Universidad Zaragoza
1542

Trabajo Fin de Grado

DESARROLLO DE UNA APLICACIÓN PARA TRADUCCIÓN DE DOCUMENTOS OFICIALES

Autor: CAC. Fernando Rojas Albarrán

Director académico: Dra. Dª Lacramioara Dranca
Director militar: Cap. D. Carlos Rico Urones
Centro Universitario de la Defensa-Academia General Militar

2023

Agradecimientos

Cuando decidí promocionar a la Escala de Oficiales, hubo quienes que me animaron y se alegraron por ello, pero fueron pocos los que me advertían del camino que tenía que superar hasta alcanzar las dos estrellas que buscaba: *las de teniente de Infantería*. Con este proyecto subo el penúltimo escalón que me queda para que, tras siete años persiguiéndolas, alcanzar *las ansiadas estrellas*.

Estas estrellas, ni que decir tiene, que me las pongo, en primer lugar, gracias a mi madre, mi hermana y mi padre. Gracias de todo corazón a mi familia, especialmente a mi abuela Marisol, y mis amigos, *que tanto aman a España* y tanto me han ayudado.

Gracias a los mandos que han dejado parte de su vida en mí, predicando con el ejemplo y sacando lo mejor del cadete que suscribe: teniente coronel Calderón, comandante Espinosa, comandante Porgueres, capitán Cazenave y sargento Ruiz.

Gracias a mis hermanos de la LXXVI, LXXVII y LXXVIII promoción de la Academia General Militar por cargar la mochila conmigo por el solar zaragozano y pasar horas de camaradería juntos.

Gracias a quienes durante mi periodo de prácticas en el Regimiento de Infantería Canarias Nº50, aportaron a mi formación como teniente de infantería y como persona. Capitán Rico, tenientes del Batallón Ceriñola, así como el resto de mandos y personal de tropa con los que he tenido la suerte de coincidir: gracias.

Especial relevancia en el desarrollo del proyecto tiene la Dra. Lacramioara Dranca, profesora de quien suscribe durante el grado y tutora de este trabajo. Sin su ayuda, interés, constancia y compromiso, todavía seguiría intentando averiguar qué es un ordenador y cómo se enciende: gracias.

Gracias a la Academia de Infantería, por forjar su emblema en mi uniforme y enseñarme a ser un Oficial de Infantería española, la mejor y más antigua del mundo.

Por último, más que agradecimientos, quiero dedicar parte de todo a: Mónica, Juan, tanto padre como hijo y Elisa, familia del *Alférez de Infantería*, Pablo Jerez Sanjuan; por enseñarme lo que es ser valiente, tener fortaleza y ser leales y abnegados; valores que han de acompañarme durante el resto de mis días.

La última punta de mis estrellas es vuestra.

RESUMEN

En la actualidad el Ejército español participa en numerosas misiones en el extranjero y coopera en tiempos de Paz, desde territorio nacional (TN) como en zona de operaciones (ZO), con numerosos ejércitos, perteneciendo a la ONU y OTAN, siendo el inglés y el francés los idiomas oficiales de estas organizaciones. Se hace imprescindible una comunicación constante, predominando el flujo e intercambio de documentos entre los ejércitos (manuales, leyes, órdenes, etc.). Pese a que la enseñanza y aprendizaje de idiomas por parte de los miembros de las Fuerzas Armadas (FAS) ha de ser la prioridad en nuestras filas, las herramientas de traducción actuales realizan un trabajo ejemplar y contribuyen a esta comunicación, aunque se observe cierta deficiencia a la hora de traducir documentos oficiales dada la especificidad del vocabulario militar para algunos traductores.

Se pretende paliar este problema generando una fuente de datos de índole militar extraída de un diccionario militar denominado “MILLGLOSS”, en los idiomas de inglés, español y francés, para dotar a los traductores de un conjunto de datos de dominio específico militar, para que puedan ser entrenados en ello. Mediante programación con Python y el empleo de expresiones regulares, se obtienen estos datos y se definen en un estándar Ontolex para su futuro empleo por diferentes plataformas y/o aplicaciones de traducción, generando datos interoperables. El objetivo principal del proyecto consta del procesamiento del diccionario MILGLOSS para la extracción de estos datos de interés y su definición en estándar Ontolex para que pueda ser empleado en futuros proyectos. La elección de este documento para la extracción de datos viene motivada de que su elaboración ha sido por miembros de las FAS y especialistas en idiomas dentro del ámbito de defensa, por lo que es una fuente fiable y con buen contenido.

El procesamiento del pdf MILGLOSS, la depuración de los datos obtenidos para su correcto empleo y la definición a estándar Ontolex para dotar a estos datos de un carácter de interoperabilidad, serán las tareas que se abordarán durante el proyecto, mediante programación con Python y detección de patrones en los datos para su obtención.

Se obtiene como producto final un pdf procesado (generando un archivo txt), una matriz de datos depurados, cuatro archivos en los que aparecen definidas en estándar Ontolex las palabras en inglés, español, francés y la relación de inglés-español. Este producto responde a los objetivos marcados inicialmente en el proyecto.

Se deja como línea futura la ampliación de recopilación y definición de datos, en la misma línea que se ha seguido en este proyecto, así como su implementación en futuras aplicaciones de traducción de documentos de índole militar y su propuesta de modelo de aplicación.

Palabras clave

Aplicación, traductor, limpieza de datos, datos, código, dominio de adaptación, Ontolex y MILGLOSS.

ABSTRACT

At present, the Spanish Army participates in numerous missions abroad and cooperates, from national territory (TN) as well as in different areas of operations (ZO), with numerous armies belonging both to the UN and NATO, being English and French the official languages of these organizations. Constant communication is essential, with a predominant flow and exchange of documents between armies (manuals, laws, orders, etc.).

Although the teaching and learning of languages by members of the Armed Forces (FAS) must be a priority in our ranks, current translation tools do an exemplary job and contribute to this communication. However, there is a certain deficiency when translating official documents due to the specificity of military vocabulary for some translators.

The aim of this document is to alleviate this problem by generating a source of military data extracted from a military dictionary called "MILLGLOSS", in English, Spanish and French, to provide translators with a set of military-specific domain data, so that they can be trained in it. Through Python programming and the use of regular expressions, this data is obtained and defined in an Ontolex standard for future use by different platforms and/or translation applications, generating interoperable data. The main objective of the project is the processing of the MILGLOSS dictionary for the extraction of these data of interest and its definition in Ontolex standard so that it can be used in future projects. The choice of this document for data extraction is motivated by the fact that it has been elaborated by members of the FAS and language specialists in the defense field, so it is a reliable source with good content.

The processing of the MILGLOSS pdf, the debugging of the data obtained for its correct use and the definition of the Ontolex standard to provide these data with an interoperability character, will be the tasks that will be addressed during the project, through programming with Python and detection of patterns in the data to obtain them.

The final product is a processed pdf (generating a txt file), a matrix of cleaned data, four files in which the words in English, Spanish, French and the English-Spanish relation are defined in Ontolex standard. This product meets the initial objectives of the project.
It is left as a future line the extension of data collection and definition, in the same line that has been followed in this project, as well as its implementation in future applications of translation of military documents and its proposed application model.

KEYWORDS

Application, translator, data cleaning, linguistic data, domain adaptation, standard Ontolex & MILGLOSS.

INDICE DE CONTENIDO

INDICE DE FIGURAS

ABREVIATURAS, SIGLAS Y ACRÓNIMOS

LO, lengua origen
LM, lengua meta
AGM, academia general militar
APP, aplicación
DL, deep learning o aprendizaje profundo
EE.UU., Estados Unidos
FAS, fuerzas armadas
IA, inteligencia artificial
ML, machine learning
ONU, Organización de las Naciones Unidas
OTAN, organización del tratado del atlántico norte
RI, regimiento de infantería
RNN, redes neuronales recurrentes
RNA, redes neuronales artificiales
TA, traducción automática
UE, Unión Europea
ZO, zona de operaciones
TPU, Unidad de procesamiento de tensores
GPU, Unidad de procesamiento gráfica

1. INTRODUCCIÓN

¿Qué se entiende por traducir?

Traducir es el hecho de reproducir tanto la función, como el mensaje de un texto que es originario en un idioma y se quiere en otro distinto. No se debe perder ni la información que se pretende transmitir con el texto original, ni la función del mismo, así como tratar de mantener del original su forma y estructura (tipo de letra, organización del texto, estructura, etc.). (Hermida, 2019)

Desde que el ser humano empezó a trabajar con ordenadores y sacarles partido a estos sistemas, uno de los principales objetivos ha sido el de poder utilizarlos como traductores, por lo que este es el principal incipiente de la aparición de la traducción automática (TA). Para poder hablar de la TA hay que dejar claro dos conceptos: lengua origen (LO) y lengua meta (LM). A grandes rasgos la TA consiste en la capacidad de traducir un texto (aunque también voz) proveniente de una LO y en formato digital a otro texto en una LM; consiguiendo este hecho mediante el análisis del conjunto total de datos con los que cuente el sistema, analizando y escogiendo los datos que mejor se adapten en función del input de LO que se tenga para proporcionar el output más adecuado de la LM. Todo ello sin supervisión humana. (véase ilustración 1).

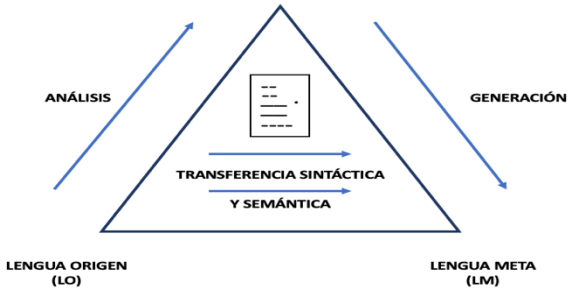

Ilustración 1- Esquema TA en forma gráfica. Fuente: elaboración propia a partir de "TFG Adan Soriano" (Soriano, 2019)

Gracias al Francés G. Artrouni y al ruso Trojanskij, podemos poner una fecha al inicio de la evolución de la traducción mecánica a la automática desde 1933 hasta día de hoy.

Fueron ellos los primeros impulsores de ciertas patentes para el uso de diccionarios mecánicos. En 1949, Warren Weaver empezó a dar importancia a la automatización de la traducción dado que observaba ciertas ambigüedades y problemas en la traducción mecánica, puesto que era un gran conocedor del ámbito informático, criptográfico y estadístico. Tras experimentos de investigación por el instituto de tecnología de Massachusetts en 1954, se obtuvo ciertos resultados significativos de un sistema de TA, suficientes para impulsar en EE.UU. la inversión en la investigación de la TA más adelante. (Peris, 2017)

La evolución de la TA hasta día de hoy ha ido aumentando de modo exponencial dados los avances de la tecnología y los conocimientos adquiridos en el campo de la informática y las matemáticas.

