.

Supóngase que D_1 es un conjunto de datos que se encarga de reconocer si una imagen de entrada es o no es un perro, y el modelo M_1 fue entrenado para resolver dicha tarea. Por otra parte, se desea entrenar un modelo M_2 sobre un conjunto de datos que se encarga de clasificar imágenes entre perros y gatos, para esto tenemos el conjunto D_2. Dadas las condiciones anteriores, si M_2 se inicializa con los pesos pre-entrenados de M_1, ese modelo ya contendrá nociones sobre como luce un perro, por lo que su aprendizaje únicamente tendrá que aprender como luce un gato y determinar a que clase pertenece la entrada.

La transferencia de aprendizaje es muy utilizada en el aprendizaje profundo aplicado a la visión computacional [Zhuang et al., 2020], sin embargo, los últimos

avances en procesamiento del lenguaje natural también han utilizado este tipo de técnicas [Devlin et al., 2018, Dong et al., 2019]. La transferencia de aprendizaje o conocimiento acelera los flujos de trabajo en el área, reduciendo los tiempos de entrenamiento y, en la mayoría de los casos, ha resultado fundamental para superar el estado-del-arte.

2.2.5. Aprendizaje profundo multimodal

Uno de los problemas más importantes del procesamiento del lenguaje natural es el "entendimiento del lenguaje natural" (NLU, por las siglas en inglés) [Parcalabescu et al., 2021], y como se argumenta en [Bender and Koller, 2020], el NLU no se puede conseguir únicamente con texto, es decir, un paso hacia adelante en el entendimiento del lenguaje natural consiste en utilizar "sentidos" adicionales, como la vista o los sonidos. Es por esto que existe un gran interés para atacar diversos problemas desde una perspectiva de aprendizaje automático multimodal.

Una definición de la multimodalidad (desde la perspectiva humana) es aquella dada por [Baltrušaitis et al., 2018]: "Nuestra experiencia del mundo es multimodal, vemos objetos, escuchamos sonidos, sentimos texturas, olemos olores y saboreamos sabores. La multimodalidad se refiere a la forma en que algo sucede o se experimenta". Sin embargo, es más correcto definir la multimodalidad desde el punto de vista de una máquina, pues, el tipo de trabajo que se realiza en esta tesis consiste en máquinas capaces de aprender de forma multimodal. En [Parcalabescu et al., 2021] se propone una definición para una tarea multimodal: "Una tarea de aprendizaje automático es multimodal cuando sus entradas o salidas están representadas de forma distinta o cuando están compuestas por distintos tipos de unidades atómicas de información", en donde una unidad atómica de información se considera un tipo de métrica para la información, por ejemplo, si se midieran objetos físicos, el tiempo sería una unidad atómica, mientras que el espacio sería otra completamente distinta.

En este trabajo se presentarán arquitecturas y modelos de aprendizaje profundo que son multimodales. Específicamente, la multimodalidad de las propuestas se puede dar en dos sentidos:

1. Respecto al tipo de entrada: los modelos serán capaces de recibir dos modali-

dades: imágenes y texto, y los procesarán de forma combinada.

2. Respecto a la entrada y la salida: los modelos recibirán dos modalidades de entrada (imágenes y texto) y serán capaces de dar como salida una única modalidad, el texto.

2.2.6. Modelos del lenguaje y *embeddings*

Las computadoras no son capaces de interpretar o comprender el lenguaje natural humano; a bajo nivel, las computadoras únicamente entienden números representados en un sistema binario. Si se desea que las computadoras sean capaces de trabajar con texto y con lenguaje natural, es necesario representar dicho lenguaje de forma que las computadoras puedan trabajar con él, es decir, se necesitan métodos de representación de texto que lleven a las palabras o frases a una representación numérica.

En la actualidad existen diversos métodos de representación (o codificación) de texto, a continuación se describen algunos de los métodos más comunes.

Representación 1-hot. Si un documento tiene un vocabulario (términos o palabras únicas que aparecen en los datos) de 1000 palabras, se puede representar cada una de esas palabras con un vector 1-hot, es decir, cada palabra se representaría con un vector de 1000 elementos, en donde únicamente se enciende (con 1) el índice que representa la palabra en cuestión, mientras que las 999 palabras restantes se quedan en 0. Una frase de N palabras estaría representada por un conjunto de N vectores 1000-dimensionales.

Representación con n-gramas. La representación por n-gramas consiste en dividir la oración o el texto completo en sub-cadenas de n palabras cada una, por ejemplo, una oración con cuatro palabras se representaría con: cuatro 1-gramas, tres 2-gramas o dos 3-gramas. Los modelos de lenguaje basados en n-gramas aprenden el comportamiento del lenguaje mediante la estimación de la probabilidad de la siguiente palabra dado un n-grama de entrada. Cuanto más grande sea n, el modelo basado en n-gramas tomará contextos más y más grandes. Este modelo funciona

bien para trabajar con contextos o memorias de corto plazo, pero todo se complica cuando se desea considerar el contexto de textos completos (como artículos o libros).

Bolsas de palabras. Las bolsas de palabras son otra manera de representar textos, en este caso, el orden en que las palabras están escritas no es tomado en cuenta. En las bolsas de palabras, los textos se representan según las palabras que contengan, únicamente considerando las frecuencias con las que dichas palabras aparecen. Comúnmente, los modelos de lenguaje basados en bolsas de palabras utilizan la frecuencia de los términos, que se define como la cantidad de veces que una palabra ocurre en un texto o documento.

Semántica vectorial o *embeddings*. Los métodos descritos anteriormente pueden ser poco útiles si se desea resolver problemas en donde el significado de las palabras es importante, por ejemplo, problemas como los resúmenes automáticos, los diálogos, la generación de texto coherente o la respuesta de preguntas dado un texto de entrada. Los *embeddings* atacan este tipo de problemas haciendo uso del contexto de la palabra, es decir, este tipo de modelos es capaz de "aprender" el significado de una palabra dado su contexto. Google[1] define los *embeddings* como sigue: "Un *embedding* es un espacio de (relativamente) baja dimensionalidad al que se pueden trasladar vectores de alta dimensionalidad". Los *embeddings* facilitan que los modelos de aprendizaje automático trabajen con entradas muy grandes (como vectores dispersos que representan palabras). Idealmente, un *embedding* captura la semántica de las palabras, porque los vectores que representan palabras similares se ubican más cerca que aquellos que representan palabras diferentes. Existen diversos modelos de incrustación de texto, entre los que se pueden encontrar: capas de incrustación [Goldberg and Hirst, 2017], word2vec [Mikolov et al., 2013b], Global Vectors [Pennington et al., 2014], FastText [Bojanowski et al., 2017], entre otros.

[1] https://developers.google.com/machine-learning/crash-course/embeddings/

2.3. Conceptos relacionados con la DAICL

La descripción automática de imágenes es un área de la visión computacional que ha crecido rápidamente en la última década. Las investigaciones en el área buscan resolver problemas como la comprensión de escenas visuales y la descripción de imágenes con lenguaje natural: mientras la comprensión se enfoca en reconocer objetos, escenas e interacciones, la parte generadora de texto se encarga de generar descripciones que reflejen correctamente el contenido de la escena [Hossain et al., 2019].

El presente trabajo de tesis se enfoca únicamente en técnicas de DAI que están basadas en aprendizaje profundo. En la DAI, la principal diferencia entre las técnicas de aprendizaje automático tradicional y las de aprendizaje profundo radica en la forma en que se extraen características de las entradas. En las técnicas tradicionales se hace uso de características manuales como patrones binarios locales, transformación de características invariantes de escala, entre otras, mientras que en el aprendizaje profundo se aprende el proceso de extracción de características durante el entrenamiento, pero también se pueden utilizar redes de extracción de características que hayan sido entrenadas previamente. El aprendizaje profundo permite superar las complejidades y los retos del subtitulado de imagen de forma satisfactoria [Hossain et al., 2019].

Hossain et al. propusieron una clasificación para más de cuarenta métodos de subtitulado de imagen disponibles en la literatura, misma que toma seis ejes principales:

- Tipo de aprendizaje: Supervisado u Otros,
- Arquitectura: Composicional o codificador-decodificador,
- Mapeo de características: Espacio visual o espacio multimodal,
- Modelado del lenguaje: *Long Short-Term Memory* (LSTM) u otros,
- Número de descripciones: Subtitulado denso o subtitulado de toda la escena,
- Otros: Basados en atención, basados en conceptos semánticos, basados en objetos o descripciones estilizadas.

2.3.1. Aprendizaje profundo aplicado a la visión computacional (VC)

Es una de las áreas de investigación más activas, dado que la visión es una tarea que los humanos realizan de forma automática y prácticamente sin esfuerzo, pero que históricamente ha resultado una de las tareas más complejas de resolver para las computadoras [Goodfellow et al., 2016]. La aplicación del aprendizaje profundo a tareas de visión ha llevado al área a resolver problemas con un rendimiento excepcional.

La mayoría de aplicaciones de la visión computacional buscan imitar capacidades humanas en el área de la percepción, como el reconocimiento de objetos, rostros o texto, sin embargo, en los últimos años también se han visto acercamientos a la creación de nuevas habilidades visuales para las computadoras [Chai et al., 2021]. Sin embargo, la mayoría de los algoritmos de aprendizaje profundo aplicados a la visión computacional se aplican al reconocimiento de objetos o formas, en otras palabras, estos algoritmos reportan que objetos y donde están dentro de una imagen. Otras aplicaciones enfocadas al texto se enfocan en encontrar texto y transcribir los símbolos en las imágenes. Otras aplicaciones buscan desarrollar modelos capaces de generar o sintetizar imágenes dadas ciertas condiciones de entrada [Goodfellow et al., 2016].

Los algoritmos de visión computacional utilizados en este trabajo se enfocan, principalmente, en tres tareas: extracción de características, reconocimiento de objetos y extracción de texto, en este caso, para los tres problemas se hace uso de soluciones basadas en aprendizaje profundo.

2.3.2. Aprendizaje profundo aplicado al procesamiento del lenguaje natural (PLN)

El procesamiento del lenguaje natural consiste en dotar a las computadoras de la capacidad de utilizar lenguajes o idiomas humanos (como el español o el inglés). Las computadoras fueron diseñadas para trabajar con lenguajes de programación, que son opciones eficientes, no ambiguas, y formalmente definidos. Por otra parte, los lenguajes naturales humanos son comúnmente ambiguos y básicamente imposibles

de definir formalmente [Goodfellow et al., 2016]. La mayoría de las aplicaciones del aprendizaje profundo al PLN están basadas en la definición de una distribución de probabilidad sobre secuencias de palabras o caracteres en un lenguaje natural.

En la actualidad, los resultados más avanzados en el área del PLN se han alcanzado gracias a modelos basados en aprendizaje profundo, sobre todo cuando se trata de tareas generativas como la traducción automática o el modelado del lenguaje [Torfi et al., 2020], sin embargo, el aprendizaje profundo también ha impulsado otras áreas como el reconocimiento de entidades nombradas o la clasificación de documentos [Torfi et al., 2020]. Aún más impresionante resulta el hecho de que ciertos modelos de aprendizaje profundo aplicados al modelado del lenguaje han demostrado aprender tareas para las que no fueron entrenados, por ejemplo, GPT-2 [Radford et al., 2018] fue capaz de resolver tareas de resumen y de pregunta-respuesta, cuando únicamente fue entrenado para predecir la palabra faltante en un determinado texto (modelado del lenguaje enmascarado).

En la mayoría de casos, los modelos de aprendizaje profundo trabajan con el lenguaje natural como secuencias de tokens (palabras, caracteres o bytes), es decir, como datos secuenciales, y dada la inmensa cantidad de tokens existentes, los modelos basados en palabras tienen que trabajar en espacios discretos y dispersos con una dimensionalidad extremadamente alta [Goodfellow et al., 2016]. Por lo anterior, es común encontrar que los modelos de aprendizaje profundo aplicados al PLN sean extremadamente grandes y complejos, aun así, se han desarrollado soluciones que hagan eficiente dicho espacio, tanto en términos computacionales como estadísticos [Goodfellow et al., 2016].

Las técnicas de procesamiento y generación de lenguaje natural utilizadas en este trabajo están basadas en aprendizaje profundo.

2.3.3. Introducción a las métricas de evaluación para TextCaps

Como fue propuesto por los autores del conjunto TextCaps [Sidorov et al., 2020], se utilizarán cinco métricas automáticas para evaluar los resultados de las descripciones generadas. A continuación se incluye una breve descripción de cada métrica y

su comportamiento. Es importante mencionar que todas las métricas utilizadas en este trabajo (que trabajan en un rango entre 0 y 1) se estandarizaron a un rango de entre 0 y 100.

Bilingual Evaluation Understudy Score-4 (BLEU-4)

La primera métrica es llamada *Bilingual Evaluation Understudy Score* (BLEU), y apareció por primera vez en el trabajo de [Papineni et al., 2002], que se encarga de evaluar una frase generada s_1 tomando como referencia otra frase considerada como el objetivo s_2. En esta métrica, si s_1 es exactamente igual a s_2, el puntaje BLEU será de 1, y si las frases son completamente diferentes, el puntaje será de 0.

Esta métrica funciona a través del uso de n-gramas, el algoritmo cuenta la cantidad de n-gramas de s_1 y s_2. Por ejemplo, la métrica utilizada en este trabajo es BLEU-4, es decir, se medirá la cantidad de 4-gramas que coinciden en ambas frases. Es importante mencionar que la comparación de n-gramas se hace sin tomar en cuenta la posición de los mismos. Según los autores, esta métrica está altamente correlacionada (0.817 a nivel de corpus) con una evaluación humana.

Metric for Evaluation of Translation with Explicit ORdering (METEOR)

METEOR [Denkowski and Lavie, 2014] es una métrica utilizada para evaluar la calidad de las traducciones realizadas por computadora. En este caso, la métrica trabaja mediante la alineación de una frase generada s_1 con una o mas frases de referencia $s_1, ...s_n$. A diferencia de otras métricas, METEOR realiza un proceso de *stemming* y es capaz de comparar sinónimos, además de buscar coincidencias exactas de palabras. La correlación con la evaluación humana sobre el conjunto de datos MS COCO [Lin et al., 2014] alcanzó los 0.53 puntos. En la actualidad, el soporte de METEOR para detección de paráfrasis incluye varios idiomas, incluyendo el inglés y el español. Al igual que con BLEU, dos frases perfectamente emparejadas obtendrán un puntaje de 1, mientras que dos frases completamente diferents obtendrán un puntaje de 0.

Recall-Oriented Understudy for Gisting Evaluation-L (ROUGE-L)

ROUGE [Lin, 2004] es un conjunto de métricas utilizadas para medir la similitud entre dos textos. Existen varias formas de utilizar esta métrica, por ejemplo, ROUGE-N mide la cantidad de n-gramas que coinciden en la salida s_1 del modelo y la referencia s_2. Sin embargo, en este trabajo se hace uso de ROUGE-L, una alternativa que mide la Subsecuencia Común más Larga (SCL) entre s_1 y s_2, es decir, que se mide la secuencia de tokens más larga que se encuentra en ambos textos. En METEOR, cuanto más alto sea el puntaje, mejor.

Semantic Propositional Image Caption Evaluation (SPICE)

SPICE [Anderson et al., 2016] es una métrica específicamente diseñada para evaluar descripciones automáticas de imagen. Según los autores, las métricas tradicionales trabajan con las intersecciones de los n-gramas en s_1 y s_2, lo que argumenta, no es ni necesario ni suficiente para simular la evaluación humana. La correlación con la evaluación humana sobre el conjunto de datos MS COCO [Lin et al., 2014] alcanzó los 0.88 puntos. SPICE es capaz de medir la similitud entre s_1 y un conjunto de referencias $s_1, ..., s_n$. Esta métrica evalúa la similitud de las entradas tomando en cuenta el contenido semántico proposicional, el cual, los autores proponen tomando como base un "grafo de la escena", que representa el contenido semántico de las entradas tomando en cuenta conocimiento previo. La correlación de SPICE con la evaluación humana sobre el conjunto de datos MS COCO [Lin et al., 2014] alcanzó los 0.88 puntos, superando con creces a METEOR. En SPICE, los valores más altos son mejores.

Consensus-based Image Description Evaluation (CIDEr)

Al igual que SPICE, CIDEr es una métrica específicamente diseñada para evaluar la calidad de las descripciones generadas para una imagen. Esta métrica usa un tipo de "consenso" humano para realizar la evaluación. CIDEr consiste de tres partes: un método que recolecta anotaciones humanas para generar el consenso, una nueva métrica que captura dicho consenso y dos conjuntos de datos que contienen 50 descripciones para cada imagen. Se podría resumir a esta métrica como una

combinación del algoritmo *Term Frquency - Inverse Document Frequency* (TF-IDF) aplicado a n-gramas y una técnica de *stemming* [Hessel et al., 2021].

CAPÍTULO 3

Estado del Arte

A la fecha de escritura de este trabajo, únicamente se encuentran publicados siete artículos científicos que buscan resolver el problema planteado por el conjunto de datos TextCaps. La novedad del problema supone varias oportunidades para presentar propuestas novedosas en el tema, que se acerquen o incluso que superen al estado-del-arte en la literatura.

El problema que se busca resolver en esta tesis pertenece a un área específica: subtitulado automático de imagen con comprensión lectora. Como se menciona en [Sidorov et al., 2020], los métodos de *image captioning* en el estado del arte se enfocan únicamente en los objetos y detalles visuales de la escena y no reconocen o "razonan" sobre texto presente en las imágenes.

Como se mencionó anteriormente, TextCaps es el primer y único conjunto de datos enfocado a resolver este problema, la investigación sobre el estado del arte está compuesta por trabajos publicados (en revistas o servidores de preimpresión) que utilicen dicho conjunto para resolver el problema.

3.1. Antecedentes

Antes de que el problema de la DAICL fuera planteado formalmente por *Facebook Artificial Intelligence Research* (FAIR), ya existían problemas similares y que ahora se consideran bases fundamentales para resolver la DAICL. A continuación se describen brevemente los tres problemas que resultan más importantes y que más aportaciones tienen al resultado final de esta tesis.

Descripción automática de imagen. Como ya se comentó en secciones anteriores, problema de DAI, también llamado "subtitulado automático de imágenes" o *image captioning* en inglés, consiste en generar descripciones en lenguaje natural para escenas visuales (imágenes). Las primeras aproximaciones al problema se basaban en plantillas, por ejemplo, primero se extraía una serie de objetos o atributos a través de clasificadores (e.g. máquinas de vectores de soporte) y posteriormente se integraba dicha información en una cadena, haciendo uso de las ya mencionadas plantillas [Wang et al., 2021a].

Los primeros acercamientos al problema ocurrieron en 2004 [Wang et al., 2021a], sin embargo, no fue hasta el año 2015 que el aprendizaje profundo comenzó a brillar por sus resultados en el problema, principalmente haciendo uso de redes neuronales convolucionales (CNN, por las siglas en inglés) para la extracción de características y redes neuronales recurrentes (RNN, por las siglas en inglés) para la generación de texto [Amirian et al., 2019]. Otros métodos más innovadores, como el uso de mecanismos de atención [Xu et al., 2015], también obtuvieron resultados sorprendentes.

Existen varios conjuntos de datos para este problema, a continuación se incluyen algunos de los más comunes en la literatura: Flickr8k [Hodosh et al., 2013], Flickr30k [Young et al., 2014] o Microsoft COCO [Lin et al., 2014], que contienen desde las ocho mil ejemplos anotados, hasta más de 164 mil, respectivamente. Sin embargo, ninguno de dichos conjuntos se enfoca en hacer uso del texto presente en las escenas [Sidorov et al., 2020]

Reconocimiento Óptico de Caracteres. En el problema a resolver, el reconocimiento óptico de caracteres (OCR, por las siglas en inglés) juega un papel sumamente importante.

El OCR usualmente implica dos pasos:

1. Encontrar la ubicación del texto,
2. extraer el texto como caracteres.

Lo anterior otorga a los sistemas la capacidad de procesar los datos extraídos de forma textual. El OCR es uno de los componentes más importantes para conseguir sistemas de subtitulado de imagen con comprensión lectora, pues este será el encargado de determinar la información textual que contenga la escena, para después utilizarla en la generación de descripciones significativas.

En la actualidad existe una gran variedad de sistemas de OCR [Smith, 2007, Li et al., 2017, Borisyuk et al., 2018, Liu et al., 2018, He et al., 2018], mismos que han evolucionado de forma satisfactoria a través de la última década, y que, en general, han presentado mejoras tanto en confiabilidad como en rendimiento, sin embargo, con el surgimiento del conjunto de datos TextCaps, el problema del reconocimiento óptico de caracteres parece estar lejos de ser un problema resuelto en escenarios del mundo real [Sidorov et al., 2020].

***Visual Question Answering* (VQA) con capacidad de leer texto.** Como se mencionó anteriormente, el problema de añadir comprensión lectora a sistemas de subtitulado de imagen es nuevo, sin embargo, en la literatura se puede encontrar un problema muy similar: dada una imagen de entrada, contestar preguntas sobre el contenido o el significado de la imagen, considerando texto escrito en la escena.

Se pueden encontrar distintos conjuntos de datos orientados a la detección de texto en sistemas de VQA:

- TextVQA, hecho público en el trabajo *Towards VQA Models That Can Read* [Singh et al., 2019],
- Scene Text VQA, publicado en *Scene Text Visual Question Answering* [Biten et al., 2019],

- OCR-VQA, presentado en la publicación *OCR-VQA: Visual Question Answering by Reading Text in Images* [Mishra et al., 2019].

Todos los conjuntos de datos mencionados anteriormente requieren de sistemas con la capacidad de leer y razonar sobre el texto observable en las escenas de entrada, además de tomar en cuenta el contexto para responder las preguntas [Sidorov et al., 2020]. La principal diferente entre este problema y el que se desea resolver en esta tesis radica en que una modalidad de preguntas y respuestas no es directamente aplicable a la generación de descripciones correctas y significativas.

3.2. Estado del arte sobre el conjunto de datos TextCaps

Todos los trabajos publicados rebasaron los puntajes del modelo base del reto TextCaps [1]. Los enfoques varían según cada una de las propuestas, algunos autores intentaron incrementar el rendimiento del modelo mediante el uso de técnicas de pre-entrenamiento sobre tareas distintas pero relacionadas a TextCaps [Yang et al., 2020], otros implementaron arquitecturas basadas en grafos, de modo que el grafo fuera auxiliar en la búsqueda y exploración de diversidad para la descripción [Xu et al., 2021]. Los trabajos restantes se enfocaron en modificar o mejorar ciertos componentes o módulos de la arquitectura M4C original [Sidorov et al., 2020], como los módulos de codificación, los generadores de texto o la red encargada de seleccionar que palabras del OCR usar [Wang et al., 2020, Wang et al., 2021b, Wang et al., 2021c, Zhu et al., 2020]. Esta sección introduce y describe brevemente cada método disponible en la literatura, de igual forma, se incluyen sus puntajes sobre los conjuntos de validación y prueba de TextCaps.

[1] https://textvqa.org/textcaps/challenge/

3.2.1. Multimodal Multi-copy Mesh Captioner (M4C Captioner)

M4C-Captioner fue propuesta por [Sidorov et al., 2020] como la primera arquitectura para resolver el problema, de igual forma, fue presentada por los organizadores como la base para el reto TextCaps. M4C-Captioner está basada en la arquitectura M4C, que en su momento rebasó el estado-del-arte para el reto de *Text Visual Question Answering* (TextVQA) [Hu et al., 2020]. Este modelo fusiona distintas modalidades mediante su incrustación en un espacio semántico común, el procesamiento posterior se lleva a cabo haciendo uso de un Transformer multimodal (MMT, por las siglas en inglés), la generación de texto es llevada a cabo por una red apuntadora, que permite al modelo generar respuestas de varias palabras, mientras mezcla tokens pertenecientes al vocabulario del modelo o provenientes del OCR.

3.2.2. Multimodal Attention Captioner with OCR Spatial Relationship (MMA-SR)

En [Wang et al., 2020], los autores proponen la arquitectura llamada MMA-SR, como un método novedoso para la DAI basada en OCR (o DAICL). Esta arquitectura primero extrae una representación de las características de la imagen, principalmente utilizando un modelo pre-entrenado de la red *Faster R-CNN* [Ren et al., 2015], posteriormente se extrae la información OCR haciendo uso de sistemas OCR externos (de forma adicional a los datos OCR ya presentes en TextCaps). Después, las dos modalidades extraídas anteriormente se introducen a una red basada en mecanismos de atención multimodales, en donde, a diferencia de M4C-Captioner, la tarea de generación de texto se lleva a cabo con una red LSTM en vez de un MMT. En el último paso, una red apuntadora dinámica selecciona palabras cantidatas, ya sea del vocabulario del modelo o provenientes del OCR.

3.2.3. Text-Aware Pre-Training for Text-VQA and Text-Caption (TAP)

El método TAP reportó algunos de los puntajes más altos durante el reto TextCaps (ver Tabla 3.1), TAP fue propuesto por primera vez en [Yang et al., 2020]. La principal contribución de este método es una estrategia novedosa de pre-entrenamiento, que ayuda al modelo a aprender representaciones mejor alineadas entre las palabras, los objetos visuales y el texto extraído de la escena (OCR). La estrategia de pre-entrenamiento mencionada anteriormente consiste en tres tareas: modelado del lenguaje, emparejamiento de imagen-texto y predicción de posición relativa. Durante el pre-entrenamiento, el modelo aprende representaciones que resultan útiles para las tres modalidades, posteriormente, todas las modalidades se procesan con un MMT (llamado módulo de fusión). Una vez que el modelo finaliza su pre-entrenamiento, el módulo de fusión es ajustado para realizar tareas específicas tal como las ya mencionadas TextVQA o TextCaps.

3.2.4. Simple is not Easy (SBD)

En el artículo de Simple is not Easy [Zhu et al., 2020], los autores argumentan que los *frameworks* de codificación multimodales utilizados en otros métodos son muy sofisticados y, que además, no son estrictamente necesarios para alcanzar puntajes estado-del-arte sobre TextCaps. En dicho artículo se propone un mecanismo de atención, al que los autores se refieren como "simple", que se encarga de filtrar las características extraídas de la imagen de entrada, antes de ser introducidas a un "codificador de fusión". La arquitectura que se propone consiste en la separación de los tokens OCR en dos ramas: información visual e información lingüística, que posteriormente son introducidos en bloques de atención. Una vez los bloques de atención procesan las entradas, estas son introducidas a un codificador de fusión (que codifica las características de las distintas modalidades). Las descripciones textuales son generadas con un MMT, que recibe las representaciones multimodales de la entrada.