Hay diferentes tipos de TA: la basada en reglas y la basada en corpus. La TA basada en reglas tiene tres enfoques: transferencia (cuenta con tres etapas, análisis, transferencia y generación), directa (traducción inmediata y directa de LO a LM) e interlingua (análisis y generación, donde en una primera etapa el programa traduce el texto origen (o lo que es lo mismo, la LO) a un texto de carácter neutral; y en una segunda etapa se genera el texto meta (o LM). La basada en corpus puede ser basada en ejemplos (utilizando ejemplos y/o frases que con anterioridad han sido cargados en la base de datos con la que cuenta el sistema y un entrenamiento previo para su correcto empleo) o basada en estadística (lenguaje A y lenguaje B, mediante alineación de significados de las palabras de una frase se calculan probabilidades de ocurrencia hasta obtener la más acertada). (Briva-Iglesias, 2021)

A continuación, se muestra un esquema explicativo para su mejor entendimiento

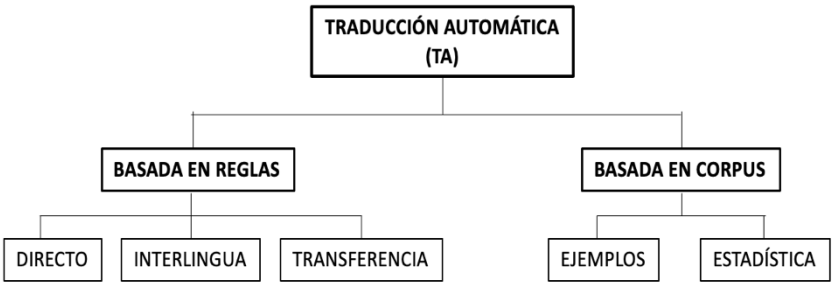

Ilustración 2- Esquema tipos de TA. Fuente: elaboración propia a partir de "TFG Adan Soriano" (2019) (Soriano, 2019)

Este último tipo de traducción, la basada en corpus, precisa contar con una gran cantidad de ejemplos para su correcto funcionamiento consiguiendo que, cuantos más ejemplos tenga el sistema, mejores resultados de traducción se obtienen de él. Para entender algo mejor este tipo de TA, se muestran las siguientes definiciones de TA basada en corpus:

Un tipo de modelo estadístico formado por un conjunto de unidades de proceso simple densamente conectadas entre sí. Los parámetros de estos modelos se estiman a través de corpus paralelos gracias a eficientes algoritmos de aprendizaje automático y a potentes procesadores gráficos" (Abril, 2017) y "Este nuevo método de traducción automática ha supuesto una revolución en cuanto a los motores de TA puesto que ofrece mejores resultados al imitar el comportamiento del cerebro humano, analizando la frase como un conjunto, las sutilezas del lenguaje y las variaciones que puedan existir (Sorolla, 2018)

¿Imitar el comportamiento del cerebro humano? Sí, mediante redes neuronales artificiales (RNA). Entrar en detalle para explicar qué es y cómo funcionan las redes neuronales sería ampuloso para este proyecto. A modo resumen, las RNA tratan de imitar el comportamiento del cerebro humano mediante asociaciones de una palabra con otros componentes, datos e información que permiten a esta red, asociar una palabra con varias características de la misma (esquema explicativo en la ilustración 3). Este hecho trata de imitar las conexiones interneuronales que los humanos tienen en su cerebro y que les permiten acceder a la información y catalogarla.

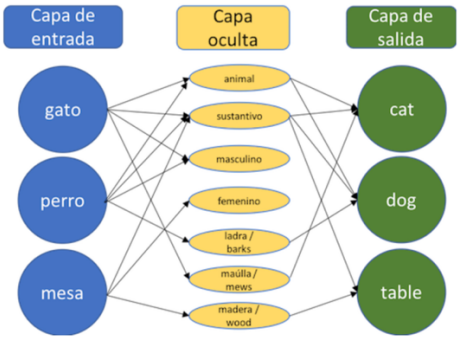

Ilustración 3- Esquema RNA para diferentes palabras (entrada, capa oculta, salida). Fuente: La linterna del traductor

Hoy día, traductores como Google Translate, DeepL. 2.2.1., Diccionario Collins, Babylon o Bing Translator, son los más utilizados y se encuentran de forma gratuita y online en internet, realizando un excelente trabajo de traducción muy completo dado que basan su funcionamiento en la TA basada en corpus, contando con grandes cantidades de datos de entrenamiento que les proporciona un resultado más preciso y natural que los resultados que se obtendrían con la TA basada en reglas.

Debido a la globalización de estas últimas décadas, se hace necesario, no sólo en el ámbito civil, sino también en ámbito militar, que entre organismos que desempeñan sus funciones de forma conjunta hablen el mismo idioma. Es necesario que entre estos pueda haber un intercambio fluido de información y documentos, los cuales, si son traducidos de un idioma a otro, no deben perder su función informativa ni el mensaje a transmitir. Desplazándonos al ámbito militar, el ejército español se encuentra desplegado en diferentes tipos de misiones internacionales, perteneciendo a la Organización de Naciones Unidas (ONU), Unión Europea (UE) y Organización del Tratado del Atlántico Norte (OTAN) de forma permanente para la defensa del territorio nacional y su soberanía, así como el cumplimiento de los diferentes tratados con el resto de países para la defensa conjunta; ya sea para llevar a cabo misiones de estabilización, mantenimiento de la paz, cooperación multinacional en tiempos de paz o acciones de intervención limitada.

Este hecho trae consigo la necesidad de una comunicación fluida tanto en territorio nacional (TN) como en zona de operaciones (ZO); ya demostró su necesidad el ejército de los Estados Unidos (EE.UU.) en Irak y Afganistán a principios de siglo, marcando como objetivo clave para el desarrollo de las operaciones *"ganar el corazón y las mentes"*. Para ello, se hacía imprescindible una comunicación fluida a tiempo real y dada la escasez de intérpretes locales en la zona y la dificultad añadida de conseguir este personal, se materializó un sistema de traducción automática en tiempo real, tanto de texto como de voz conocido como el Machine Foreign Language Translation System (MFLTS), poniéndose en uso en 2016. (MADOC, 2017)

Coincidiendo en fecha, la sección de idiomas del Mando de Adiestramiento y Doctrina (MADOC) junto con estudiantes de la universidad de Granada, crearon la aplicación denominada "*Idiomas Operativos"* o *"Idiomas OP"*. Disponible para Android sin requerir el uso de red para su funcionamiento. Esta aplicación, puede traducir a idiomas como alemán, árabe, libanés, marroquí, inglés y francés, tanto mediante el uso de expresiones escritas como por voz. La aplicación cuenta con unas 1.200 palabras en su base de datos. (Defensa, 2016)

Puede surgir la pregunta de por qué se necesitan estas herramientas si se cuenta en todas las misiones con intérpretes locales de la zona, para el apoyo a las operaciones, y traductores e intérpretes de los jefes de las unidades que realizan labores de contacto estrecho con la población o el personal destinado en TN, que cuenta con acceso a diferentes sistemas de traducción.

Aparte de la *formación en idiomas del personal de las fuerzas armadas (FAS), que debe ser el pilar fundamental para contribuir a la buena comunicación con los demás ejércitos y que, ha de ser tratado en las pequeñas unidades como prioridad junto con la instrucción diaria*, una aplicación que permita traducir documentos para complementar la comunicación entre nuestras FAS y el resto de ejércitos de los países aliados, puede ser de utilidad.

La comunicación entre ejércitos es necesaria y de modo continuado ya sea en TN o ZO.

Son muchas las dependencias de nuestro ejército (y de los aliados también) que trabajan día a día en contacto estrecho con otros de la fuerza aliada, intercambiando informes, análisis, manuales y demás. Por tanto, se hace preciso el contar con una herramienta que nos permita traducir de la forma más precisa documentos oficiales provenientes de otros ejércitos para una comunicación más precisa. Resaltar que se entiende por documentos oficiales aquellos documentos que no se limitan tan solo a comunicarse entre entidades como podría ser una carta, sino que, se entiende como tal, aquellos documentos que se generan como producto de estudios, creación de manuales o procedimientos, órdenes de operaciones para las misiones, reglamentos de diferentes índoles, etc. En la OTAN se cuenta con un Standarization Agreement (STANAG),

en español, “Acuerdo de Normalización”, cuyo fin último es contribuir a la interoperabilidad de los ejércitos. Aparte de normalizar procedimientos, requisitos y condiciones de equipamiento, cuenta con la estandarización de publicaciones, siendo el inglés y el francés los idiomas de estas (dado que son los idiomas oficiales de la OTAN). Estos documentos, conocidos como documentos STANAG, se actualizan de forma periódica para garantizar la interoperabilidad, la eficacia y la eficiencia de los ejércitos pertenecientes a la OTAN. (SITE, s.f.)

1.1 Motivación del trabajo

Como se ha mencionado anteriormente, es cierto que ya existen traductores que realizan una actividad ejemplar de traducción, dado que, en los últimos años, la demanda de estos servicios y herramientas ha crecido exponencialmente, motivado por la globalización y el uso extendido de internet.

Es cierto que los traductores ya mencionados, y otros, realizan un ejemplar trabajo de traducción y son muy efectivos en su producto, por lo que puede surgir la pregunta de: ¿por qué este proyecto si ya tenemos herramientas para la traducción? La respuesta puede ser sencilla: debido la especificidad de dominio que conlleva el vocabulario militar.

El desarrollo de técnicas de aprendizaje profundo ha permitido que los modelos de traducción automática neuronal (NMT) se vuelvan extremadamente poderosos, dados los datos de entrenamiento y el tiempo de entrenamiento suficientes. Sin embargo, los sistemas tienen dificultades al traducir texto de un nuevo dominio con un estilo o vocabulario distinto. (Saunders, 2022)

Dada la particularidad del vocabulario militar, como cualquier otro de índole específico (científico, de historia, matemático, etc.), trae a las herramientas de traducción problemas que ponen a prueba su eficacia, es decir, cuanto más específico es un dominio en un texto, más deteriorado se ve el producto obtenido en su traducción, es decir, el texto meta.

La motivación del trabajo radica en la intención de querer paliar el deterioro en la traducción de documentos del ámbito de defensa, dada la importancia que esto puede tener en el desarrollo de misiones o la cooperación con otros ejércitos por el hecho de la especificidad del vocabulario militar.

La Academia General Militar (AGM) cuenta con un diccionario “MILGLOSS” que contiene terminología militar en diferentes idiomas: español, inglés y francés. Pero, ¿cómo este diccionario puede contribuir a un traductor?

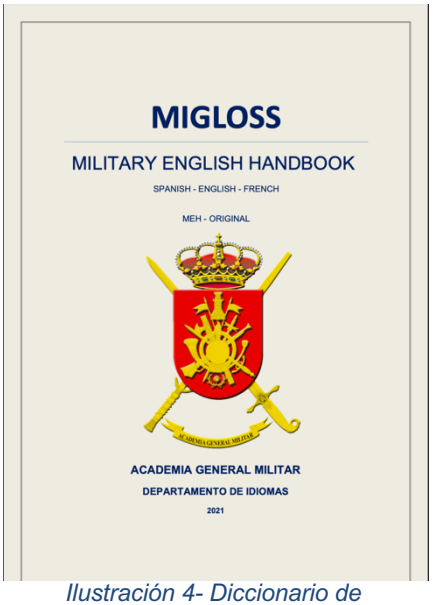

Ilustración 4- Diccionario de terminología militar MILGLOSS, de la AGM. Fuente: AGM

Recordando que la TA basada en corpus se nutría de grandes cantidades de datos que le permitían usarlos de ejemplos para entrenar las redes neuronales, este diccionario podría ser útil para aumentar en cantidad y calidad de estos datos. Sabiendo de primera mano que los datos y la información aportada son veraces y adecuados dado que han sido obtenidos y elaborados por profesionales dentro del ámbito de defensa, tanto en el propio campo de la defensa como en el de los idiomas (por el hecho de ser personal militar dedicados al área de idiomas, mencionados en el apartado de agradecimientos de este trabajo). Pese a que este documento podría ser de gran ayuda por lo anteriormente mencionado (contenido, procedencia, etc.) cabe destacar el problema que se tiene en cuanto al formato del mismo. Dado que es un formato pdf no estructurado, se precisa “informatizar” el mismo, es decir, convertirlo a un formato machine-readable para extraer, organizar y definir su

contenido de tal forma que sea compatible con un sistema de TA y este pueda utilizarlo.