3.2.5. Confidence-aware Non-repetitive Multimodal Transformers for TextCaps (CNMT)

La arquitectura CNMT [Wang et al., 2021c] consiste de tres componentes principales: un módulo de lectura, un módulo de razonamiento y un módulo de generación de texto. El módulo de lectura fue mejorado, si se compara con otras alternativas en la literatura, pues los autores mencionan utilizar mejores sistemas de OCR y además, utilizar la confidencia de reconocimiento (i.e. que tan confiado está el sistema sobre el token detectado) para mejorar las predicciones. El módulo de razonamiento (un MMT) fusiona las características de los tokens OCR con las características de los objetos visuales, posteriormente, dichas características son introducidas al módulo de generación de texto, que predice las descripciones de forma iterativa. Al final, se utiliza una red apuntadora encargada de seleccionar tokens del vocabulario del modelo o del OCR, sin embargo, la red apuntadora de CNMT es mejorada haciendo uso de una "máscara de repetición" que ayuda al modelo a evitar descripciones con palabras repetidas.

3.2.6. Anchor-Captioner (AnC)

De la literatura explorada, la arquitectura [Xu et al., 2021] presenta el único enfoque basado en grafos, AnC está principalmente diseñada para explorar contenido y extender la diversidad de las descripciones generadas. La arquitectura consiste de cuatro componentes principales: un extractor de características (tanto para la información textual como visual), un módulo de fusión (compuesto de capas de auto-atención), un módulo de propuesta de grafo (AnPM) y un módulo de generación de descripciones (AnCM). Una vez que las características de entrada se fusionan en las capas de auto-atención, el AnPM construye grafos para agrupar los tokens más relevantes, después, el AnCM utiliza un módulo generador para construir una descripción "global centrada en la información visual", que después es utilizada como semilla para generar descripciones diversas.

3.2.7. Long Short-Term Memory plus Relation-aware pointer network (LSTM-R)

La arquitectura LSTM-R [Wang et al., 2021b] se enfoca en mejorar la capacidad de "razonamiento" que el modelo tiene respecto al texto de la escena, lo anterior se logra mediante el reemplazo de los tokens OCR por información visual y semántica enriquecida, dicha mejora es llevada a cabo mediante la explotación de las relaciones geométricas existentes entre los tokens OCR. Los autores toman en cuenta la altura, anchura, distancia y orientación de las relaciones, de modo que se pueda construir un conjunto de relaciones geométricas. Además, la arquitectura LSTM-R hace uso de una red LSTM en vez de un MMT para realizar el procesamiento de las características de entrada. La última aportación de este trabajo consiste de una red apuntadora que mejora sus resultados (cuando se compara con otras redes apuntadoras utilizadas en TextCaps) mediante el uso de las relaciones geométricas obtenidas anteriormente.

3.2.8. Descripciones generadas por humanos

En [Sidorov et al., 2020] se puede encontrar una explicación detallada sobre la creación del conjunto de datos TextCaps. Para estimar el rendimiento que los humanos tendrían en la tarea, los autores recolectaron una sexta descripción adicional durante el proceso de anotación de los conjuntos de entrenamiento y prueba. Las mencionadas descripciones hechas por humanos fueron evaluadas utilizando todas las métricas anteriormente descritas, y los resultados se pueden observar en la Tabla 3.1. El rendimiento humano estimado no fue calculado para el conjunto de validación.

3.2.9. Resumen del estado del arte

En general, se han descrito siete métodos para la DAICL. En la Tabla 3.1 se presenta un resumen de los resultados reportados por cada método (únicamente se incluyeron los mejores puntajes experimentales) sobre el conjunto de prueba de TextCaps[2], también se incluye el rendimiento humano estimado por los autores del

[2]La evaluación se realiza en el servidor del reto.

conjunto de datos.

Tabla 3.1: Rendimiento de los métodos de DAICL disponibles en la literatura, comparados con el rendimiento humano esperado. El método marcado con un $*$ corresponde al modelo base del reto, mientras que los valores en negritas indican los mejores puntajes. Los resultados están ordenados de mayor a menor según su puntaje CIDEr.

	Conjunto de prueba de TextCaps				
Método	BLEU-4	METEOR	ROUGE-L	SPICE	CIDEr
TAP	21.9	**21.9**	45.7	**14.6**	**103.2**
LSTM-R	**22.9**	21.3	**46.1**	13.8	100.8
CNMT	20.1	20.9	44.2	13.5	93.2
SBD	20.2	20.3	44.2	12.8	89.6
MMA-SR	19.8	20.6	44.0	13.2	88.0
AnC	20.7	20.7	44.6	13.4	87.4
$*$ M4C-Captioner	18.9	19.8	43.2	12.8	81.0
Humano	24.4	26.1	47.0	18.8	125.5

CAPÍTULO 4

Datos y métodos

En esta sección se incluyen los detalles del conjunto de datos utilizado, toda la información necesaria para entender las arquitecturas que se proponen en este trabajo y la configuración experimental para los dos problemas a resolver.

Primero se dará una pequeña introducción al reto TextCaps (*TextCaps challenge*), además de una breve descripción del conjunto de datos TextCaps, así como una tabla que resume la cantidad de ejemplos en cada uno de los subconjuntos. Por último, se describirá la arquitectura base propuesta por los autores del conjunto de datos, conocida como M4C-Captioner.

Posteriormente se abordarán las propuestas para resolver el problema de la reducción del tamaño de los modelos en memoria. Se describirán las dos arquitecturas tomadas como ejemplo, así como las modificaciones necesarias para reducir el tamaño de los modelos. Así mismo, se presenta una tabla que resume todos los experimentos a realizar.

Por último, se presentan las propuestas para convertir la arquitectura base del reto en una viable para resolver el problema en idiomas distintos al inglés. Se describirá el proceso de traducción automática del conjunto de datos y, posteriormente, se abordan las modificaciones propuestas para generar una versión bilingual de M4C-Captioner.

4.1. Reto y conjunto de datos TextCaps

El *TextCaps challenge* (que incluye el conjunto de datos TextCaps) fue propuesto por FAIR en el año 2020. Según los autores, TextCaps requiere que los modelos sean capaces de leer y razonar sobre el texto presente en las imágenes, para posteriormente generar descripciones sobre las mismas. De forma más específica, los modelos necesitan incorporar una nueva modalidad de texto extraído de las imágenes y razonar sobre su contenido visual y semántico, así como su impacto para las descripciones finales.

4.1.1. TextCaps Challenge

La principal motivación de los autores para proponer este reto radica en las debilidades de los modelos de DAI convencionales. La mayoría de ellos falla a la hora de generar descripciones sobre el conjunto TextCaps, pues no son capaces de leer o de razonar sobre el texto presente en las imágenes. El reto consiste en la evaluación sobre el conjunto de prueba de TextCaps, que contiene anotaciones privadas (los participantes no pueden acceder a ellas) y la evaluación únicamente se puede realizar enviando las predicciones a un servidor de evaluación. Cada equipo participante tiene únicamente cinco oportunidades de enviar sus resultados, y los modelos generados no pueden ser ensambles, es decir, el problema debe resolverse de principio a fin con un único modelo. Los ganadores tuvieron la oportunidad de presentar su propuesta en la *Conference on Computer Vision and Pattern Recognition* (CVPR), edición 2021.

4.1.2. Conjunto de datos TextCaps

A la par del reto, se presentó el conjunto de datos TextCaps, cuya descripción detallada puede encontrarse en [Sidorov et al., 2020]. Este conjunto de datos sigue las particiones de TextVQA [Singh et al., 2019]. Un resumen sobre la cantidad de muestras en cada subconjunto se puede encontrar en la Tabla 4.1. Los conjuntos de entrenamiento y validación contienen tanto las imágenes como las anotaciones, mientras que el conjunto de prueba únicamente contiene las imágenes. El conjunto

Tabla 4.1: Cantidad de muestras en cada partición del conjunto TextCaps.

Tipo de dato	Entrenamiento	Evaluación	Prueba	Total
Imágenes	21,953	3,166	3,289	28,408
Descripciones	109,756	15,830	16,445	142,031

de datos es público y puede ser descargado desde el servidor del reto [1].

TextCaps está compuesto de imágenes naturales, extraídas del conjunto de datos *Open Images* [2], todas usan el modelo de color RGB. Cada imagen puede contener una extensa variedad de objetos (en promedio, 8.4 objetos por imagen) y puede contener texto escrito (OCR). Además, cada muestra contiene su respectiva anotación, que incluye cinco descripciones anotadas por humanos (de 12.4 palabras de largo, en promedio) y los tokens OCR presentes en dicha imagen (incluyendo la información espacial), extraídos con el sistema Rosetta [Borisyuk et al., 2018].

4.1.3. Arquitectura base: M4C-Captioner

Esta arquitectura ya se describió en el Capítulo 3, sin embargo, aquí se presenta una definición más clara del mismo y, además, se introduce su representación gráfica, que será de mucha utilidad para distinguir los componentes con los que se trabajará en las versiones alternativas.

Originalmente, la arquitectura M4C se propuso para resolver el problema de planteado por el conjunto TextVQA, y posteriormente fue modificada para establecer la arquitectura base del reto TextCaps, lo que originó a M4C-Captioner. Para una explicación más clara de la arquitectura, se presenta la Figura 4.1.

La arquitectura M4C-Captioner codifica la información visual haciendo uso de un modelo pre-entrenado de Faster R-CNN [Ren et al., 2015] (bloques amarillos en la Figura 4.1), a este paso también se le conoce como un proceso de extracción de características. Por otra parte, la información textual se codifica haciendo uso de un modelo pre-entrenado de un *Bidirectional Encoder Representations from Transformers* (BERT) [Devlin et al., 2018] de 3 capas, este modelo fue pre-entrenado

[1] https://textvqa.org/textcaps
[2] https://opensource.google/projects/open-images-dataset

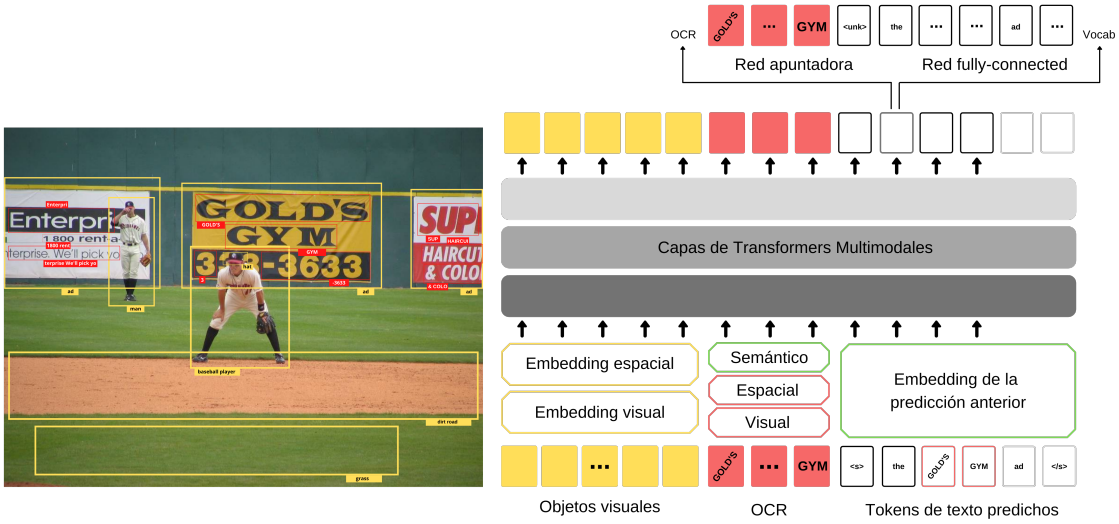


Figura 4.1: Arquitectura M4C-Captioner. Gráfico adaptado de [Sidorov et al., 2020]. Los bloques amarillos corresponden a las características de los objetos en la imagen, los bloques rojos señalan todo lo relacionado con las características del OCR. Los bloques verdes corresponden a los módulos que se modificarán para obtener arquitecturas alternativas.

para resolver una tarea de modelado del lenguaje enmascarado (MLM, por las siglas en inglés); el módulo de incrustación (*embedding*, en inglés) se puede visualizar en la Figura 4.1, en el bloque nombrado "Embedding de la predicción anterior". Aún más importante para los objetivos de este trabajo es el módulo de incrustación de OCR (bloques en rojo en la Figura 4.1), en donde M4C-Captioner codifica una representación enriquecida del OCR (incluye información visual, espacial y semántica). El bloque de procesamiento de OCR realiza lo siguiente: las características visuales del OCR se extraen con el modelo de Faster R-CNN, la información respecto de los caracteres se extrae con una PHOCNet (*Pyramidal Histogram of Characters network*) [Sudholt and Fink, 2016] y la información semántica del OCR se realiza con vectores pre-entrenados de FastText [Mikolov et al., 2018]. La información espacial (ubicación de los tokens OCR) únicamente se basa en los cuadros delimitadores calculados por el módulo de lectura.

Una vez que todas las modalidades han sido incrustadas en un espacio semántico en común, estos vectores se dan como entrada a un *Transformer* [Vaswani et al., 2017] multimodal de 4 capas (bloques de color gris en la Figura 4.1), de modo que este se encarge del procesamiento. Finalmente, la salida del MMT se pasa al módulo

encargado de la generación de texto, que es el responsable de generar la descripción en lenguaje natural. El módulo generador de texto consiste de dos módulos: una red apuntadora y una capa completamente conectada (*fully-connected*); la red apuntadora decide si un token OCR debe ser copiado e introducido en la descripción, mientras que la capa completamente conectada se encarga de elegir tokens desde el vocabulario del modelo.

4.2. Proponiendo alternativas más ligeras en memoria

Desarrollar propuestas capaces de alcanzar rendimientos cercanos al estado-del-arte, mientras son reducidas en su uso de memoria es uno de los objetivos que se plantearon anteriormente. En esta sección se introducen dos nuevas arquitecturas: Lighter-M4C (L-M4C) y Lighter-CNMT (L-CNMT), ambas son versiones modificadas de enfoques que se enfocan en resolver el problema de la DAICL con TextCaps. L-M4C está basada en la arquitectura M4C-Captioner descrita en la sección anterior, y L-CNMT está basada en una arquitectura *Confidence-Aware Non-repetitive Multimodal Transformer* (CNMT) [Wang et al., 2021c]. Las dos propuestas consisten en el reemplazo de los módulos de procesamiento semántico de OCR y en el módulo de incrustación de texto. A continuación se detalla la selección de las arquitecturas base, así como los bloques a modificar, también se incluye una descripción detallada de los experimentos a realizar.

4.2.1. Arquitecturas de referencia: M4C-Captioner y CNMT

Como se observa en la Tabla 3.1, el método TAP superó a todos los otros métodos en tres de las cinco métricas utilizadas, mientras que el método LSTM-R superó a TAP y los métodos restantes en dos de las cinco métricas. Desafortunadamente, el código para los métodos TAP, LSTM-R y MMA-SR no está disponible públicamente a la hora de escribir este trabajo. Por otra parte, los métodos SBD, AnC y CNMT son variaciones de la arquitectura base (M4C-Captioner). Dado que los diseños a proponer

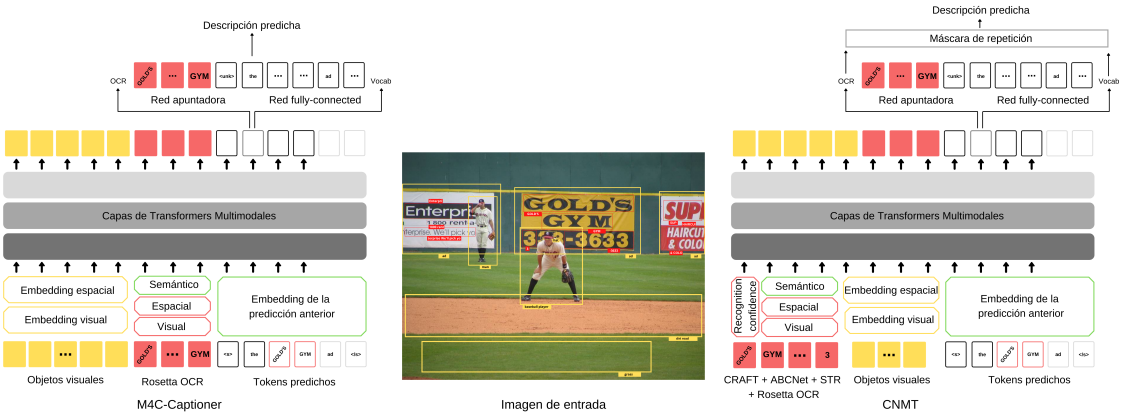


Figura 4.2: Comparación gráfica de las dos arquitecturas de referencia utilizadas en esta sección. A la izquierda se encuentra M4C-Captioner, mientras que CNMT se encuentra a la derecha.

tienen que ser más ligeros en memoria (pero con desempeño cercano al estado-del-arte) y los experimentos deben ser fácilmente reproducibles, se seleccionaron dos arquitecturas de referencia, a las cuales se les modificará para lograr los objetivos, dichas arquitecturas son: M4C-Captioner y CNMT, la primera porque es la más utilizada y mejor documentada (pues es el modelo base del reto) y la última porque es la arquitectura con mejores puntajes y que, además, hizo público el código y su conjunto de datos aumentado.

La Figura 4.2 presenta una comparación gráfica de ambas arquitecturas de referencia, la arquitectura M4C-Captioner está ilustrada a la izquierda, mientras que CNMT se encuentra a la derecha. En la Sección 4.1.3 se puede encontrar información detallada sobre el funcionamiento de M4C-Captioner, por otra parte, la información sobre CNMT se incluye a continuación.

CNMT

Anteriormente se mencionó que CNMT está basada en M4C-Captioner. Las principales diferencias entre los dos diseños se encuentra en dos módulos: el módulo de lectura y el módulo de generación de texto. Todos los bloques implicados en módulo de lectura de ambas arquitecturas están coloreados con rojo en la Figura 4.2, mientras que el módulo de generación se encuentra localizado en la parte superior de cada arquitectura, justo después de la salida de los bloques grises. A continuación se

detalla cada uno de estos módulos y su funcionamiento en CNMT.

Módulo de lectura. El módulo de lectura de M4C-Captioner está basado en Rosetta [Borisyuk et al., 2018], un sistema desarrollado por FAIR para la detección y reconocimiento de texto a larga escala. Además, en M4C-Captioner, todas las características relacionadas con el OCR son directamente extraídas de las salidas de Rosetta. Por otra parte, los autores de CNMT decidieron mejorar el módulo de lectura mediante el uso de métodos adicionales para la detección y el reconocimiento de texto. Primero, CNMT usa dos modelos para la detección de texto, *Character region awareness for text detection* (CRAFT) [Baek et al., 2019] y *Adaptive bezier-curve network* (ABCNet) [Liu et al., 2020]. Las regiones de texto extraídas por cada modelo se combinan y después se ingresan al modelo de reconocimiento: un framework STR de 4 etapas, como se describe en [Baek et al., 2019]. Los tokens OCR extraídos con el nuevo sistema OCR se combinan con los tokens OCR originales extraídos con Rosetta, dicha combinación se realiza en el conjunto de datos, es decir, todos los tokens OCR se incluyen en las anotaciones para cada imagen. Los autores de la arquitectura CNMT reportaron un estudio de ablación que muestra un incremento de 5.9 puntos CIDEr en el conjunto de evaluación, únicamente extendiendo las características OCR (Rosetta + CRAFT + ABCNet).

Además, en CNMT se utiliza la "confidencia de reconocimiento" x^{conf} de cada token extraído con el sistema OCR. La confidencia x^{conf} puede obtener valores de 0 a 1, en donde 1 significa una completa seguridad sobre el texto extraído. La confidencia x^{conf} para cada token OCR se añade a las características de entrada en el conjunto de datos.

Módulo de generación. En la arquitectura CNMT, las capacidades del módulo de generación de texto se ven aumentadas por la adición de una máscara de repetición al final del procesamiento. Los autores argumentaron que la repetición de palabras causa efectos negativos en la fluidez de las descripciones generadas [Wang et al., 2021c]. La máscara de repetición se añade para reducir o evitar dicha repetición de palabras, en este caso, la máscara minimiza los puntajes de los elementos que ya han aparecido en pasos de generación previos. Incrustar y utilizar x^{conf} incrementó el puntaje CIDEr

en 2 puntos, evaluando sobre el conjunto de validación.

4.2.2. Hacia arquitecturas más ligeras en memoria

Ambas arquitecturas de referencia han sido brevemente descritas. En esta sección se incluye información detallada sobre los cambios en cada diseño y los bloques a modificar. Como se mencionó anteriormente, la principal diferencia entre las arquitecturas originales y las propuestas de la presente tesis radican en el bloque de incrustación de OCR, específicamente, en la incrustación de las características semánticas del OCR, y en el bloque de incrustación de texto. Ambos módulos están coloreados con verde en la Figura 4.2.

Esta sección se enfocará en detallar los procesos para reducir el uso de memoria en las dos arquitecturas antes mencionadas, la modificación principal se encuentra en el componente con mayor uso de memoria tanto en M4C-Captioner como en CNMT. El componente más pesado en términos de memoria es el procesador de las características semánticas del OCR. Originalmente, ambos enfoques utilizaban vectores pre-entrenados 300-dimensionales de FastText para obtener las incrustaciones de los tokens OCR, mientras que una de las aportaciones de este trabajo consiste en utilizar vectores pre-entrenados, pero basados en diccionarios, la propuesta se enfoca en el uso de *Gloval Vectors* (GloVe) [Pennington et al., 2014], que también incrustan los tokens en un espacio 300-dimensional. Además, el módulo de incrustación de texto un componente que incrementa de forma considerable la cantidad de parámetros en el modelo, la propuesta consiste en reemplazar el modelo pre-entrenado de BERT con una versión destilada y más ligera del mismo, llamada *DistilBERT* [Sanh et al., 2019].

Dado que tanto el procesador semántico de OCR como el módulo de incrustación de texto son invariantes entre M4C-Captioner y CNMT (ambos utilizan vectores FastText (300-d) y el mismo modelo pre-entrenados de BERT), las modificaciones que se proponen pueden ser aplicadas a ambos diseños sin necesidad de alterar otros componentes de las arquitecturas. Para ilustrar con más claridad los cambios realizados, se presentan las Figuras 4.3 y 4.4, que compara los codificadores del diseño original contra los codificadores más ligeros en memoria. En ambas figuras, el diagrama a la izquierda corresponde al codificador original, mientras que el diagrama

a la derecha corresponde con la versión propuesta.

La Figura 4.3 muestra la comparación entre el codificador de M4C-Captioner (a la izquierda) y de la nueva arquitectura ligera (a la derecha): Lighter M4C (L M4C). Las entradas para ambos codificadores están formateadas de la misma manera que en la Figura 4.2.

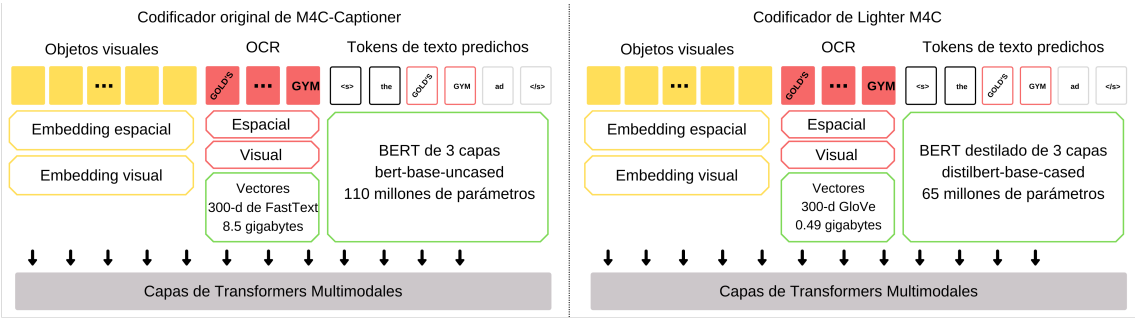


Figura 4.3: Comparación de los bloques codificadores. A la izquierda se encuentra el codificador de M4C-Captioner; a la derecha, el codificador de Lighter M4C. Los bloques modificados están resaltados en verde.

La Figura 4.4 muestra la comparación entre el codificador de CNMT (a la izquierda) y del diseño ligero propuesto de CNMT (a la derecha): Lighter CNMT (L CNMT).

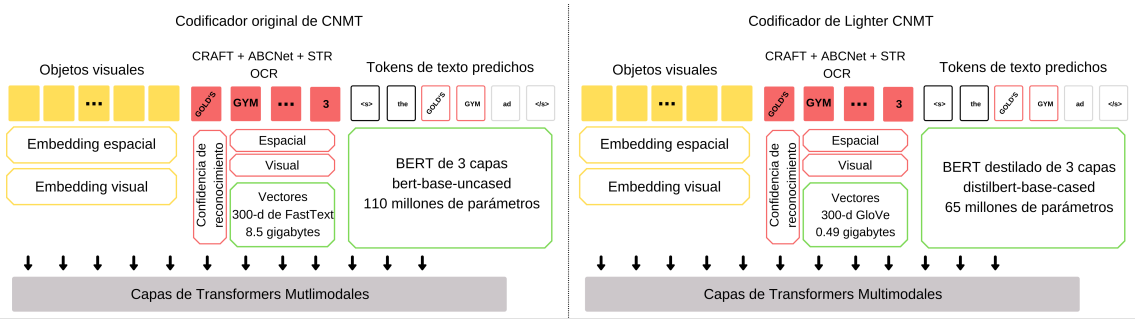


Figura 4.4: Comparación de los bloques codificadores. A la izquierda se encuentra el codificador de CNMT; a la derecha, el codificador de Lighter CNMT. Los bloques modificados están resaltados en verde.

Procesador semántico de OCR: FastText versus GloVe

Vectores FastText. Los modelos de FastText [Mikolov et al., 2018] son una extensión de un modelo de incrustación de texto publicado en 2013, comúnmente

conocido como word2vec [Mikolov et al., 2013a]. Sin embargo, un modelo de FastText representa cada palabra como un conjunto de n-gramas de caracteres, en vez de representar los vectores para cada palabra directamente. Dicha representación, conocida como "representación de sub-palabras", permite capturar el significado de palabras más cortas y procesar de mejor manera los prefijos y sufijos. Otra forma de interpretar los vectores de FastText es tomarlos como un modelo de bolsa de palabras con una ventana que se desliza sobre las palabras, pero sin una estructura interna, es decir, que mientras los caracteres se encuentren dentro de la ventana, el orden de los n-gramas no afecta la representación. Las arquitecturas M4C-Captioner y CNMT primero extraen un conjunto de N tokens OCR a través de un sistema OCR externo, posteriormente, dichas arquitecturas extraen un vector 300-dimensional para cada uno de los tokens OCR extraídos previamente. El modelo por defecto utilizado tanto en M4C-Captioner como en CNMT tiene un tamaño aproximado de 8.5 gigabytes.