Un formato machine-readable es un formato de datos que una computadora puede leer y procesar fácilmente de forma automática, sin intervención humana. Estos datos han de ser estructurados. (Ávila, s.f.)

1.2 Ámbito de aplicación

Tras un breve periodo en el RI Canarias Nº50 de quien suscribe, estando destinado con la compañía de Mando y Apoyo del Batallón Ceriñola y una entrevista con el coronel de artillería D. Antonio Martínez de Baños Carrillo, destinado en la AGM y militar especializado en idiomas e historia, se llega a la conclusión que el ámbito de aplicación abarca a todo el personal de las FAS que precisen, por los cometidos de su puesto, de una herramienta que le permita traducir documentos oficiales.

El RI Canarias Nº50 ha sido una buena muestra para la obtención de información por diversos motivos:

- Cuentan con personal de antigüedad y experiencia dentro de las FAS
- La unidad es una de las unidades del E.T. que más veces ha participado en misiones en el extranjero, siendo las más participadas Afganistán, Líbano, Mali y Bosnia y Herzegovina.
- Dada la antigüedad del personal destinado, no sólo se obtiene información sobre el RI Canarias Nº50, sino que, es personal que en el inicio de su carrera militar fueron destinados en otras unidades del territorio nacional, aumentando los conocimientos y experiencia de la unidad.

1.3 Estructura de la memoria

La estructura de la memoria se dividirá en diferentes capítulos:

- CAPÍTULO 1: INTRODUCCIÓN. En este apartado se introducirá al lector en el tema a tratar durante el trabajo para hacerle saber la necesidad del mismo y la intención que se tiene para abordar la posible solución.
- CAPÍTULO 2: OBJETIVOS Y METODOLOGÍA. En este apartado se detallan los objetivos que se deben alcanzar para poder llevar a cabo el trabajo, detallando en cada uno su utilidad y aportación al mismo. También se detallará el alcance del trabajo, así como los métodos empleados durante su realización.
- CAPÍTULO 3: MARCO TEÓRICO. En este apartado se tratará de recopilar información suficiente y apropiada para poder dotar tanto al lector, como al propio autor del trabajo, de información que permita investigar, discernir y dar solución al motivo del trabajo.
- CAPÍTULO 4: DESARROLLO. En este apartado se expondrán los productos a aportar por el autor y que son los que darán apoyo y soporte a futuras implementaciones de una aplicación para traducir documentos militares, así como la propuesta del modelo de esta.
- CAPÍTULO 5: CONCLUSIONES, LIMITACIONES Y LÍNEAS FUTURAS. Finalizado el trabajo, se expondrán las conclusiones relevantes obtenidas durante la realización del mismo.

2 OBJETIVOS Y METODOLOGÍA

2.1 OBJETIVOS Y ALCANCE

Como objetivo principal del trabajo se tiene la extracción, organización y definición de datos provenientes del MILGLOSS para su futuro empleo en un traductor. Este objetivo nace del propio objetivo de contribuir al perfeccionamiento de las herramientas de traducción automática existentes, adaptándolos con terminología militar contribuyendo así, a la mejora de la comunicación de nuestras FAS en el desempeño de sus funciones.

Para poder materializar el objetivo principal del proyecto se precisa alcanzar una serie de objetivos intermedios e inter relacionados que permitan, en conjunto, obtener algún producto tangible e innovador del proyecto.

Tras una revisión bibliográfica de los diferentes métodos de traducción de documentos presentes en la actualidad se definirán los requisitos de calidad con los que este proyecto deberá contar para su correcto funcionamiento. Otro de los objetivos intermedios es poder procesar el documento pdf MILGLOSS en un formato estándar, extrayendo, transformando y codificando el mismo.

Ya extraído el conjunto de datos que interesen del MILGLOSS, nace la necesidad de definirlos de modo que cualquier sistema (o la mayoría de ellos) puedan trabajar con estos datos, para su futuro empleo en traductores. Surge por tanto el objetivo de definir estos datos en un formato estándar.

Para cumplir estos objetivos deben llevarse a cabo las siguientes tareas:

- Tarea 1: revisión bibliográfica y documentación.
- Tarea 2: definición de requisitos
- Tarea 3: propuesta de una solución al modelo de traducción
- Tarea 4: elaboración de un código para la extracción de los datos
- Tarea 5: elaboración de un código para la definición de los datos extraídos a un formato estándar.

2.2 METODOLOGÍA

En un primer momento, la metodología seguida se basará en la recopilación y revisión de información bibliográfica. Se ha procedido a la adquisición de esta mediante la búsqueda, lectura y análisis de información a través de Google académico, documentos de estudios de carácter científico, trabajos de fin de grado que pudiesen aportar algo de información y tanto vídeos como documentales acerca de la materia para su mejor comprensión. También en esta etapa temprana se proponen los aspectos a tratar en una futura encuesta dirigida a una muestra de personal destinado en el RI Canarias N.º 50 así como una entrevista con el coronel Martínez de Baños, destinado en la AGM encargado del departamento de idiomas, para conocer de primera mano las necesidades, motivación y fin que se pretende alcanzar con este proyecto.

Materializando el trabajo, en una primera etapa, para la extracción de datos del diccionario MILGLOSS en la cual, se emplearán métodos científico-informáticos para la elaboración del código fuente que nos permita extraer la información necesaria de documentos pdf mediante el lenguaje de programación Python en un entorno de trabajo de Colab. Dentro de esta etapa coge especial importancia de la implementación de un código mediante expresiones regulares. Estas expresiones regulares son una herramienta para buscar y manipular patrones de texto. Cabe destacar que para comprender como se han empleado estas expresiones regulares y su forma de aplicarlas, se empleará la página web regexper, que mostrará de forma gráfica el uso de las mismas.

Para la transformación del contenido de MILGLOSS a un formato estándar se utilizará ontolex (lenguaje de modelado para representar información lingüística para su integración e intercambio entre diferentes sistemas y/o aplicaciones, facilitando su procesamiento y análisis).

La metodología irá en la misma línea que la anterior, mediante la implementación de un código en Python que permita definir los datos extraídos de MILGLOSS en este entorno para facilitar su acceso y futuro empleo en cualquier sistema.

A continuación, se definen las herramientas utilizadas que ya han sido mencionadas:

1. Encuesta y entrevista para detectar necesidades, requisitos y obtener datos. Se ha procedido a realizar un cuestionario compuesto de seis preguntas cerradas y una abierta. Esta encuesta la han realizado militares destinados en el RI Canarias Nº50, con un total de 37 respuestas. La encuesta se ha elaborado a través de Google drive, generando un enlace desde el cual podían acceder a responderla sin necesidad de identificarse. La entrevista personal se realizó al coronel Martínez de Baños en su despacho, siendo esta un compendio de 4 preguntas consideradas clave para comprender y conocer de primera mano las necesidades de las FAS y del personal en esta materia.

2. Python es un lenguaje de programación altamente utilizado para la creación de páginas web y el machine Learning, explicado más adelante. Los códigos con lo que Python trabaja pueden encontrarse y extraerse de diferentes fuentes para ser adaptados a las necesidades particulares de cada sistema.

3. Colab es un producto de Google.research el cual nos permite diseñar, escribir y ejecutar código en lenguaje Python de forma gratuita en un entorno de trabajo en red. Es un entorno de trabajo especialmente recomendado para el análisis de datos. Una de las limitaciones es que no permite alojar archivos, pero se solventa mediante un código que permite trabajar con archivos en línea, en el caso de este proyecto, un archivo subido a Google drive. Otra de las limitaciones es que en la versión gratuita el tiempo de uso y la memoria son limitadas, aunque suficientes para llevar a cabo la elaboración del proyecto. (Anon., 2019)

4. Regexper es una página web que nos permite visualizar de forma gráfica las expresiones regulares que se definen durante la elaboración del código en Python, permitiendo decodificarlas y mostrando los grupos generados según las diferentes reglas que se han implementado para poder extraer la información requerida del .pdf en forma y cantidad. (Quijano, 2013)

3 MARCO TEÓRICO

En este apartado se tratarán conceptos que se consideran clave para la comprensión del proyecto, así como la intención de informar al lector y transmitir los conocimientos de ciertos campos referentes a la traducción.

3.1 Traducción automática neuronal

Ya se han citado los tipos de traducción automática y algo de su funcionamiento, pero, no se ha mencionado hasta ahora la TA neuronal, ¿qué la hace especial? O, ¿qué la diferencia del resto?

La traducción automática ha evolucionado significativamente en las últimas décadas gracias a los avances en la inteligencia artificial y el aprendizaje automático. En particular, la TA neuronal ha surgido como una técnica prometedora para mejorar la precisión y la fluidez de las traducciones. Traductores como DeepL o Google Translate emplean este tipo de traducción, siendo actualmente los traductores más utilizados y qué mejor resultado ofrecen en sus traducciones.

Pues sin dar muchas vueltas, la TA neuronal no discierne en gran medida de la TA basada en corpus (estadística y ejemplos) vista hasta ahora; podría decirse que este nuevo tipo de traducción es una especie avanzada de traducción de la basada en corpus, representando cada palabra como vectores en vez que, de forma discreta, lo que la diferencia del resto.

Pero, ¿cómo funciona en sí la traducción automática neuronal? Las redes neuronales que la componen tratan de asignar un vector a cada palabra, realizando productos escalares y asignando vectores de pesos asociados a estas y a la relación que tienen entre ellas (midiendo lo que las palabras están relacionadas entre sí); por tanto, se tiene que, la palabra n, depende de las palabras n-i en algún peso concreto y de otros parámetros del propio sistema de traducción.

3.1.1 Redes neuronales e Inteligencia artificial

Es evidente que para entender mejor la TA neuronal se precisa definir mejor qué es una red neuronal. Ya se ha mencionado en el apartado de introducción brevemente qué es y cómo trabajan estas redes neuronales. Una red neuronal artificial busca imitar tanto la forma como el comportamiento del cerebro humano, basando su arquitectura y conexiones de forma similar a las neuronas presentes en el cerebro humano, ahora bien, ¿cómo funcionan las neuronas? su función es transmitir y recibir información mediante impulsos eléctricos entre ellas, estando interconectadas formando redes neuronales; por lo que una neurona por sí sola no tendría ninguna función.