Vectores GloVe. Anteriormente se describió brevemente la opción por defecto utilizada en la literatura (FastText). Una de las aportaciones de este trabajo consiste en reemplazar la forma en que se representa la información semántica del OCR, en este caso, se reemplazaron los vectores FastText por una alternativa más ligera: *Global Vectors* (GloVe). El modelo GloVe [Pennington et al., 2014] fue publicado como una alternativa al modelo word2vec, pues los autores argumentaron que el enfoque de escaneo utilizado por word2vec era sub-óptimo, pues no explota la información estadística global correspondiente a las co-ocurrencias de palabras en el texto. GloVe está construido con base en dos operaciones: factorización global de matrices y una ventana de contexto local. La primera consiste en un proceso en donde matrices que contienen las frecuencias de cada término se reducen usando factorización, dichas matrices usualmente representan la ocurrencia o la ausencia de palabras en cierto documento. Por otra parte, la ventana de contexto local puede funcionar de dos maneras distintas: como una Bolsa de Palabras Continua (CBOW, por las siglas en inglés) o un *skip-gram*. La CBOW se enfocaría en predecir la palabra actual basada en un contexto de entrada, mientras que el modelo *skip-gram* buscaría predecir el contexto, dada una palabra de entrada. Los modelos GloVe optimizan

las incrustaciones directamente, de manera que el producto de dos vectores sea igual al logaritmo de la cantidad de veces que ambas palabras aparecen juntas en el texto. Además, los modelos GloVE se enfocan en aprender las probabilidades de co-ocurrencia de palabras completas, mientras que FastText trabaja con sub-palabras. Las arquitecturas propuestas (L-M4C y L-CNMT) extraen un conjunto de N tokens OCR, de la misma forma en que las arquitecturas originales lo hace, sin embargo, L-M4C y L-CNMT reemplazan la representación original de FastText con vectores GloVe. Es decir, las versiones ligeras extraen vectores 300-dimensionales para cada uno de los tokens OCR. Para L-M4C y L-CNMT, el modelo pre-entrenado de GloVe tiene un tamaño aproximado de 0.49 gigabytes, lo que podría reducir la cantidad de memoria utilizada por el codificador hasta en un 94 % (si se carga el modelo completo). El procesador semántico de OCR que se propone (con vectores GloVe) es basado en diccionarios, lo que significa que se entrena sobre un vocabulario fijo, que, para L-M4C y L-CNMT contiene un total de 75,501 tokens.

Incrustación de texto: BERT versus BERT destilado

El otro componente que se busca aligerar es el módulo de incrustación de texto, mismo que está presente tanto en M4C-Captioner como en CNMT. Los módulos de incrustación de texto de ambas arquitecturas son modelos pre-entrenados de la biblioteca de HuggingFace [Wolf et al., 2019] con identificador: *bert-base-uncased*, como se puede observar en las Figuras 4.3 y 4.4. El modelo original (*bert-base-uncased*) consiste de 12 capas, y tiene un total de 110 millones de parámetros entrenables. La propuesta de este trabajo de tesis consiste en reemplazar el modelo BERT original con una versión destilada del mismo, llamada DistilBERT [Sanh et al., 2019], que también cuenta con un modelo pre-entrenado en HugginFace (identificador: *distilbert-base-cased*). El modelo destilado es una arquitectura con 6 capas y un total de 65 millones de parámetros. El modelo original de BERT tiene un tamaño aproximado de 0.45 gigabytes, mientras que la versión destilada es de aproximadamente 0.25 gigabytes. Dado que los módulos de incrustación de texto de M4C-Captioner y CNMT están constituidos por solo algunas capas del modelo de BERT (solo usan 3 capas), el impacto de esta modificación en el uso final de memoria podría ser menos

marcado si se compara con el reemplazo del módulo de procesamiento de los tokens OCR.

4.3. Proponiendo alternativas multi-lenguaje

Uno de los objetivos de este trabajo consiste en buscar la viabilidad de las arquitecturas de DAICL para ser utilizadas en idiomas distintos al inglés, pues, como se mencionó anteriormente, todas las arquitecturas disponibles en la literatura hacen uso de componentes específicos para el idioma inglés. En esta sección se describe el proceso de diseño para una arquitectura llamada ML M4C-Captioner, que es una alternativa que elimina las limitantes de la arquitectura original (M4C-Captioner) respecto a su uso con idiomas distintos al inglés. Igualmente, se incluye el proceso de traducción automática del conjunto de datos TextCaps al idioma español.

4.3.1. Hacia arquitecturas multilingües: ML M4C-Captioner

En la Sección 4.1.3 se describió la arquitectura base propuesta por los autores del conjunto TextCaps. Siendo M4C-Captioner la arquitectura mejor documentada y con más referencias en la literatura, se decidió que las modificaciones para diseñar la primera arquitectura bilingüe estarían basadas en dicha arquitectura.

Los detalles sobre la arquitectura M4C-Captioner se pueden encontrar en la Sección 4.1.3, para obtener información aún más detallada se pueden consultar [Sidorov et al., 2020] y [Hu et al., 2020]. En esta sección se introduce la arquitectura *Multilingual M4C-Captioner* (ML M4C-Captioner).

La arquitectura M4C-Captioner está construida con un diseño *encoder-decoder* (codificador-decodificador). La alternativa multilingüe que se propone modifica la parte codificadora de M4C-Captioner, principalmente se busca remover los módulos que generan dependencias con el idioma inglés, como lo son: el procesador semántico de los tokens OCR y el módulo de incrustación de texto, ambos componentes están resaltados en verde en la Figura 4.1. Para presentar una explicación más clara de los cambios realizados a las arquitecturas, se introduce la Figura 4.5, que contiene ambas versiones del bloque codificador de la arquitectura, a la izquierda se encuentra el

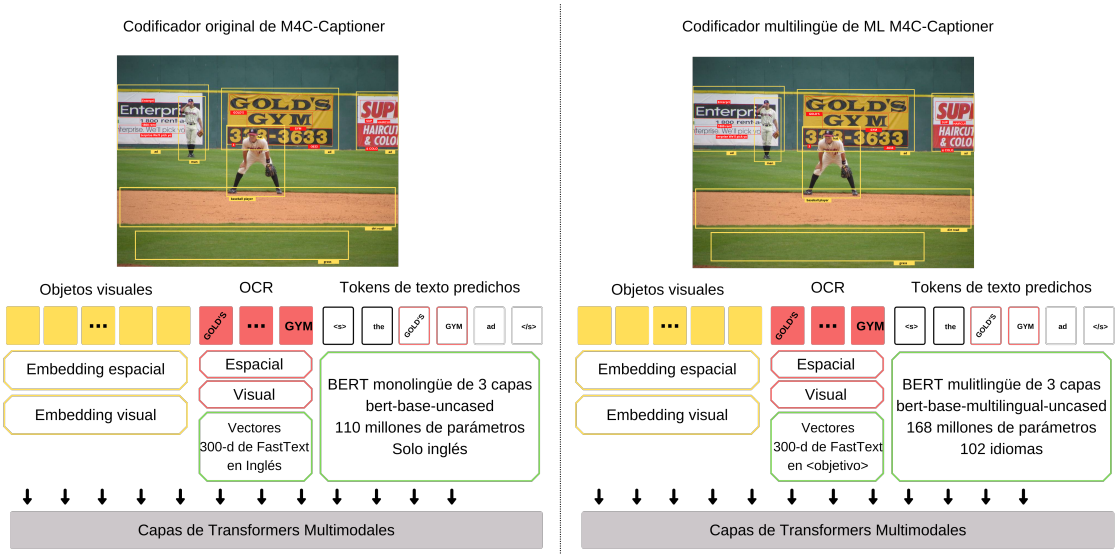


Figura 4.5: Comparación de los codificadores de ambas arquitecturas. El codificador original (izquierda) y el codificador multilingüe (derecha). Los bloques en verde resaltan las principales diferencias entre ambos codificadores.

codificador de M4C-Captioner y el de ML M4C-Captioner se encuentra a la derecha.

Anteriormente se mencionó que las principales diferencias entre ambas arquitecturas radican en la parte codificadora, y están resaltadas con color verde en las Figuras 4.1 y 4.5, dichos bloques corresponden al procesamiento de las características semánticas del OCR y al módulo de incrustación de texto.

Módulo de incrustación de texto. Mientras que el codificador original está limitado al idioma inglés porque su módulo de incrustación de texto es una versión pre-entrenada únicamente con inglés (por lo tanto es monolingüe) de BERT, el diseño que se propone reemplaza dicho módulo con una versión multilingüe (pre-entrenada con textos en 102 idiomas) de BERT [3]. La versión multilingüe de BERT fue entrenada con textos de las 102 Wikipedias con más contenido, y puede ajustarse (*fine-tune*) con facilidad para funcionar con cada uno de esos idiomas.

Módulo de procesamiento semántico del OCR. Otra de las limitantes del codificador original se encuentra en el bloque que procesa la información semántica de los tokens OCR. Originalmente, dicho bloque se compone únicamente por vectores

[3] https://github.com/google-research/bert/blob/master/multilingual.md

pre-entrenados de FastText para el idioma inglés. En la alternativa que se propone, dichos vectores pre-entrenados pueden seleccionarse de un total de 157 idiomas pre-entrenados[4], sin embargo, los idiomas utilizables se ven limitados únicamente a aquellos idiomas también disponibles en el modelo multilingüe de BERT. Es decir, la arquitectura ML M4C-Captioner es capaz de realizar entrenamiento e inferencia con todos los idiomas que están disponibles, tanto en la versión multilingüe de BERT como en los vectores pre-entrenados de FastText.

4.3.2. Traducción del conjunto de datos

Para evaluar la capacidad de la arquitectura ML M4C-Captioner para entrenar e inferir en un idioma distinto al inglés, se decidió traducir el conjunto de datos TextCaps al idioma español. Dicha traducción se realizó de forma sintética, es decir, se utilizaron algoritmos de traducción automática para realizar la tarea.

La traducción se realizó utilizando la biblioteca de *Transformers* de HugginFace [Wolf et al., 2019], específicamente, se utilizó el modelo *Helsinki-NLP/opus-mt-en-es*, pre-entrenado para resolver la tarea de traducción automática inglés-español. El modelo mencionado consiste de una versión ajustada de la arquitectura BART [Lewis et al., 2019], entrenada sobre el *Open Parallel Corpus*[5] para traducción de inglés a español. El modelo *Helsinki-NLP/opus-mt-en-es* obtuvo un puntaje de 54.9 puntos BLEU en el *Tatoeba Translation Challenge* [Tiedemann, 2020].

La Figura 4.6 muestra tres ejemplos del resultado de la traducción automática del conjunto de datos. Se incluye la imagen, la anotación en inglés y la versión traducida automáticamente al español. El excelente desempeño de los modelos de traducción automática basados en aprendizaje profundo [Tan et al., 2020] (como BART), permite que las traducciones se acerquen a una anotación humana, por ejemplo, en la imagen 1, todo el texto se tradujo al español excepto el nombre de la marca, es decir, el modelo es capaz de identificar entidades nombradas para así evitar su traducción. Otro ejemplo se puede encontrar en la imagen 2, el modelo también fue capaz de detectar que “Dealing in Desire” es el nombre del libro, y por

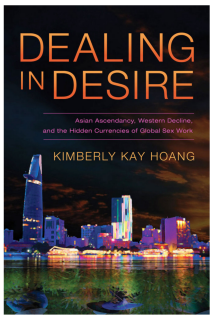
[4]https://fasttext.cc/docs/en/crawl-vectors.html
[5]https://opus.nlpl.eu/

1

Inglés: A green colored bottle of Trapiche Melodias Chardonnay sitting on a wood shelf.

Español: Una botella de color verde de Trapiche Melodias Chardonnay sentado en un estante de madera.

2



Inglés: The cover of a book named Dealing in Desire.

Español: La portada de un libro llamado Dealing in Desire.

3

Inglés: A picture with images along with instructions on how to make a Mojito.

Español: Una imagen con imágenes junto con instrucciones sobre cómo hacer un Mojito.

Figura 4.6: Tres ejemplos sobre el resultado de la traducción automática de TextCaps al idioma español.

lo tanto, no debe ser traducido.

CAPÍTULO 5

Resultados experimentales

Este capítulo detalla la configuración experimental para cada uno de los dos objetivos planteados, además, cada sección (correspondiente a un objetivo) contiene los resultados para esa configuración experimental. Primero, se presenta la Sección 5.1, llamada "Hacia arquitecturas más ligeras en memoria", en donde se incluye toda la información concerniente al desarrollo de alternativas más ligeras en memoria, se incluye la información sobre los experimentos y también los resultados. Después, se presenta la Sección 5.2, titulada "Hacia arquitecturas multi-lenguaje", en donde se incluye toda la información que respecta al desarrollo de arquitecturas que sean viables para funcionar en idiomas distintos al inglés. Al igual que en la Sección 5.1, en la Sección 5.2 se incluyen los detalles sobre los experimentos y los resultados obtenidos.

5.1. Hacia arquitecturas más ligeras en memoria

Esta sección contiene la información relacionada con los experimentos y los resultados que respectan al objetivo específico "Proponer arquitecturas reducidas en cantidad de parámetros y en uso de memoria".

5.1.1. Configuración experimental

A continuación se incluye la información necesaria para seguir el proceso experimental, desde el diseño hasta la configuración para cada arquitectura. Primero, se describen las configuraciones de software y hardware. Finalmente, se incluye una tabla con los detalles de cada arquitectura, así como su configuración y los modelos pre-entrenados que utiliza.

Hardware y software utilizado

Software. Todo el código para estos experimentos está escrito en Python y el *framework* de aprendizaje profundo principal es PyTorch, sin embargo, todos los experimentos están diseñados sobre el *Multimodal Framework*[1] (MMF) de FAIR. Además, generó un repositorio de GitHub, en donde se puede encontrar todo el código, los scripts y las configuraciones para cada modelo presentado en esta sección: `https://github.com/gallardorafael/multilingual-mmf`. La versión destilada del modelo de BERT se puede encontrar en la biblioteca de HugginFace[2]. Los vectores pre-entrenados de GloVe se pueden encontrar en: `https://nlp.stanford.edu/projects/glove/`.