Las redes neuronales artificiales tratan de imitar este comportamiento, siendo la estructura más básica de las redes neuronales artificiales la formada por tres capas: capa de entrada, oculta y de salida. Las redes neuronales más complejas poseen más de una capa oculta, pasándose a denominar redes profundas. Por lo tanto, se tiene que el conjunto total de estas capas funciona del mismo modo que el cerebro, procesando y transmitiendo la información entre las diferentes capas, iniciándose este flujo en la capa de entrada y finalizando en la de salida, todo ello mediante algoritmos y ecuaciones matemáticas que asignan diferentes pesos a las entradas de cada palabra y demás para lograr que la red neuronal pueda asignar de algún modo un peso a una palabra concreta.

Para el proceso de traducción de una frase en este tipo de traductores, la red neuronal trabaja y se basa sobre traducciones preexistentes en el sistema y las posibles combinaciones que el input le ofrece mediante métodos estadísticos que asignan a una entrada una probabilidad de ocurrencia dentro de una frase (mediante algoritmos y matemáticas, como se ha dicho).

La inteligencia artificial (IA) se ha convertido en una herramienta fundamental para los sistemas de traducción, en particular, las técnicas de aprendizaje automático y procesamiento del lenguaje natural que son utilizadas para mejorar la calidad y la velocidad de los traductores.

Alan Mathison Turing, británico matemático, fue propulsor de la IA desarrollando una máquina electromecánica que fue capaz de descifrar los códigos nazis durante la II Guerra Mundial, mediante un algoritmo informático que permitía al sistema interpretar y aprender por sí sólo los datos que obtenía. Y es que es este el motivo que hace de este tipo de traducción la más utilizada y eficiente a día de hoy, sabiendo que se basan en IA.

A grandes rasgos, los traductores automáticos que basan su funcionamiento en IA funcionan identificando patrones en grandes conjuntos de datos de texto multilingüe. A partir de estos datos y patrones, el sistema de traducción puede aprender a traducir.
Los sistemas de traducción automática más avanzados utilizan redes neuronales profundas y modelos de lenguaje basados en Transformers para generar traducciones más precisas y naturales. Estos modelos son capaces de aprender la estructura y el significado de los textos en diferentes idiomas y, por lo tanto, son capaces de producir traducciones más precisas y coherentes (Escrivá, 2021)

La IA no es una habilidad en sí misma, no es hacer, es con qué eficiencia puedes aprender cosas nuevas (Anon., 2021)

Sin embargo, es importante tener en cuenta que la calidad de las traducciones generadas por los sistemas de traducción automática todavía no es perfecta y puede variar ampliamente dependiendo del idioma y del contexto. Por lo tanto, los traductores automáticos basados en inteligencia artificial todavía requieren la supervisión y el trabajo de la supervisión humana para garantizar la calidad y la precisión de sus traducciones. (F., 2022)

3.1.2 Machine Learning, aprendizaje profundo y modelo encoder-decoder.

Un subconjunto de la IA es el machine Learning (ML) o *aprendizaje automático*. Este ayuda a mejorar la calidad y velocidad de las traducciones automáticas, haciendo que el sistema aprenda mediante ejemplos previos de traducciones y que él mismo obtenga sus patrones para futuras traducciones. Este aprendizaje automático tiene tres campos o formas: supervisado, no supervisado y de refuerzo, siendo el más empleado para la traducción el primero de ellos; en los que como su propio nombre indica, modelan la forma de intervención de la IA en el proceso de aprendizaje, limitando la intervención y control humana en función de la necesidad. (Hernández-Sosa, 2022)

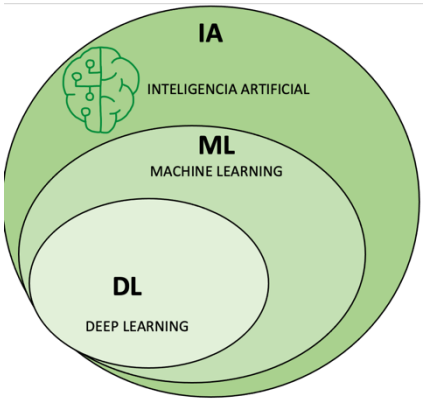

Ilustración 5- Representación de conjuntos/subconjuntos IA-ML-DL. Fuente: elaboración propia

Para generar una traducción automática, el modelo toma como entrada una oración en el idioma de origen y, utilizando los patrones y relaciones que ha aprendido durante el entrenamiento previo, genera una traducción en el idioma de destino. Es evidente que cuanto más se entrene el modelo y mayor cantidad de datos, mejor será su capacidad para realizar traducciones precisas.

Un subconjunto del ML y forma más avanzada de este es el aprendizaje profundo o *Deep Learning* (DL).

DL se basa en el intento de dotar de comportamiento humano a las máquinas para el autoaprendizaje. Esto dota al sistema de TA neuronal de capacidad de aprendizaje adquirido mediante el entrenamiento por medio de segmentos de la lengua de origen que, emparejados con su traducción forman una enorme base de datos que nutren a la red neuronal durante su entrenamiento. “Son memorias de traducción enormes que contienen miles, incluso millones, de unidades de traducción” (Peris, 2017)

Este nuevo método de traducción automática ha supuesto una revolución en cuanto a los motores de TA puesto que ofrece mejores resultados al imitar el comportamiento del cerebro humano, analizando la frase como un conjunto, las sutilezas del lenguaje y las variaciones que puedan existir (Sorolla, 2018)

DL utiliza para la traducción automática redes neuronales artificiales con múltiples capas ocultas. Estas capas ocultas permiten a la red aprender de manera jerárquica, desde características simples a complejas, lo que le permite identificar patrones y relaciones en los datos de entrada.

El proceso de entrenamiento de estos modelos implica dotar al sistema de grandes cantidades de datos de entrada y salida en dos idiomas diferentes. La red neuronal aprende a mapear las entradas en el idioma de origen a las salidas en el idioma de destino. A medida que se presenten más datos durante el entrenamiento, la red se ajustará gradualmente a la tarea de traducción, siendo directamente proporcional la precisión de traducción con la cantidad de datos de entrenamiento que cuente el sistema (cuantos más datos, mejor traduce). Esta acción se asemejaría a la acción de un alumno que, tras revisar un examen y ver sus fallos, se presenta a hacerlo de nuevo y vuelve a revisarlo, observar sus fallos, volver a presentarse, etc.

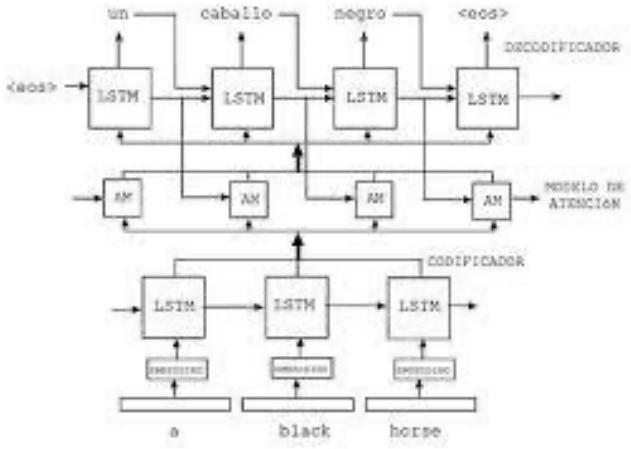

Ilustración 6- Esquema del proceso de traducción de una frase. Fuente: Traducción Automática Neuronal, Álvaro Peris Abril (Peris, 2017)

Es durante la fase de inferencia, cuando se realiza la traducción en sí misma. La red neuronal toma como entrada una oración en el idioma de origen y genera una oración en el idioma de destino. Este proceso implica pasar la entrada a través de la red neuronal, que genera una distribución de probabilidad sobre todas las posibles palabras en el idioma de destino. La red selecciona entonces la palabra más probable y la agrega a la oración de destino en proceso de construcción, y se repite el proceso hasta que se completa la oración de destino.

Se han mencionado anteriormente los conceptos de LO y LM. Suele ser habitual que el texto a traducir en la LO tenga una longitud diferente que el que se obtiene en LM, pese a que la traducción sea la correcta, es decir, que, aunque no todas las palabras en la LO tengan una homónima en la LM, el significado sea el correcto. Para dar solución a ello, se utiliza el modelo encoder-decoder, o lo que es lo mismo, codificador-decodificador. Este, en una primera etapa mapea la oración de LO con su vector de longitud fija y en una segunda etapa, este vector codificado en la primera, se decodifica para poder dar un input en LM de diferente tamaño (o no). (Bahdanau, 2014)

En la ilustración 6 puede observarse que “un caballo negro” y “a black horse” cuentan con el mismo número de entradas (x) que de salidas (y). Se muestra en la ilustración 7 un breve esquema de cómo funciona este en el caso de obtener, como se ha mencionado, outputs de diferente tamaño que los inputs.

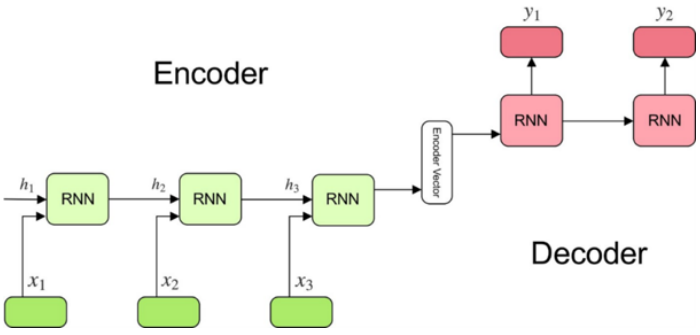

Ilustración 7- Esquema encoder-decoder para longitudes diferentes. Fuente: Predicción multi-horizontes de series temporales con redes neuronales, Guillermo Delgado Martínez. (Briva-Iglesias, 2021)

3.2 Transformers (o transformador)

Es hora de mencionar el término Transformers o Transformador. Dada la complejidad del concepto y función de los Transformers, pongamos primeramente un ejemplo para materializar qué es y para qué sirven con dos sencillas frases:

La infantería española es la <u>más</u> antigua del mundo

La infantería española es la <u>más</u> admirable del mundo

Como se puede observar, las frases son prácticamente idénticas. Sólo una inteligencia humana podría discernir en la diferencia del "más" dado que, en la primera oración se refiere a antigua y en la segunda a admirable. Lo que un Transformers trata hacer es eso mismo, discernir e identificar esta diferencia de significado, siendo capaz de identificar que el "más" hace referencia en cada frase a una palabra diferente. Pero, ¿por qué aparecen los Transformers? Simplemente por el propio proceso de evolución y mejora.

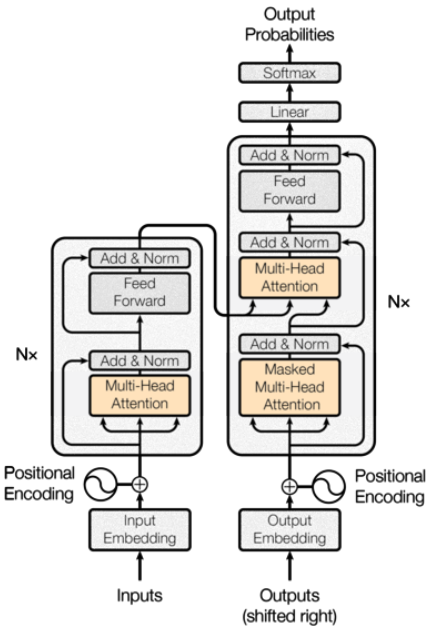

Ilustración 8- Transformes en TA neuronal. Fuente: aprendemachinelearning.com (Bagnato, 2019)

Previo a los Transformers, podíamos encontrar las redes neuronales recurrentes (RNN). Para entender la evolución y mejora, se procede a explicar brevemente en qué consisten estas RNN,s: para una frase concreta de entrada (input), la RNN procesa en orden desde la primera palabra hasta la última (*many-to-many*), según los parámetros con los que se haya ajustado esta, hasta obtener un output.

Teniendo en cuenta que para la primera palabra obtiene un output que, a su vez, servirá de input junto con la siguiente palabra hasta completar la frase al completo. Con lo cual, el output final obtenido es un procesamiento desde la primera hasta la última palabra, en orden y relacionadas. A continuación, se muestra una imagen explicativa, donde las "x" son los inputs de entrada (palabras) y las "y" los outputs de salida.

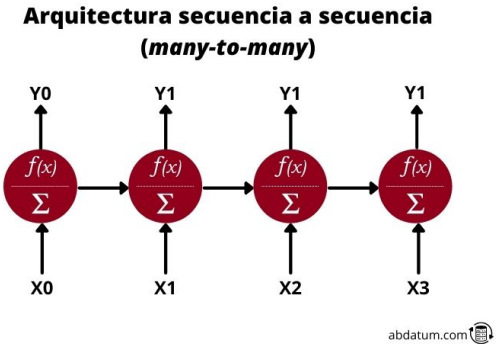

Ilustración 9- Funcionamiento de las RNN,s. Fuente: Rubén Cañadas (cañadas, 2022)

Estos modelos presentan un problema cuando los datos de entradas son de un tamaño considerable dado que, como se puede suponer, los datos que el proceso ha de manejar durante la traducción del mismo serían enormes, por el hecho de tener que obtener un output para cada palabra de entrada a traducir proveniente de la palabra anterior, generando problemas de comprensión si una palabra del inicio de la frase, hace referencia por ejemplo, a una del final, ya que durante el proceso, el sistema no sería capaz de relacionar estas palabras y obtendría un resultado erróneo. Es por este motivo que surgen los Transformers que, en resumen, son una

arquitectura de red neuronal muy efectiva para la traducción automática y otras tareas de procesamiento de lenguaje natural, debido a su capacidad para procesar secuencias de texto de longitud variable de manera eficiente y capturar relaciones a largo plazo entre las palabras en una oración. (Edunov, 2018)

3.3 Adaptación de dominio o domain adaption

Los traductores basados en TA neuronal funcionan de forma muy eficiente en los dominios en los que han sido entrenados, pero, ¿para dominios en los que no? Podría llevar a error el hecho de pensar en entrenar a estas redes neuronales en todos los dominios que interese, lo que sería un caos dada la gran cantidad de datos que se manejan, el tiempo empleado y la rentabilidad del proceso. Si bien es posible entrenar un nuevo modelo en todos los conjuntos de datos desde cero para cada nuevo dominio de interés, generalmente no es práctico, ya que los conjuntos de entrenamiento típicos se ejecutan en millones de pares de oraciones y requieren varios días para el entrenamiento (Thompson, 2019)

Una solución a este problema es la reutilización del modelo para, una vez entrenado, volver a hacerlo para introducirle este lenguaje más específico, conociéndose este hecho como ajuste fino.

Para abordar el concepto del problema de la adaptación de dominio primero se tratará de definir qué es un dominio por medio de los tres elementos principales que lo componen: procedencia, tema y género (Van der Wees, 2017)

Se conoce por procedencia a la propia fuente del texto. La procedencia se caracteriza por poder ser estrecha o amplia.

El tema es el propio asunto del texto, muy ligado al tipo de vocabulario empleado en él, diferenciando entre tema explícito e implícito.

El género se entiende por un concepto que engloba el conjunto del tipo de sintaxis que forma el texto, su estilo, el registro o el propio tema. Textos de un mismo género tienen señales y patrones que lo identifican por su similitud.

Para lograr la adaptación de dominio se procede con el llamado ajuste fino; este ajuste fino se basa, a grandes rasgos, en el propio hecho de volver a entrenar un modelo que ya ha sido entrenado previamente, añadiendo los nuevos datos del dominio específico que queremos utilizar. (Saunders, 2022)

Como se ha comentado, dada la importancia de los datos de entrenamiento en la construcción de un modelo de traducción, vamos a hablar a continuación del diccionario MILGLOSS, del cual será el punto de partida del que se extraerán los datos de interés.

3.4 Diccionario MILGLOSS

Ya se ha comentado que el diccionario MILGLOSS, empleado en la AGM por sus alumnos, es un diccionario específico de terminología militar. Su primera edición data en el año 2002 y ha sufrido numerosas actualizaciones hasta la última, datada en el año 2021.

Sus autores son personal perteneciente al Centro Universitario de la Defensa (CUD), personal de las FAS destinados en la Academia General Militar (AGM) y/o en el CUD, así como personal civil que ha colaborado en él. Dada la naturaleza de los autores, siendo esta una mezcla entre personal de las FAS y profesionales de los diferentes idiomas que forman el diccionario, se hace inminente la fiabilidad y calidad de la información que hay en él. Como se ha mencionado anteriormente la peculiaridad del vocabulario militar es muy singular en cada país y sólo expertos en las FAS podrían aportar información veraz y precisa sobre las diferentes terminologías que el diccionario empleará. Todo ello no cobra sentido si no tiene la sinergia de la lingüística, es decir,

si no se está rodeado de profesionales que sepan plasmar las ideas y el conocimiento sobre el papel.

Sus autores:

DRA. CRISTINA SÁIZ ENFEDAQUE
COL ART. ANTONIO MARTÍNEZ DE BAÑOS CARRILLO
LDA. ARACELI PONTAQUE GRACIA
LDA. INMACULADA SERICHOL BLASCO
TCOL ART.JOSÉ LUIS GARCÍA ALQUÉZAR
TCOL INF. JOSÉ LUIS GRANDE RUIZ DE LA TORRE
LDA. M a del CAMEN DUBOIS AZNAL

Editora:

LADISLAÁ SÁNCHEZ GASCÓN

El índice del mismo es: prólogo, agradecimientos, explicación de simbología, abreviaturas, etc., diccionario militar y conjuntos de terminologías específicas de ciertas áreas. Para el proyecto nos centraremos en el apartado de diccionario (el más extendido) dado que en él obtenemos palabras en inglés, francés y español, así como algunos ejemplos de frases de español e inglés homólogas entre sí.

3.4.1 SÍMBOLOGÍA QUE APARECEN EN MILGLOSS:

Con la aplicación de “búsqueda” del pdf se pueden localizar fácilmente palabras. La información sobre la que vamos a extraer los datos de interés viene dada por:

- Las entradas en inglés.
- Tipo de palabra que es (vt, adj, adv, n, v y vi).
- Su definición en inglés.
- Traducción al español.
- Traducción al francés.
- Así mismo hay entradas que cuentan con siglas y/o acrónimos.
- Algunas de las entradas cuentan con frases de ejemplo de uso de esa misma entrada, tanto en inglés como su traducción al español.

Destacar que en él aparece simbología referente a banderas de España, Gran Bretaña, Estados Unidos, OTAN y Francia, que harán referencia a los idiomas de los distintos ejércitos con los que trabaja el diccionario.

Ilustración 10- Muestra de fragmento del diccionario MILGLOSS con la información de interés para extraer los datos. Fuente: elaboración propia.

3.5 Otros conceptos

En este apartado se abordarán otros conceptos que, pese a estar en segundo plano dentro del marco teórico, precisan de su mención para la comprensión del trabajo.

3.5.1 Google Colaboratory, Colab

Colab permite a los usuarios crear y ejecutar cuadernos Jupyter en un entorno de nube de forma gratuita, sin necesidad de configurar su propio servidor o instalar software en su computadora. Un cuaderno Jupyter es un documento interactivo que puede contener código, texto y elementos multimedia, y se utiliza comúnmente para análisis de datos, aprendizaje automático, visualización de datos y otros fines científicos y de ingeniería e informática.

Además, Colab proporciona acceso a recursos de hardware acelerados, como GPU (que permite el manejo y empleo de imágenes, vídeos, etc.) y TPU (que permite manejar el procesamiento de redes neuronales), que son útiles para aplicaciones de aprendizaje automático. (Anon., 2019)

Así mismo se puede colaborar en un mismo cuaderno de Colab al compartir el URL con otros usuarios, lo que permite editar y ejecutar el código en tiempo real a diferentes usuarios, lo cual ha sido beneficioso en este proyecto por el trabajo compartido-supervisado alumno-tutor.

También pueden importar y exportar datos y modelos de Colab desde y hacia su computadora o desde y hacia otros servicios de almacenamiento en la nube, como Google Drive y GitHub.[1]

[1] GitHub funciona a modo de un "Google drive" de código, es decir, permite gestionar y almacenar código fuente en la nube, permitiendo un seguimiento y control del mismo por diferentes usuarios en línea; ofrece también la posibilidad de revisar el código y cuenta con herramientas de seguimiento de problemas. (Google, 2023)

3.5.2 Ontolex

El estándar OntoLex (Ontology-Lexicon) es una especificación del W3C (Consorcio World Wide Web) que indica cómo representar y modelar información lingüística (palabras, frases, etc.) en la web semántica. OntoLex proporciona una ontología para describir y conectar términos, palabras, frases y otros elementos lingüísticos con información semántica y ontológica.

El objetivo principal de OntoLex es mejorar la interoperabilidad y la reutilización de los recursos léxicos y terminológicos en la web semántica[2], lo que facilita la integración de diferentes fuentes
de datos lingüísticos y su uso en aplicaciones de procesamiento del lenguaje natural y la traducción automática.

OntoLex se basa en otros estándares relacionados con la web semántica, como RDF (Framework de Descripción de Recursos), RDFS (Esquemas de Recursos RDF) y OWL (Lenguaje Web Ontology). Además, OntoLex también está diseñado para ser compatible con otros estándares lingüísticos.
(Jorge Garcia, 2022)

En resumen, OntoLex es un estándar que permite modelar léxicos y terminologías en la web semántica de una forma consistente y reutilizable, lo que facilita su integración y uso en aplicaciones lingüísticas y de traducción automática.

En la imagen se observa, en la parte superior de la misma, las estructuras comunes en los recursos lexicográficos y en la parte inferior se describe el léxico, es decir, los elementos que describen la información léxica.

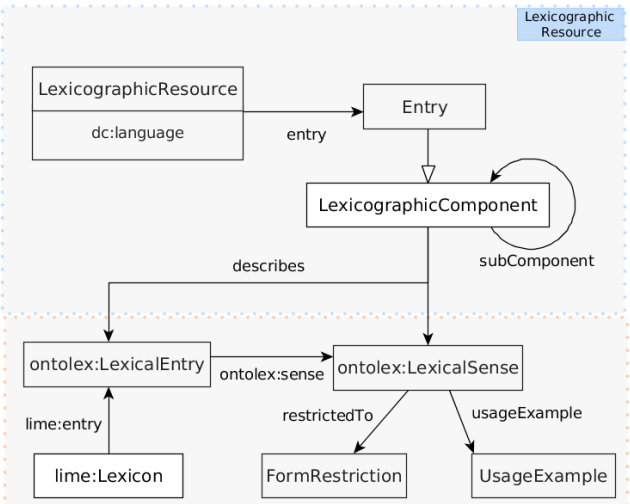

Ilustración 11- Módulo lexicográfico de Ontolex. Fuente: W3C - The Ontolex Lemon Lexicography Module (W3School, 2022)

3.5.3 Expresiones regulares y Regexper

Con el lenguaje de programación Python trabajaremos expresiones regulares.