Hardware para entrenamiento de modelos. El entrenamiento de todas las arquitecturas para este conjunto de experimentos se llevó a cabo en una instancia de *Azure Machine Learning* en la nube. Específicamente, se utilizó una instancia Standard-NC6, que incluye un procesador de 6 núcleos, 56 gigabytes de memoria y un acelerador gráfico NVIDIA K80 con 11 gigabytes de memoria.

Hardware para inferencia. Los procedimientos de inferencia se llevaron a cabo en una computadora personal, modelo Dell G3 3590, con 20 gigabytes de memoria, un procesador Intel Core i7-9750H y un acelerador gráfico NVIDIA GTX 1660 Ti con diseño Max-Q.

[1] https://mmf.sh/
[2] https://huggingface.co/

Tabla 5.1: Resumen de todos los diseños propuestos. La arquitectura marcada con el símbolo † corresponde al modelo base del reto, y se incluye únicamente como referencia.

Arquitectura	Módulo de incrustación de texto	Procesador de contexto OCR	Sistema OCR
CNMT-DF	DistilBERT	FastText 300-d	Rosetta + CRAFT + ABCNet
L-CNMT-DG	DistilBERT	GloVe 300-d	Rosetta + CRAFT + ABCNet
CNMT-BF	BERT	FastText 300-d	Rosetta
L-M4C-DG	DistilBERT	GloVe 300-d	Rosetta
L-M4C-BG	BERT	GloVe 300-d	Rosetta
M4C-DF	DistilBERT	FastText 300-d	Rosetta
† M4C-BF	BERT	FastText 300-d	Rosetta

Arquitecturas y configuraciones

Se realizaron diversos experimentos, de modo que sea posible probar la efectividad de las modificaciones propuestas. El impacto de cada componente modificado se midió mediante una comparación directa de la arquitectura con el componente original, versus la arquitectura con el componente modificado (un componente a la vez). En la Tabla 5.1 se puede encontrar el nombre de cada arquitectura, así como su respectiva configuración.

El modelo base del reto TextCaps está nombrado M4C-BF, la "B" indica que el modelo utiliza BERT para codificar el texto, mientras que la "F" indica que el procesador de contexto OCR está basado en FastText, es decir, las siglas "BF" indican que este modelo en específico utiliza un modelo BERT y vectores de FastText, lo que corresponde al diseño original. También se incluye el modelo CNMT-BF, que corresponde al modelo original de CNMT, es decir, que utiliza el modelo BERT original junto con vectores FastText, de manera similar a M4C-BF.

Para evaluar el impacto de reemplazar BERT con una versión destilada del mismo, se presenta la configuración con nombre M4C-DF, en donde la única variación respecto a M4C-BF se encuentra en el módulo de incrustación de texto. Además, para medir el impacto de reemplazar los vectores FastText con vectores GloVe, se presenta la arquitectura L-M4C-BG, que únicamente varía, respecto de M4C-BF, en el procesador de contexto OCR, pues L-M4C-BG utiliza vectores GloVe en vez de FastText.

Por último, todos los nombres que inicien con una "L" corresponden a las arquitecturas propuestas en este trabajo, que tienen como principal objetivo reducir el tamaño que ocupan en memoria. Sin embargo, las configuraciones CNMT-DF

y M4C-DF también son resultado de este trabajo, pero se decidió no presentarlas como arquitecturas ligeras pues el impacto de reemplazar el modelo BERT original con la versión destilada es prácticamente imperceptible a la hora de realizar los experimentos.

Evaluación de resultados

Los autores del reto TextCaps establecieron un servidor en la plataforma EvalAI[3], de modo que los resultados se evaluaran de forma justa. En este servidor de evaluación, cada equipo participante puede someter hasta cinco archivos de predicción (formateados según los requisitos del reto), posteriormente, los archivos son evaluados tomando como referencia las anotaciones humanas no públicas para el conjunto de prueba de TextCaps.

Las métricas utilizadas por el servidor de evaluación son: BLEU-4, METEOR, ROUGE-L, SPICE, y CIDEr. Para más información sobre las métricas, consultar la Sección 2.3.3.

5.1.2. Resultados

Originalmente se pensó que utilizar una versión destilada del modelo de codificación de texto (BERT) decrementaría el rendimiento del modelo, sin embargo, como se puede observar en la Tabla 5.2, el reemplazo del modelo original de BERT (M4C-BF) con una versión destilada (M4C-DF) incrementó ligeramente el rendimiento del modelo (sobre el conjunto de validación) en tres de las cinco métricas utilizadas: BLEU-4 (+0.1), ROUGE-L (+0.2), SPICE (+0.6), y CIDEr (+0.3). Dado el comportamiento mencionado, se decidió que las dos arquitecturas basadas en CNMT utilizarían directamente el modelo destilado de BERT.

Además, el impacto de utilizar vectores GloVe en vez de vectores FastText se midió de manera similar. Las arquitecturas L-M4C-BG y M4C-BF utilizan el modelo original de BERT, sin embargo, la arquitectura L-M4C-BG se entrenó utilizando un enfoque de procesamiento de OCR basado en vectores GloVe, y, como se puede observar en la

[3]https://eval.ai/

Tabla 5.2: Resultados de reemplazar el modelo original de BERT con su versión destilada, en arquitecturas basadas en M4C-Captioner. Los mejores puntajes están escritos en negritas.

Conjunto de validación de TextCaps

Arquitectura	BLEU-4	METEOR	ROUGE-L	SPICE	CIDEr	Tamaño del modelo BERT
M4C-DF	22.3	**22.3**	45.8	**15.6**	**89.4**	**0.25 gigabytes**
M4C-BF	**23.4**	21.8	**46**	15	89.1	0.45 gigabytes

Tabla 5.3: Resultados de reemplazar los vectores FastText originales con vectores GloVe, en arquitecturas basadas en M4C-Captioner. Los mejores puntajes están escritos en negritas.

Conjunto de validación de TextCaps

Arquitectura	BLEU-4	METEOR	ROUGE-L	SPICE	CIDEr	Tamaño del procesador de OCR
L-M4C-BG	22.5	**22.2**	45.8	**15.5**	**89.4**	**0.49 gigabytes**
M4C-BF	**23.4**	21.8	**46**	15	89.1	8.5 gigabytes

Tabla 5.3, nuestra propuesta (L-M4C-BG) superó ligeramente al modelo original de M4C-BF en tres de las cinco métricas: METEOR (+0.2), ROUGE-L (+0.2), SPICE (+0.5), y CIDEr (+0.3), mientras que se redujo la cantidad de memoria utilizada por el procesador de OCR en un 94 % (de 8.5 a 0.49 gigabytes), tanto en la etapa de entrenamiento como de inferencia.

Como se puede observar en la Tabla 5.3, la reducción en el uso de memoria (que es resultado del reemplazo del procesador semántico de OCR) es una aportación importante, es por esto que se decidió medir también el impacto de esta modificación en las arquitecturas basadas en CNMT, por lo tanto, se proponen dos diseños: uno con vectores FastText y otro con vectores GloVe, sin embargo, ambos modelos utilizarán la versión destilada de BERT para las tareas de incrustación de texto. Los resultados de estos experimentos pueden encontrarse en la Tabla 5.4.

En el caso de los experimentos reportados en la Tabla 5.4, los experimentos se

Tabla 5.4: Resultados de reemplazar los vectores FastText originales con vectores GloVe, en arquitecturas basadas en CNMT. Los mejores puntajes están escritos en negritas.

Conjunto de validación de TextCaps

Arquitectura	BLEU-4	METEOR	ROUGE-L	SPICE	CIDEr	Tamaño del procesador de OCR
CNMT-DF	**22.8**	**22.7**	**46.3**	15.9	**96.3**	8.5 gigabytes
L-CNMT-DG	22.7	22.5	46.2	**16.1**	94.9	**0.49 gigabytes**

comportaron distinto; en este caso, el reemplazo (con GloVe) de los vectores FastText redujo el rendimiento del modelo en la mayoría de las métricas (ver métricas en Tablas 5.3 y 5.4). Por ejemplo, en la Tabla 5.4 se puede observar que el modelo L-CNMT-DG únicamente superó a CNMT-DF en una de las cinco métricas, mientras que el enfoque basado en vectores FastText (CNMT-DF) fue ligeramente mejor que L-CNMT-DG en: BLEU-4 (+.1), METEOR (+.2), ROUGE-L (+.1), y CIDEr (+1.4). EL modelo L-CNMT-DG superó a CNMT-DF únicamente en la métrica SPICE (+0.3).

Sin embargo, el modelo L-CNMT-DG sigue ocupando un espacio en memoria mucho menor al de CNMT-DF, pues, como se mencionó anteriormente, el módulo de incrustación de OCR basado en GloVe puede ahorrar hasta 94 % de la memoria (ver última columna de Tablas 5.3 y 5.4), cuando se le compara con la alternativa basada en vectores FastText.

5.1.3. Comparación con el Estado del Arte

Tanto las propuestas presentadas en este trabajo como aquellas disponibles en la literatura utilizan las cinco métricas descritas en la Sección 2. A continuación se presentará una comparación de las arquitecturas ligeras propuestas y las arquitecturas disponibles en la literatura.

Al igual que en el artículo original de TextCaps, la Tabla 5.5 presenta la comparación antes mencionada y, dado que la métrica CIDEr fue considerada como la métrica más importante en [Sidorov et al., 2020], los resultados en la Tabla 5.5 se presentan ordenados de manera descendente según el puntaje CIDEr.

Tanto el método TAP como LSTM-R obtuvieron puntajes que superan a las propuestas restantes. TAP superó a LSTM-R en tres de las cinco métricas, mientras que LSTM-R superó a TAP en dos de las cinco. Sin embargo, es importante resaltar que el enfoque principal de este trabajo no consiste en superar a las arquitecturas estado-del-arte en las métricas, sino encontrar alternativas que reduzcan el uso de memoria. Así, el modelo L-CNMT-DG se encuentra bien posicionada, cuando se le compara con otros métodos en la Tabla 5.5, obteniendo la quinta posición, superando a otros métodos como MMA-SR o AnC, mientras es capaz de ahorrar

Tabla 5.5: Comparación del rendimiento de las arquitecturas propuestas en este trabajo y los métodos estado-del-arte. La arquitectura marcada con el símbolo † es la base del reto TextCaps, mientras que los modelos marcados con el símbolo ⋆ indican alguna de las propuestas de este trabajo. Los mejores resultados están escritos en negritas.

		Conjunto de prueba de TextCaps				
Posición	Arquitectura	BLEU-4	METEOR	ROUGE-L	SPICE	CIDEr
1	TAP	21.86	**21.85**	45.65	**14.64**	**103.22**
2	LSTM-R	**22.93**	21.32	**46.11**	13.81	100.82
3	CNMT	20.1	20.94	44.41	13.55	93.24
4	SBD	20.16	20.3	44.23	12.82	89.63
5	⋆ L-CNMT-DG	19.11	20.66	43.72	13.46	88.24
6	MMA-SR	19.8	20.6	44.0	13.2	88.0
7	AnC	20.7	20.7	44.6	13.4	87.4
8	⋆ L-M4C-BG	18.84	20.22	43.41	13.05	81.24
9	† M4C-BF	18.86	19.77	43.23	12.77	80.99
10	⋆ L-M4C-DG	18.77	19.98	43.07	12.92	80.59

hasta 8 gigabytes en el módulo de procesamiento de OCR, comparado con el diseño original de la tercera posición (CNMT), logrando una reducción del 94 % del uso de memoria en dicho componente. Además, los modelos L-CNMT-DG y L-M4C-DG superaron al modelo base del reto (M4C-BF), aun cuando ambos enfoques están basados en vectores GloVe y, por ende, son mucho más eficientes en términos de memoria. Por otra parte, se tiene al modelo L-M4C-DG, que ocupa la última posición en la Tabla 5.5, y es incluso superado por el modelo base del reto, lo que indica que la combinación del modelo destilado de BERT con un módulo de procesamiento de OCR basado en GloVe reduce el rendimiento del modelo original (M4C-BF), este comportamiento se atribuye directamente al uso de *DistilBERT* con vectores GloVe de forma conjunta, pues los resultados de la Tabla 5.3 demuestran que el enfoque basado en GloVe superó al diseño basado en FastText. Además, como se puede observar en la Tabla 5.2, la implementación de la versión destilada de BERT en vez del modelo original incrementó el rendimiento del modelo, sin embargo, este comportamiento no se reprodujo cuando ambos módulos se utilizaron juntos. Por lo tanto, es necesario realizar más investigación para determinar por qué el uso de *DistilBERT* y GloVe mejoró el modelo de M4C cuando se implementaron separados, pero lo empeoraron cuando se utilizaron juntos.

5.2. Hacia arquitecturas multi-lenguaje

Esta sección contiene la información relacionada con los experimentos y los resultados que respectan al objetivo específico "Proponer arquitecturas que, con mínimas variaciones en su diseño, puedan resolver el problema para los idiomas inglés y español.". Es importante aclarar que aquí solo se detallan los experimentos para llevar a los modelos tradicionales hacia un enfoque multilingüe.

5.2.1. Configuración experimental

A continuación se incluye la información necesaria para seguir el proceso experimental, desde el diseño hasta la configuración para cada arquitectura. Primero, se describen las configuraciones de software y hardware. Finalmente, se incluye una tabla con los detalles de cada arquitectura, así como sus hiperparámetros.