Las expresiones regulares podrían definirse como un lenguaje diminuto dentro de Python de carácter muy especializado, el cual nos permite definir patrones para buscar y extraer texto de un documento o cadenas de texto que cumplan las características que buscamos. En resumen, es una secuencia de caracteres que define un patrón de búsqueda. (Goyvaerts, 2020)

[2] La Web Semántica es una extensión de la World Wide Web que tiene como objetivo hacer que el contenido de la web sea más legible para las máquinas y, por lo tanto, más útil para sus usuarios. En lugar de simplemente mostrar información a los usuarios, la Web Semántica trata de crear una estructura de datos que permita a las computadoras comprender el significado del contenido web.

Regexper es una página web a modo de una herramienta en línea que permite visualizar expresiones regulares y convertir estas expresiones regulares en diagramas de flujo visuales consiguiendo mostrar de forma gráfica lo que se está requiriendo con la expresión formulada. Es decir, es una herramienta que muestra de forma gráfica lo que la expresión va a hacer, ayudando a conocer su funcionamiento al mostrarlo de forma visual. (Avallone, 2021)

Por ejemplo, para la adquisición de una entrada del diccionario en inglés, ya sea con acrónimos o sin ellos, la expresión regular en nuestro caso es: "\n(.*?)(-.*)?", a simple vista y más para quien no esté familiarizado con la programación y con las expresiones regulares, esta sucesión de dígitos no aporta nada pero, si se muestra gráficamente lo que hace, se aclara algo más su empleo:

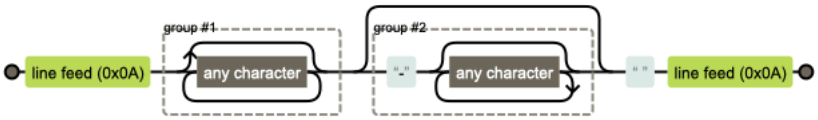

Ilustración 12- Representación gráfica de expresiones regulares a modo de ejemplo. Fuente: Regexper

Puede apreciarse que, después de un salto de línea, busca un grupo de cualesquiera que sean los caracteres más otro grupo que comienza con un guion (siendo este opcional). Más adelante se detalla con más precisión la función de cada expresión regular.

4 DESARROLLO: ANÁLISIS Y RESULTADOS

4.1 ANÁLISIS DE LOS REQUISITOS

Definidos los conceptos sobre la traducción, las necesidades y motivación del proyecto para las FAS y el problema que se trata de solventar, se precisa conocer y marcar los requisitos que la aplicación debe poseer para satisfacer las necesidades de los futuros usuarios.

Dejando a un lado los requisitos técnicos que un software debe cumplir, se ha realizado una encuesta informativa para conocer de primera mano las necesidades y requerimientos que el personal de las FAS solicita a este proyecto. La muestra ha sido seleccionada de la compañía de Mando y Apoyo del RI Canarias Nº50, considerada significativa y representativa del conjunto de las FAS por los motivos expuestos en el apartado 1.2. Esta encuesta cuenta con un total de 37 encuestados, siendo las preguntas de la misma las siguientes:

1. Indicar la escala a la que pertenecen
2. Indicar la especialidad fundamental a la que pertenecen
3. Indicar si ha sido desplegado en zona de operaciones
4. Indicar si han realizado ejercicios conjuntos con otros ejércitos
5. Elegir qué idiomas se consideran más importantes para ser adquiridos, pudiendo elegir más de uno entre francés, inglés, alemán, ruso, árabe y chino.
6. Indicar si han encontrado deficiencias al haber usado un traductor para la traducción de documentos de índole militar
7. Indicar qué requisitos cree que debería tener este traductor, siendo esta pregunta de carácter abierto, permitiendo a los encuestados escribir lo que requieran.

Los resultados obtenidos se muestran como anexos.

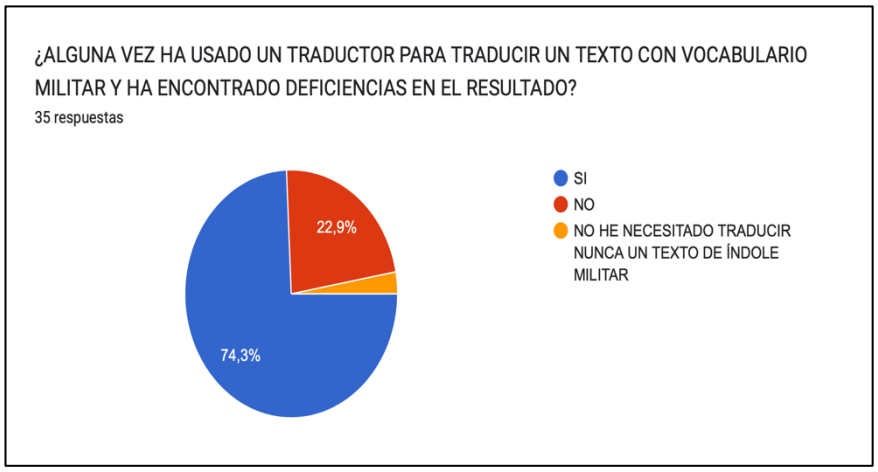

Ilustración 13- Encuesta realizada a través de google drive - elaboración propia

En resumen, en la encuesta la compone un 20% de oficiales, un 25% de suboficiales y un 55% de personal de tropa. Dentro de este personal, un 80% pertenece a la especialidad fundamental de infantería, un 6% a la de artillería, un 6% a la de caballería y un 8% a la de ingenieros. El 60% del personal ha sido desplegado alguna vez en zona de operaciones y el 75% ha participado en algún ejercicio conjunto con otros ejércitos.

Dato muy curioso es que, de los posibles idiomas a elegir, el 100% ha elegido entre sus opciones el inglés y como otras opciones relevantes se obtiene que un 37% selecciona el francés, un 28% el árabe y un 14% el ruso, quedando el chino y alemán en última posición con un 6% y 3% respectivamente.

75% de los encuestados han encontrado algún problema o deficiencia en el resultado de la traducción al haber empleado algún traductor para textos con terminología militar, un 23% no han tenido este problema y un 2% nunca han tenido la necesidad de traducir un documento de índole militar. Esta pregunta se muestra en la imagen 14 en la que, mediante representación gráfica, podemos observar el resultado.

Para la última pregunta "¿QUÉ REQUISITOS CREE QUE DEBERÍA TENER UN TRADUCTOR PARA QUE FUESE ÚTIL PARA DEFENSA?" se muestra una breve síntesis de los resultados obtenidos remarcando que 27 encuestados han respondido qué requisitos solicitan:

- 16 encuestados requieren que el traductor tenga un dominio específico de carácter militar (en vocabulario, frases, terminología, etc.).

- 4 encuestados requieren que cuente también con siglas OTAN.

- 4 encuestados requieren que pueda traducir también por voz

- En menos proporción (dos o menos encuestados) se han obtenido otros requisitos no significativos para el proyecto.

Con todos los datos obtenidos, se llega a la conclusión esperada: la encuesta valida la premisa de la necesidad de una herramienta de traducción algo más específica en terminología militar.

Por tanto, los requisitos que debe cumplir el sistema, una vez analizado todo lo anteriormente citado a un nivel más técnico, son:

- Contenido en inglés, francés y español, relacionado cada uno con su homónimo.

- Mayor relevancia al inglés/español.

4.2 DISEÑO DEL SISTEMA

4.2.1 Análisis del documento .pdf MILGLOSS y detección de patrones

La sección del diccionario sobre el que se realizará el preprocesado de datos es desde la página 18 hasta la 126. Se selecciona esta sección del mismo a modo de ejemplo dado que en ella se encuentra información de interés. En la siguiente ilustración se muestra un pequeño fragmento del MILGLOSS ya explicado anteriormente:

Ilustración 14- Ejemplo diccionario milgloss para la explicación de patrones presente. Fuente: elaboración propia a partir del MILGLOSS

Analizado el contenido del mismo, se observa la secuencia que tiene cada diferente entrada cumpliéndose el patrón para una misma entrada de:

- Palabra en inglés (azul negrito): pudiendo estar compuesta por más de una palabra, pero siempre termina en el tipo de palabra que es.

- Tipo de palabra (azul negrito cursiva): apareciendo siempre definido el tipo de palabra como adj, vt, vi, n o pp y seguido de un espacio.

- Definición en inglés (azul): pudiendo estar formada por más de una palabra y terminando en un espacio. Puede contar también con saltos de línea en ella.

- Palabra en español (azul): pudiendo estar formada por más de una palabra y terminando en un punto.

- Palabra en francés (verde cursivo): pudiendo estar formada por más de una palabra y terminando en un punto.

- Frases (marrón): hay otros patrones que son opcionales, en los que las palabras aparecen acompañadas de frases de ejemplo, siguiendo un patrón menos intuitivo y difícil de definir por completo dada su variedad. Pero por patrón general se tiene que, después del punto de la palabra en francés, aparece un grupo opcional que es la frase en cuestión, tanto en inglés como en español, separadas por "(*Spñl*)".

4.2.2 Extracción de datos y transformación a formato estándar (Ontolex)

Las RNA basadas en Transformers son los métodos actuales en TA. Dada la necesidad de muchos datos y la existencia de MILGLOSS, este modelo de aplicación se centra en los datos que serán extraídos de este diccionario que, constando de terminología militar, aborda el desafío que presenta la traducción de terminología técnica y especializada en un ámbito, como ya se ha mencionado.

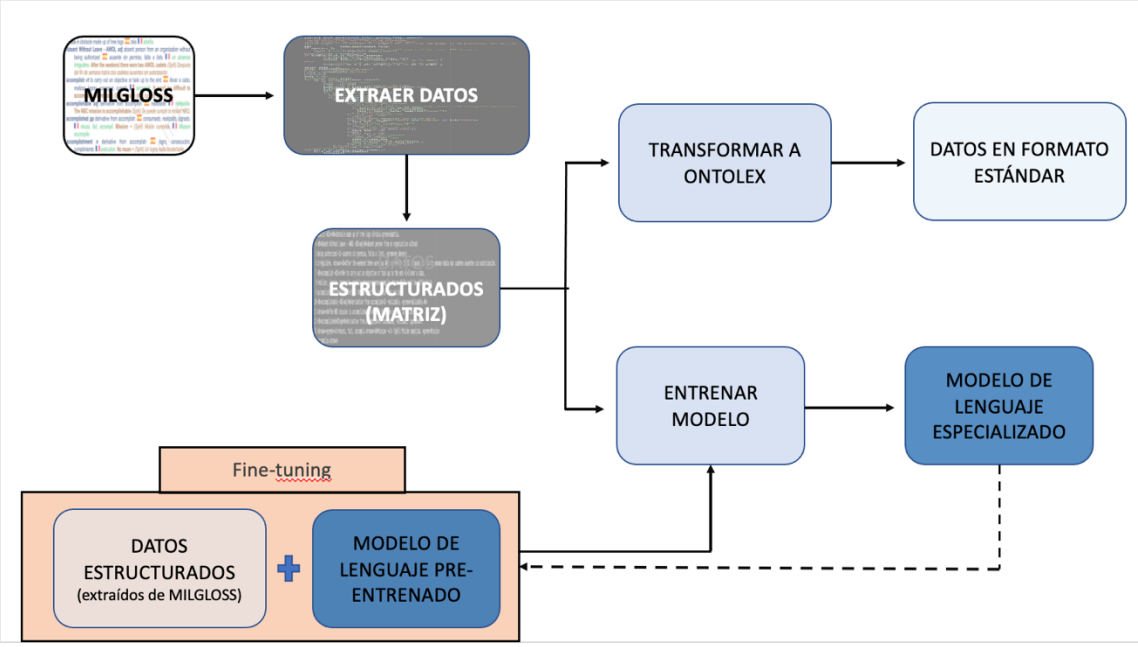

Ilustración 15- Esquema explicativo de extracción de datos / entrada MILGLOSS / datos estructurados del proyecto del modelo de aplicación. Fuente: elaboración propia en colaboración con la Dra. L. Dranca.

El contenido que se extrae de este diccionario, tanto palabras como las frases de ejemplo a pares, serán los inputs que utilizará el modelo de traducción para sí mismo y su entrenamiento, teniendo la frase en español con su homóloga en inglés para este proceso. A continuación, se muestra un esquema explicativo del proceso desde el procesamiento del diccionario hasta su definición en un formato estándar Ontolex, como ya se ha explicado en apartados anteriores.

Para poder entrenar el modelo ya se ha comentado que DL precisa de gran cantidad de datos de entrada y salida que sean paralelos, es decir, pares de oraciones en diferentes idiomas, para poder generar los modelos de RNA,s.

Con el llamado ajuste fino o fine-tuning a este modelo ya entrenado, se le incide y entrena con los datos de dominio específico ya extraídos de MILGLOSS, del que se consiguen extraer frases emparejadas que permitirá al modelo nutrirse de estos datos y re-entrenarse en este dominio específico. A continuación, se muestra un ejemplo tangible (ilustración 17) de la aportación del diccionario con una frase en inglés y su homóloga en español, tanto del pdf en sí como del archivo txt que generamos para poder ser procesado por nuestro ordenador.

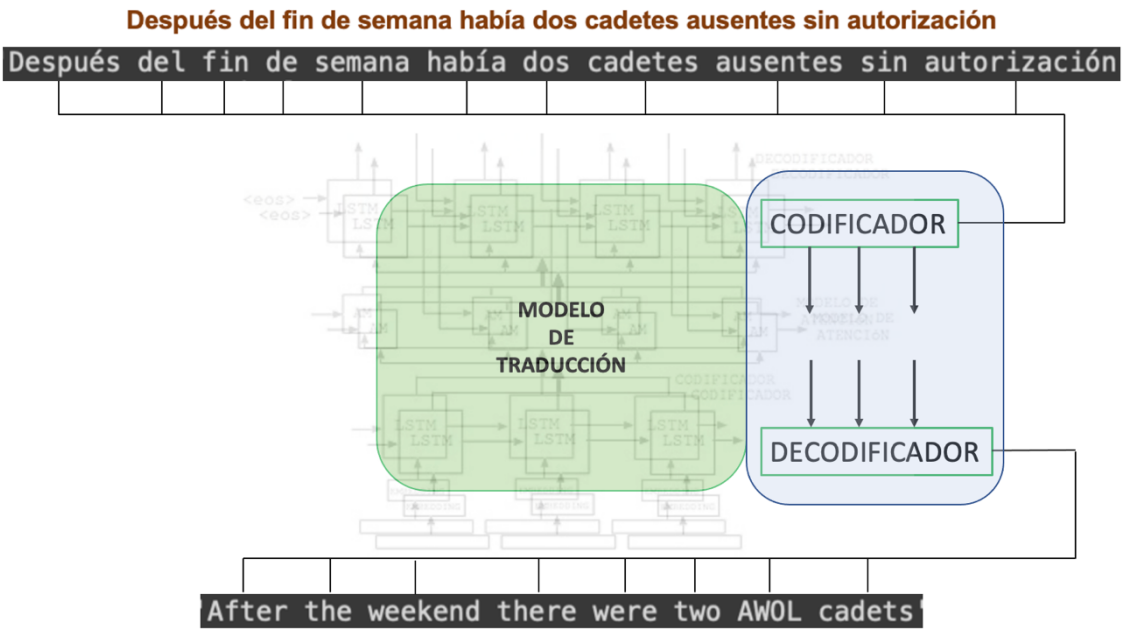

Ilustración 16- Par de frase inglés/español que utilizará el modelo de traducción. Fuente: elaboración propia

Los pasos seguidos para la extracción de datos han sido:

1. Procesamiento del documento pdf para generar un archivo txt procesable en Python.

2. Empleo de expresiones regulares para la generación de datos estructurados (matriz) según preferencias y necesidades.

Y los pasos seguidos para la definición a formato estándar:

1. Obtención de formato estándar Ontolex (explicado en apartados anteriores)

2. Elaboración de código para la definición de las entradas de interés (inglés, español, francés y relación inglés-español)

4.3 IMPLEMENTACIÓN DEL CÓDIGO

4.3.1 Elección del lenguaje y método de programación

Observado el patrón que sigue el documento y estudiadas las ventajas e inconvenientes de los diferentes métodos de obtención de información y lenguajes de programación, se considera que el lenguaje de programación que mejor se adapta a las necesidades requeridas es Python.

Python es considerado un buen lenguaje de programación para extraer información de un PDF por varias razones:

1. Tiene una gran cantidad de bibliotecas para procesamiento de PDF: Python cuenta con bibliotecas como PyPDF2, pdftotext, PyMuPDF, pdfminer, entre otras, que permiten el procesamiento de archivos PDF, desde la extracción de texto y metadatos, hasta la manipulación de imágenes y formas.

2. Es un lenguaje fácil de aprender: Python es un lenguaje de programación de alto nivel que tiene una sintaxis clara y sencilla, lo que facilita la programación de aplicaciones que involucren el procesamiento de archivos PDF y es similar al lenguaje pascal estudiado en una asignatura durante el grado.

3. Tiene una gran comunidad de desarrolladores: Python cuenta con una gran cantidad de desarrolladores en todo el mundo, lo que significa que hay muchas herramientas y bibliotecas disponibles para su uso en el procesamiento de archivos PDF y es sencillo encontrar ayudas para el desarrollo de código o funciones puntuales que puedan aportar al proyecto.

4. Es multiplataforma: Python es un lenguaje multiplataforma, lo que significa que se puede utilizar en diferentes sistemas operativos, como Windows, macOS y Linux. (Python Software Fundation, 2023)

4.3.2 Obtención y adaptación del código para el procesamiento del fichero pdf MILGLOSS

El código completo para la extracción del documento .pdf se encuentra en el anexo II.

Un primer bloque destinado a la búsqueda y obtención en la red de un código que nos permita procesar el pdf MILGLOSS. Aquí se generará un archivo de texto .txt con la información y datos únicamente que precisamos para trabajar. Sobre este archivo se trabajarán las expresiones regulares. Este código puede obtenerse de páginas webs y/o foros de programadores dado que es muy común y está extendido el uso de códigos para la extracción de texto de un pdf por lo que, para este caso concreto, basándonos en un código preexistente que se adapte lo mejor posible a las características del documento a procesar, con pequeñas modificaciones y adaptándolo a nuestro input, se obtiene el código que nos permite sacar la información deseada. (Rockikz, 2022).

Parte del código obtenido de *PythonCode* ha sido adaptado y mejorado también por la Dra. Lacramioara Dranca, adaptando este a las necesidades particulares del proyecto.

Para trabajar el código se hace imprescindible el uso de la biblioteca PyMuPDF. Esta es una biblioteca de Python que nos permite trabajar con archivos PDF utilizando la biblioteca MuPDF, la cual, es una biblioteca de procesamiento de PDF de código abierto y de alto rendimiento, que proporciona capacidades de renderizado, interpretación y manipulación de PDF. Algunas de las funciones que nos proporciona son:

- Leer y escribir archivos PDF.
- Extraer texto y metadatos de los archivos PDF.
- Renderizar PDF en diferentes formatos de imagen.
- Manipular el contenido de los archivos PDF, como imágenes, formas y marcadores.
- Crear nuevos archivos PDF y combinar archivos PDF existentes.
- Aplicar transformaciones a los archivos PDF, como rotar o recortar páginas. (Software, 2022)

Y así poder abordar los problemas con los que nos encontremos en cuanto a forma y/o contenido del pdf a la hora de querer extraer contenido filtrado.

Los principales problemas a los que se han dado solución son:

1. Patrones de colores que no pueden procesarse

2. Tipo de texto (itálico, bold, etc.)

A continuación, se muestra en la siguiente figura, parte del código que permite transformar el pdf y sus patrones de colores para facilitar la obtención de los datos necesarios mediante las expresiones regulares.

Puede observarse la edición de los patrones de color del pdf según el código de color; mostrándose como <brown>, <green> o <blue>, por ejemplo.

El objetivo final de esta parte de código es conseguir generar un archivo txt (anexo III) a partir del pdf desde el cual poder emplear las expresiones regulares de forma que los patrones sean más intuitivos y sencillos en su extracción.

```
def extract_text_dict(input_file, output_file, pages):
    ''' extract the text content of a pdf file
    :param input_file: pdf file to be processed
    :param output_file: txt file to write the result
    :param page: -1 to process all pages, or list with the pages to be processed, ex: [0,1,3]
    '''
    pdf          = fitz.open(input_file)
    if pages==-1:
        # if pages is not set, default is all pages of the input PDF document
        pages = list(range(pdf.page_count))

    output_files = output_file
    if output_file is not sys.stdout:
            # a single file, open it
            output_file = open(output_file, "w")
            output_files = { pn: output_file for pn in pages }
    else:
            # if output file is standard output, do not open
            output_files = { pn: output_file for pn in pages }

    color_code = {}
    color_code[9914117]="<brown>"
    #color_code[8031]="<blue>"
    color_code[44880]="<green>"
    color = 0
    c = False
    font = "N"
    for pg in range(pdf.page_count):
        if pg in pages:
            # get the page object
            page = pdf[pg]
            # get the dict structure of the page
            page_dict = page.get_text("dict",sort=True)
            # get the output file
            file = output_files[pg]
            #for each block in the page
            for bl in page_dict["blocks"]:
              line = ""
              #only for text text
              if "lines" in bl:
                for span in bl["lines"]:
                  #for each span in the line
                  for txt in span["spans"]:
                      #detect color has changed
                      if txt["color"] != c:
                        # if color is one of the monitored colors (see color_code definition)
                        if txt["color"] in color_code:
                          #concatenate color annotation
                          line = line + color_code[txt["color"]]
                        c = txt["color"]
                      #if text is a blank char
                      if txt["text"]==' ':
                        continue
                      #if text font contains italic or bold
                      elif "Bold" in txt["font"] and "Italic" in txt["font"]:
                        font_txt = "BI"
                      # if text font contains bold
                      elif "Bold" in txt["font"]:
                        font_txt = "B"
                      #if text font contains italic
                      elif "Italic" in txt["font"]:
                        font_txt = "I"
                      else:
                        font_txt = "N"
                      #detect font has changed
                      # print line to output file
                      print(line,file=file)
    # close the files
    for pn, f in output_files.items():
        if f is not sys.stdout:
            f.close()
```

Ilustración 17- Código implementado para poder acceder mediante expresiones regulares a los patrones de colores del pdf. Fuente: elaboración propia a partir del ejecutable

El producto final generado es el que se muestra a continuación en la ilustración 16, siendo la parte de arriba el original de MILGLOSS y la de abajo el txt sobre el que trabajaremos estas expresiones.