Hardware y software utilizado

Software. Todo el código para estos experimentos está escrito en Python y el *framework* de aprendizaje profundo principal es PyTorch, sin embargo, todos los experimentos están diseñados sobre el *Multimodal Framework*[4] (MMF) de FAIR. Además, generó un repositorio de GitHub, en donde se puede encontrar todo el código, los scripts y las configuraciones para cada modelo presentado en esta sección: `https://github.com/gallardorafael/multilingual-mmf`. Tanto el modelo de traducción automática como el modelo de BERT multilingüe pertenecen a la biblioteca de HugginFace[5]. Por otra parte, los vectores pre-entrenados de FastText para el idioma español se encuentran disponibles en `https://fasttext.cc/`.

Hardware para traducción de TextCaps. Las tareas relacionadas con la traducción del conjunto de datos se realizaron en el Cluster IBM Power9 del Laboratorio Nacional de Supercómputo del Sureste de México[6]. Para acelerar el proceso, se hizo uso de una tarjeta gráfica (GPU, por las siglas en inglés) NVIDIA Tesla V100 con 16

[4]https://mmf.sh/
[5]https://huggingface.co/
[6]https://lns.buap.mx/

Tabla 5.6: Todas las arquitecturas entrenadas y evaluadas. Las configuraciones completas y los registros están disponibles en el repositorio del trabajo.

Modelo	Idioma de TextCaps	Vectores FastText	Modelo BERT	Tam. Vocab. BERT	Tam. Vocab. Modelo	Parámetros totales
m4c-captioner-zoo (Baseline)	English	English: wiki.en.bin	bert-base-uncased	30522	6736	92,185,168
m4c-captioner-local	English	English: wiki.en.bin	bert-base-uncased	30522	6736	92,185,168
en_ml-m4c-captioner	English	English: wiki.en.bin	bert-base-multilingual-uncased	105879	6736	150,059,344
es_ml-m4c-captioner	Spanish	Spanish: cc.es.300.bin	bert-base-multilingual-uncased	105879	7207	150,421,543

gigabytes de memoria. El proceso de traducción tomó, aproximadamente, un tiempo de 42 horas de GPU en la infraestructura ya mencionada. Es importante mencionar que el proceso de traducción puede ser ejecutado en la mayoría de computadoras personales con más de 4 GB de memoria, incluyendo aquellas sin acelerador gráfico, sin embargo, el tiempo de ejecución podría incrementar en gran medida. Se recomienda utilizar aceleradores gráficos con soporte CUDA [7].

Hardware para entrenamiento de modelos. El entrenamiento de todas las arquitecturas para este conjunto de experimentos se llevó a cabo en una instancia de *Azure Machine Learning* en la nube. Específicamente, se utilizó una instancia Standard-NC6, que incluye un procesador de 6 núcleos, 56 gigabytes de memoria y un acelerador gráfico NVIDIA K80 con 11 gigabytes de memoria.

Hardware para inferencia. Los procedimientos de inferencia se llevaron a cabo en una computadora personal, modelo Dell G3 3590, con 20 gigabytes de memoria, un procesador Intel Core i7-9750H y un acelerador gráfico NVIDIA GTX 1660 Ti con diseño Max-Q.

Arquitecturas y configuraciones

La lista de arquitecturas y su respectiva configuración se puede observar en la Tabla 5.6. La tabla contiene el nombre del modelo, el idioma del conjunto con el que se entrena, el nombre de los vectores de FastText, el nombre del modelo de BERT, el tamaño del vocabulario del modelo BERT, el tamaño del vocabulario del modelo M4C y la cantidad total de parámetros entrenables.

Todas las arquitecturas fueron entrenadas con la versión en inglés del conjunto

[7] https://developer.nvidia.com/cuda-toolkit

TextCaps, para evitar algún tipo de sesgo, se entrenaron todas con los mismos hiperparámetros. El entrenamiento de la arquitectura *es-ml-m4c-captioner* fue diferente, en este caso, se realizó el entrenamiento con un tamaño de lote igual a 8 y una frecuencia de actualización de gradientes de 8, de modo que dichos gradientes se acumulen hasta haber procesado un total de 64 muestras, dicha acumulación de muestras emula un tamaño de lote de 64, utilizado en la arquitectura base original. La primera columna de la tabla corresponde al modelo pre-entrenado por los autores originales de M4C-Captioner, disponible a través de MMF. La segunda fila corresponde a la misma arquitectura, entrenada localmente sobre la versión en inglés de TextCaps pero con los hiperparámetros definidos anteriormente. La tercera fila corresponde a la versión de ML M4C-Captioner entrenada sobre TextCaps en inglés, esta versión ya incluye las modificaciones mencionadas en la Sección 4.1.3. La última fila corresponde al modelo de ML M4C-Captioner entrenada con la versión en español de TextCaps.

Es importante mencionar que, dada la naturaleza "privada" del conjunto de prueba de TextCaps, la única forma de realizar la evaluación desde un enfoque multilingüe radica en trabajar únicamente con el conjunto de validación. Es decir, la única forma de obtener las descripciones *ground truth* para un conjunto se encuentra en la traducción de los conjuntos de entrenamiento y validación, en este caso, el conjunto de entrenamiento se utilizará para entrenar a los modelos y el de validación para medir su desempeño. Los resultados cuantitativos correspondientes a este objetivo específico se enfocan en el desempeño de los modelos sobre el conjunto de validación.

Evaluación de resultados

Para evaluar los resultados de esta sección, se comparan las descripciones generadas por los modelos para el conjunto de validación contra las descripciones traducidas automáticamente. Ambos archivos están disponibles localmente, por lo tanto, la evaluación se realiza de forma local.

Los resultados cuantitativos se miden con las métricas utilizadas por el servidor de evaluación: BLEU-4, METEOR, ROUGE-L, SPICE, y CIDEr. Para más información

Tabla 5.7: Rendimiento de cada modelo sobre el conjunto de validación de TextCaps, para las versiones en inglés y español. Los mejores puntajes están escritos con negritas.

			Conjunto de validación de TextCaps (inglés)				
Modelo	Modelo FastText	Modelo BERT	BLEU-4	METEOR	ROUGE-L	SPICE	CIDEr
TAP	English: wiki.en.bin	bert-base-uncased	**25.8**	**23.8**	**47.9**	**17.1**	**109.2**
m4c-captioner-zoo (Baseline)	English: wiki.en.bin	bert-base-uncased	23.4	21.8	46.0	15.0	89.1
m4c-captioner-local	English: wiki.en.bin	bert-base-uncased	23.1	22.3	46.1	15.7	90.4
en-ml-m4c-captioner	English: wiki.en.bin	bert-base-multilingual-uncased	22.4	22.2	46.0	15.6	88.7
			Conjunto de validación de TextCaps (español)				
Model	FastText	Text BERT	BLEU-4	METEOR	ROUGE-L	SPICE	CIDEr
es-ml-m4c-captioner	Spanish: cc.es.300.bin	bert-base-multilingual-uncased	21.0	21.6	41.6	6.1	63.2

sobre las métricas, consultar la Sección 2.3.3.

5.2.2. Resultados

La Tabla 5.7 contiene los resultados de todos los modelos diseñados, entrenados y evaluados para satisfacer el objetivo que concierne al desarrollo de modelos multilenguaje. En dicha tabla, el mejor método para el idioma inglés es TAP, sin embargo, el segundo mejor modelo es la alternativa entrenada localmente de M4C-Captioner (m4c-captioner-local), misma que superó al modelo pre-entrenado (m4c-capioner-zoo) de la misma arquitectura por 1.3 puntos CIDEr. La arquitectura es la misma, pero la variación en los puntajes se debe a la configuración del proceso de entrenamiento, es decir, en la configuración local se utilizó un tamaño de lote más pequeño, se utilizó la acumulación de gradientes y se incrementó el número máximo de actualizaciones. En general, se puede describir el impacto de la configuración local como sigue: BLEU-4 (-0.3), METEOR (+0.5), ROUGE-L (+0.1), SPICE (+0.7), y CIDEr (+1.3). El modelo entrenado localmente superó a la versión original en cuatro de las cinco métricas, lo que ayuda a descartar que las configuraciones locales tengan algún impacto negativo en los puntajes de los modelos restantes.

Una vez que el posible impacto negativo de las configuraciones locales se descartó, se puede proceder a analizar los resultados de la arquitectura *Multilingual M4C-Captioner*, entrenada con la versión en inglés de TextCaps (en-ml-m4c-captioner). Esta arquitectura utiliza la versión multilingual de BERT para incrustar texto y utiliza vectores FastText, de forma similar a como lo hacen los métodos TAP, m4c-captioner-local y m4c-captioner-zoo. Lo anterior es útil para medir el impacto del modelo de incrustación de texto en el rendimiento y en las métricas. El impacto (medido en

Figura 5.1: Tres ejemplos de las descripciones generadas por el diseño bilingüe: ml-m4c-captioner

porcentaje de mejora relativa), de reemplazar la versión monolingüe de BERT en M4C (m4c-captioner-local) con una versión multilingüe (en-ml-m4c-captioner) se puede resumir de la siguiente forma: BLEU-4 (-3 %), METEOR (-0.44 %), ROUGE-L (-0.21 %), SPICE (-0.6 %), y CIDEr (-1.8 %). Sin embargo, un ligero decremento en las métricas era de esperarse, pues la capacidad del módulo de incrustación de texto incrementó de manera significativa (101 lenguajes adicionales), mientras únicamente utiliza un 52 % más parámetros que la versión monolingüe de BERT (es decir, se incrementó de 110 millones a 168 millones de parámetros).

Los experimentos sobre el conjunto de datos traducido únicamente se proponen como una prueba de concepto, sobre todo para mostrar las capacidades bilingües de ML M4C-Captioner. En la Figura 5.1 se presentan tres ejemplos de las descripciones generadas por ML M4C-Captioner, las descripciones en inglés fueron generadas por el modelo *en-ml-m4c-captioner* mientras que las descripciones en español fueron generadas por el modelo *es-ml-m4c-captioner*, en dicha imagen, las palabras subrayadas corresponden a tokens tomados directamente del OCR, las palabras restantes provienen del vocabulario del modelo.

Por otra parte, los puntajes obtenidos para el conjunto en español son notablemente menores que los obtenidos para el conjunto en inglés, sin embargo, este comportamiento era esperado, pues es muy probable que el conjunto traducido automáticamente haya heredado los sesgos y errores del modelo BART. Además, los

tokens OCR disponibles en TextCaps pueden pertenecer a distintos lenguajes, lo que añade un nivel extra de complejidad para realizar la traducción del conjunto. Aun así, aunque el modelo en español no se puede comparar con algún estado-del-arte (pues es el primer modelo en resolver dicha tarea), los puntajes obtenidos son aceptables (ver Tabla 5.7): 21 para BLEU-4, 21.6 para METEOR, 41.6 en ROUGE-L, 6.1 en SPICE y 63.2 en CIDEr, y establecen las bases para el desarrollo de este tipo de arquitecturas en el idioma español.

CAPÍTULO 6

Conclusiones

Este trabajo de tesis tenía dos objetivos; el primero consistía en el diseño de alternativas que fueran más ligeras en memoria, y el segundo, en desarrollar arquitecturas que fueran capaces de trabajar con idiomas distintos al inglés, sin la necesidad de modificar completamente el diseño.

Como parte del primer objetivo, en este trabajo se presentaron cinco distintas arquitecturas para resolver el problema de la DAICL, que son alternativas a los diseños estado-del-arte (como M4C-Captioner y CNMT). La principal contribución de dichas arquitecturas se encuentra en el reemplazo del módulo de procesamiento semántico de los tokens OCR, la propuesta consistió en reemplazar el módulo original basado en vectores FastText, por una alternativa más ligera basada en vectores GloVe. Lo anterior demostró reducir drásticamente la cantidad de memoria utilizada por dicho módulo, tanto en entrenamiento como en inferencia. De los cinco modelos entrenados, tres de ellos pueden considerarse alternativas más ligeras en memoria, pues pueden reducir el coste (en memoria) del procesador de OCR hasta en un 94 %, cuando se compara con el procesador basado en FastText. Además, uno de los modelos (L-CNMT-DG) propuestos en este trabajo superó a las arquitecturas MMA-SR y AnC, obteniendo la quinta posición en la tabla de posiciones para

el conjunto de prueba de TextCaps. También es importante mencionar que dos de los modelos (L-CNMT-DG, L-M4C-BG) presentados en esta tesis superan el modelo base (M4C-BF) propuesto para TextCaps, obteniendo puntajes altos aun cuando son mucho más ligeros en términos de memoria. En general, los resultados se pueden considerar exitosos, pues se ha demostrado que las arquitecturas con diseños enfocados en el rendimiento pueden alcanzar puntajes similares al estado-del-arte, e incluso, superarlo.

Por otra parte, se tiene el objetivo que buscaba dirigir a las arquitecturas actuales hacia un enfoque multi-lenguaje. En este trabajo, se propuso una arquitectura multilingüe, pero la prueba de concepto que se presentó únicamente cubre sus capacidades bilingües, es decir, se trabajó con los idiomas inglés y español. La arquitectura propuesta es llamada *Multilingual M4C-Captioner*, y demostró ser capaz de alcanzar puntajes cercanos al estado-del-arte para el idioma inglés, incluso cuando el módulo de incrustación de texto está diseñado para trabajar con 102 idiomas (el modelo original únicamente se enfoca en el idioma inglés). El mayor problema radica en la disponibilidad de los datos, pues la traducción automática del conjunto de datos conlleva que el modelo entrenado herede los errores y problemas del sistema de traducción automática. Sin embargo, la mayoría de las descripciones generadas son informativas y las métricas reportadas indican un buen rendimiento en general. Siendo este el primer trabajo que resuelve el problema para un idioma distinto al inglés, se puede considerar un éxito, pues los modelos obtuvieron puntajes dentro del rango que se obtuvieron para el idioma inglés, y las descripciones generadas para el inglés y el español son claras e informativas.

6.1. Aportaciones

A continuación se enlistan las principales aportaciones que este trabajo brinda a la literatura:

- El primer acercamiento al desarrollo de arquitecturas para DAICL, que rindan de forma similar al estado-del-arte, pero que son mucho menos costosas en términos de memoria.

- Un método para diseñar arquitecturas para DAICL multi-lenguaje, que requiere un mínimo esfuerzo para generar modelos en distintos idiomas.
- El primer sistema de DAICL específicamente diseñado para trabajar con el idioma español.

6.2. Trabajo futuro

Primero, se necesita más investigación para mejorar los resultados de las arquitecturas enfocadas en la reducción del uso de memoria, por ejemplo, la investigación de los hiperparámetros óptimos, o incluso la reducción de la cantidad de parámetros entrenables en el modelo. También, las técnicas de pre-entrenamiento como aquellas usadas por el equipo de TAP, o las técnicas de aumento de datos y características en el conjunto de datos (como las usadas por CNMT) han demostrado incrementar de forma importante el rendimiento del modelo, sin necesidad de incrementar directamente la capacidad del modelo; dichas técnicas, combinadas o diseñadas desde un enfoque de rendimiento, podrían reducir fuertemente las necesidades de hardware para este tipo de problemas. Estos experimentos quedan como trabajo futuro.

Además, hay mucho trabajo por realizar si se desea empujar al área hacia la investigación desde una perspectiva multilingüe. Por ejemplo, para atacar el problema desde la raíz, primero es necesario obtener datos con una cantidad mucho menor de errores, ya sea anotando los datos manualmente, o desarrollando mejores sistemas de traducción automática. Otra línea de investigación interesante consiste en entrenar los modelos únicamente con el conjunto de datos en inglés, pero que la arquitectura integre un sistema de traducción efectivo, de esta manera, un único modelo podría generar las descripciones, mientras otro es el encargado de traducirlas.

En general, el trabajo futuro consiste en mejorar los sistemas actuales, o desarrollar nuevas propuestas, que se acerquen aún más al desempeño humano, sin perder de vista los objetivos de este trabajo de tesis.

6.3. Publicaciones

Esta tesis es el producto de diversos trabajos relacionados con el aprendizaje profundo aplicado al procesamiento del lenguaje natural y a la visión computacional. A continuación se presenta una lista de las publicaciones derivadas:

- Towards Multilingual Image Captioning Models that Can Read (2021). Rafael Gallardo García, Beatriz Beltrán Martínez, Carlos Hernández Gracidas, Darnes Vilariño Ayala. **Springer LNAI – Lecture Notes in Artificial Intelligence**. Presentado en: *20th Mexican International Conference on Artificial Intelligence*.

- Searching for Memory-Lighter Architectures for OCR-Augmented Image Captioning (2021). Rafael Gallardo García, Beatriz Beltrán Martínez, Carlos Hernández Gracidas, Darnes Vilariño Ayala. Aceptado en: *8th International Symposium on Language & Knowledge Engineering* para presentación oral. Para ser publicado en **Journal of Intelligent and Fuzzy Systems**.

- Evaluación del modelo neuronal de atención visual en la descripción automática de imágenes en Español (2020). Rafael Gallardo García, Beatriz Beltrán Martíne, Darnes Vilariño Ayala. **Research in Computer Science** Vol. 149(8). Presentado en: *12th Mexican Congress of Artificial Intelligence*.

Bibliografía

[Amirian et al., 2019] Amirian, S., Rasheed, K., Taha, T. R., and Arabnia, H. R. (2019). A short review on image caption generation with deep learning. In *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, pages 10–18. The Steering Committee of The World Congress in Computer Science. 1.5, 3.1

[Anderson et al., 2016] Anderson, P., Fernando, B., Johnson, M., and Gould, S. (2016). Spice: Semantic propositional image caption evaluation. In *European conference on computer vision*, pages 382–398. Springer. 2.3.3

[Anderson et al., 2018] Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., and Zhang, L. (2018). Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086. 1.5

[Baek et al., 2019] Baek, Y., Lee, B., Han, D., Yun, S., and Lee, H. (2019). Character region awareness for text detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9365–9374. 4.2.1

[Baltrušaitis et al., 2018] Baltrušaitis, T., Ahuja, C., and Morency, L.-P. (2018). Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443. 2.2.5

[Bender and Koller, 2020] Bender, E. M. and Koller, A. (2020). Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics. 2.2.5

[Biten et al., 2019] Biten, A. F., Tito, R., Mafla, A., Gómez, L., Rusiñol, M., Valveny, E., Jawahar, C., and Karatzas, D. (2019). Scene text visual question answering. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4290–4300. 3.1

[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. 2.2.6

[Borisyuk et al., 2018] Borisyuk, F., Gordo, A., and Sivakumar, V. (2018). Rosetta: Large scale system for text detection and recognition in images. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 71–79. 3.1, 4.1.2, 4.2.1

[Chai et al., 2021] Chai, J., Zeng, H., Li, A., and Ngai, E. W. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, page 100134. 2.3.1

[Chen et al., 2019] Chen, Y.-C., Li, L., Yu, L., El Kholy, A., Ahmed, F., Gan, Z., Cheng, Y., and Liu, J. (2019). Uniter: Learning universal image-text representations. 1.5

[Denkowski and Lavie, 2014] Denkowski, M. and Lavie, A. (2014). Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380. 2.3.3

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. 2.1.1, 2.2.4, 4.1.3

[Dong et al., 2019] Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M., and Hon, H.-W. (2019). Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*. 2.2.4

[Efron, 1969] Efron, R. (1969). What is perception. 1

[Farhadi et al., 2010] Farhadi, A., Hejrati, M., Sadeghi, M., Young, P., Rashtchian, C., Hockenmaier, J., and Forsyth, D. (2010). Every picture tells a story: Generating sentences from images. In *ECCV*. 1.5

[Géron, 2019] Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media. 2.1, 2.1.1, 2.1.1, 2.1.1, 2.1.1, 2.1.2

[Goldberg and Hirst, 2017] Goldberg, Y. and Hirst, G. (2017). *Neural Network Methods in Natural Language Processing*. Morgan, Claypool Publishers. 2.2.6

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep Learning*, volume 1. MIT press Cambridge. (document), 2.1, 2.1.3, 2.1, 2.2, 2.2, 2.2.3, 2.2.4, 2.3.1, 2.3.2

[He et al., 2018] He, T., Tian, Z., Huang, W., Shen, C., Qiao, Y., and Sun, C. (2018). An end-to-end textspotter with explicit alignment and attention. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5020–5029. 3.1

[Hessel et al., 2021] Hessel, J., Holtzman, A., Forbes, M., Bras, R. L., and Choi, Y. (2021). Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*. 2.3.3

[Hodosh et al., 2013] Hodosh, M., Young, P., and Hockenmaier, J. (2013). Framing image description as a ranking task: Data, models and evaluation metrics (extended abstract). *J. Artif. Intell. Res.*, 47:853–899. 3.1

[Hossain et al., 2019] Hossain, M. Z., Sohel, F., Shiratuddin, M. F., and Laga, H. (2019). A comprehensive survey of deep learning for image captioning. *ACM Computing Surveys (CsUR)*, 51(6):1–36. 1.5, 2.3

[Hu et al., 2020] Hu, R., Singh, A., Darrell, T., and Rohrbach, M. (2020). Iterative answer prediction with pointer-augmented multimodal transformers for textvqa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9992–10002. 3.2.1, 4.3.1

[Huang et al., 2019] Huang, L., Wang, W., Chen, J., and Wei, X.-Y. (2019). Attention on attention for image captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4634–4643. 1.5

[Kumar and Goel, 2017] Kumar, A. and Goel, S. (2017). A survey of evolution of image captioning techniques. *Int. J. Hybrid Intell. Syst.*, 14:123–139. 1

[Lewis et al., 2019] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*. 4.3.2

[Li et al., 2017] Li, H., Wang, P., and Shen, C. (2017). Towards end-to-end text spotting with convolutional recurrent neural networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5248–5256. 3.1

[Lin, 2004] Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics. 2.3.3

[Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S. J., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *ECCV*. 2.3.3, 2.3.3, 3.1

[Liu et al., 2018] Liu, X., Liang, D., Yan, S., Chen, D., Qiao, Y., and Yan, J. (2018). Fots: Fast oriented text spotting with a unified network. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5676–5685. 3.1

[Liu et al., 2020] Liu, Y., Chen, H., Shen, C., He, T., Jin, L., and Wang, L. (2020). Abcnet: Real-time scene text spotting with adaptive bezier-curve network. In

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9809–9818. 4.2.1

[Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*. 2.1.1

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In *ICLR*. 4.2.2

[Mikolov et al., 2018] Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2018). Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*. 4.1.3, 4.2.2

[Mikolov et al., 2013b] Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751. 2.2.6

[Mishra et al., 2019] Mishra, A., Shekhar, S., Singh, A. K., and Chakraborty, A. (2019). Ocr-vqa: Visual question answering by reading text in images. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 947–952. 3.1

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill. 2.1, 2.1.3

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *ACL*. 2.3.3

[Parcalabescu et al., 2021] Parcalabescu, L., Trost, N., and Frank, A. (2021). What is multimodality? *CoRR*, abs/2103.06304. 2.2.5

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543. 2.2.6, 4.2.2, 4.2.2

[Radford et al., 2018] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. 2.1.1, 2.3.2

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9. 2.1.1

[Ren et al., 2015] Ren, S., He, K., Girshick, R. B., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149. 3.2.2, 4.1.3

[Russell and Norvig, 2002] Russell, S. and Norvig, P. (2002). Artificial intelligence: A modern approach. 1, 2.2

[Sanh et al., 2019] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*. 4.2.2, 4.2.2

[Sidorov et al., 2020] Sidorov, O., Hu, R., Rohrbach, M., and Singh, A. (2020). Textcaps: a dataset for image captioning with reading comprehension. In *European Conference on Computer Vision*, pages 742–758. Springer. (document), 1, 1.1, 1.1, 1.5, 2.3.3, 3, 3.1, 3.1, 3.1, 3.2, 3.2.1, 3.2.8, 4.1.2, 4.1, 4.3.1, 5.1.3

[Singh et al., 2019] Singh, A., Natarajan, V., Shah, M., Jiang, Y., Chen, X., Batra, D., Parikh, D., and Rohrbach, M. (2019). Towards vqa models that can read. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8309–8318. 3.1, 4.1.2

[Smith, 2007] Smith, R. (2007). An overview of the tesseract ocr engine. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2:629–633. 3.1

[Sudholt and Fink, 2016] Sudholt, S. and Fink, G. A. (2016). Phocnet: A deep convolutional neural network for word spotting in handwritten documents. In *2016*

15th International Conference on Frontiers in Handwriting Recognition (ICFHR), pages 277–282. IEEE. 4.1.3

[Tan et al., 2020] Tan, Z., Wang, S., Yang, Z., Chen, G., Huang, X., Sun, M., and Liu, Y. (2020). Neural machine translation: A review of methods, resources, and tools. *AI Open*, 1:5–21. 4.3.2

[Tiedemann, 2020] Tiedemann, J. (2020). The Tatoeba Translation Challenge – Realistic data sets for low resource and multilingual MT. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1174–1182, Online. Association for Computational Linguistics. 4.3.2

[Torfi et al., 2020] Torfi, A., Shirvani, R. A., Keneshloo, Y., Tavvaf, N., and Fox, E. (2020). Natural language processing advancements by deep learning: A survey. *ArXiv*, abs/2003.01200. 2.3.2

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008. 2.1.1, 4.1.3

[Wang et al., 2021a] Wang, C., Zhou, Z., and Xu, L. (2021a). An integrative review of image captioning research. In *Journal of Physics: Conference Series*, volume 1748, page 042060. IOP Publishing. 3.1

[Wang et al., 2020] Wang, J., Tang, J., and Luo, J. (2020). Multimodal attention with image text spatial relationship for ocr-based image captioning. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4337–4345. 3.2, 3.2.2

[Wang et al., 2021b] Wang, J., Tang, J., Yang, M., Bai, X., and Luo, J. (2021b). Improving ocr-based image captioning by incorporating geometrical relationship. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1306–1315. 3.2, 3.2.7

[Wang et al., 2021c] Wang, Z., Bao, R., Wu, Q., and Liu, S. (2021c). Confidence-aware non-repetitive multimodal transformers for textcaps. In *AAAI*. 3.2, 3.2.5, 4.2, 4.2.1

[Wolf et al., 2019] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2019). Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*. 4.2.2, 4.3.2

[Xu et al., 2021] Xu, G., Niu, S., Tan, M., Luo, Y., Du, Q., and Wu, Q. (2021). Towards accurate text-based image captioning with content diversity exploration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12637–12646. 3.2, 3.2.6

[Xu et al., 2015] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *ICML*. 3.1

[Yang et al., 2020] Yang, Z., Lu, Y., Wang, J., Yin, X., Florencio, D., Wang, L., Zhang, C., Zhang, L., and Luo, J. (2020). Tap: Text-aware pre-training for text-vqa and text-caption. *arXiv preprint arXiv:2012.04638*. 3.2, 3.2.3

[Young et al., 2014] Young, P., Lai, A., Hodosh, M., and Hockenmaier, J. (2014). From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78. 3.1

[Zhu et al., 2020] Zhu, Q., Gao, C., Wang, P., and Wu, Q. (2020). Simple is not easy: A simple strong baseline for textvqa and textcaps. *ArXiv*, abs/2012.05153. 3.2, 3.2.4

[Zhuang et al., 2020] Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76. 2.2.4