En la siguiente ilustración se observa un fragmento originario del diccionario MILGLOSS tal y como contamos con él. Como ya se ha comentado anteriormente, este pdf hay que procesarlo para generar a partir de él un archivo txt desde el que poder emplear las expresiones regulares para la extracción de la información necesaria. Este archivo txt puede observarse en la parte inferior de la ilustración.

abatis ***n*** obstacle made up of tree logs *tala.* *abattis.*
Absent Without Leave - AWOL ***adj*** absent person from an organization without being authorized *ausente sin permiso, falta a lista.* *en absence irrégulière.* **After the weekend there were two AWOL cadets** *(Spñl) Después del fin de semana había dos cadetes ausentes sin autorización.*
accomplish ***vt*** to carry out an objective or task up to the end *llevar a cabo, realizar, lograr, conseguir, cumplir.* *accomplir.* **It won't be difficult to accomplish this mission** *(Spñl) No será difícil cumplir esta misión.*
accomplishable ***adj*** derivative from accomplish *realizable.* *réalisable.* **The NBC mission is accomplishable** *(Spñl) Se puede cumplir la misión NBQ.*
accomplished ***pp*** derivative from accomplish *consumado, realizado, logrado.* *réussi, fait, accompli.* **Mission ~** *(Spñl) Misión cumplida.* *Mission accomplie.*

```
@MILGLOSS-AGM-DPTO. IDIOMAS
<B>MILITARY GLOSSARY ENG-ESP-FRA
abatis <BI>n<N>obstacle made up of tree logs <I>tala.<green>abattis.
<B>Absent Without Leave - AWOL <BI>adj<N>absent person from an organization without
being authorized <I> ausente sin permiso, falta a lista. <green>en absence
irrégulière. <brown><B>After the weekend there were two AWOL cadets<I> (Spñl) Después del fin de
<B>accomplish <BI>vt<N> to carry out an objective or task up to the end <I>llevar a cabo,
realizar, lograr, conseguir, cumplir.<brown><green>accomplir.<brown><B>It won't be difficult to
accomplish this mission <I>(Spñl) No será difícil cumplir esta misión.
<B>accomplishable <BI>adj<N>derivative from accomplish<I> realizable. <green>réalisable.<N>
<brown><B>The NBC mission is accomplishable<I>(Spñl) Se puede cumplir la misión NBQ.
<B>accomplished<BI>pp<N>derivative from accomplish<I> consumado, realizado, logrado<B>.
<brown><green><I>réussi, fait, accompli.<brown><B>Mission ~<I> (Spñl) Misión cumplida. <green>Mis
accomplie.<brown>
<B>accomplishment <BI>n<N>derivative fromaccomplish<I> logro, consecución,
cumplimiento.<green>exécution<N>.<brown><B>No mean ~<brown><I>(Spñl) Un logro nada desdeñable.
<B>accuracy <BI>n<N>exactness <I>precisión. <brown><green>précision. <brown><B>If the material we
weapon would have more accuracy<I> (Spñl) Si el material fuera mejor, ese arma tendría más precis
<B>accurate <BI>adj <N>exact <I>preciso.<green>précis<N>.<brown><B>That ammunition is really inac
<I>(Spñl) Esa munición es realmente imprecisa.
<B>ache <BI>n<N>pain<I>dolor. <green>douleur, mal.<brown><B>I've got a horrible pain in my leg<br
<brown>Tengo un dolor terrible en la pierna.<B>To have head/tummy/back ~<brown><I>Tener
dolor de cabeza/tripa/espalda. <green>Avoir mal à la tête/au ventre/au dos.<N>
<B>ache <BI>vi<N>to have spread and general pain<I>doler.<green>avoir mal<N>.<brown><B>My head is
after long hours of reading <I>(Spñl) Me duele la cabeza después de estar muchas horas leyendo<N>
achievable <BI>adj <N>derivative fromachieve<I> alcanzable, posible. <green>réalisable,
```

Ilustración 18- Muestra de obtención del archivo .txt a partir del archivo pdf MILGLOSS. Fuente: elaboración propia.

4.3.3 Uso de expresiones regulares

Python cuenta con una biblioteca incorporada llamada re (regular expression) para trabajar con expresiones regulares. Esta biblioteca proporciona una amplia gama de funciones y métodos para trabajar con expresiones regulares. Algunas de las funciones más utilizadas de la biblioteca re son:

- re.search(): Busca la primera ocurrencia de una expresión regular en una cadena de texto y devuelve un objeto "match" si se encuentra una coincidencia.

- re.findall(): Busca todas las ocurrencias de una expresión regular en una cadena de texto y devuelve una lista con todas las coincidencias encontradas.

- re.sub(): Reemplaza todas las ocurrencias de una expresión regular en una cadena de texto con una cadena dada.

Character	Description	Example	Try it
[]	A set of characters	"[a-m]"	Try it »
\	Signals a special sequence (can also be used to escape special characters)	"\d"	Try it »
.	Any character (except newline character)	"he..o"	Try it »
^	Starts with	"^hello"	Try it »
$	Ends with	"planet$"	Try it »
*	Zero or more occurrences	"he.*o"	Try it »
+	One or more occurrences	"he.+o"	Try it »
?	Zero or one occurrences	"he.?o"	Try it »
{}	Exactly the specified number of occurrences	"he.{2}o"	Try it »
|	Either or	"falls|stays"	Try it »
()	Capture and group		

Ilustración 19- Muestra de algunos caracteres de expresiones regulares. Fuente: w3schools

En la ilustración podemos observar diferentes símbolos que se utilizan en estas expresiones regulares y nos permiten diferentes acciones según los empleemos. (W3School, 2022)

Por ejemplo, en nuestro caso particular, si queremos extraer las diferentes entradas en inglés debemos utilizar la función que nos permita buscar todas las ocurrencias “findall()”. → re.findall(‘\n(.*?)(-.*)? Donde:

1. \n significa un salto de línea que junto a (.?*) lo que conseguimos es capturar la entrada al diccionario (palabras en inglés)

2. (-.*)? Captura un grupo, que es opcional, de acrónimo, para aquellas entradas que cuenten con él (siempre acompañadas de un guion en su inicio).

Lo que se consigue con ello se muestra a continuación:

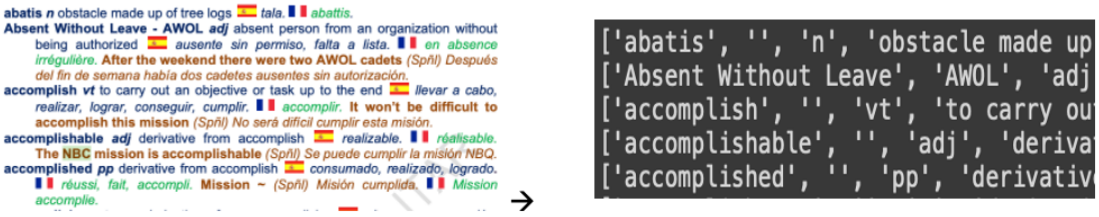

Ilustración 20- Muestra de obtención mediante expresiones regulares de las acepciones en inglés del diccionario MILGLOSS. Fuente: elaboración propia.

Teniendo en la parte izquierda de la ilustración una muestra tal y como aparece en el pdf MILGLOSS y a la derecha la matriz (anexo IV) que conseguimos a partir del archivo .txt la expresión regular empleada (siendo este ejemplo sólo la primera columna ya que aparecen también en la ilustración la continuación del producto final obtenido, que abarcaremos más adelante). El paso intermedio para poder haber obtenido la matriz mostrada en la parte derecha de la ilustración 15 es el archivo txt. No olvidar que una vez hemos generado este, las expresiones regulares con las que seleccionaremos los patrones que queremos extraer actuarán únicamente sobre el archivo txt. En este ejemplo, el fragmento de txt del que proviene se muestra en la siguiente ilustración.

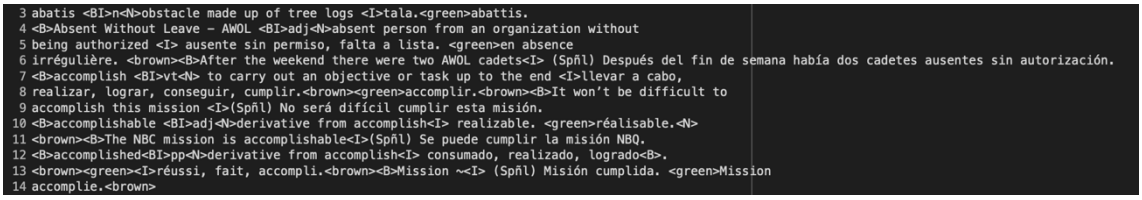

Ilustración 21- Fragmento de archivo txt sobre el que se emplean las expresiones regulares para la obtención de palabras y/o frases. Fuente: elaboración propia.

4.3.4 Desarrollo de la expresión regular para el preprocesado de datos final

Durante esta etapa del proceso de elaboración del código, ya de forma más autónoma dada la naturaleza del texto y sus particularidades específicas, se diseña un código mediante expresiones regulares que permitirá extraer los datos que se precisan y en la forma en que se precisan.

Ya se ha indicado anteriormente la forma en que se quieren extraer y los patrones que siguen los datos en su aparición en el documento. Llegados a este punto, mediante análisis, pruebas y ensayos se llega a un código final que ha permitido la extracción de datos deseada. Recordemos que hasta este momento y para poder trabajar con la expresión regular que a continuación se muestra, hemos realizado:

1. Procesado del pdf para poder trabajar con él de una forma accesible “haciéndolo tangible”

2. Generación archivo txt (anexo III) editando cierta información del pdf para poder trabajar desde aquí con las expresiones regulares, observando los patrones.

- El código de la expresión regular total es el siguiente: con él, se pretende obtener la información deseada en forma de matriz según las diferentes entradas del diccionario. Se pretende obtener → *palabra (EN)/tipo de palabra/definición/palabra (ES)/palabra (FRA)/frases ejemplo (opcional)*

x = re.findall('\n(.*?)(-.*)?
?<Bl>(adj|pp|vt|n|vi)((?:.|\n)*?.*?)<l>((?:.|\n)*?.*?)<green>((?:.|\n)*?.*?)\.(?:.*?<brown>((?:.|
\n)*?)<l>.*\(Spñl\)((?:.|\n)*?)(?:\.|\?))?',txt)

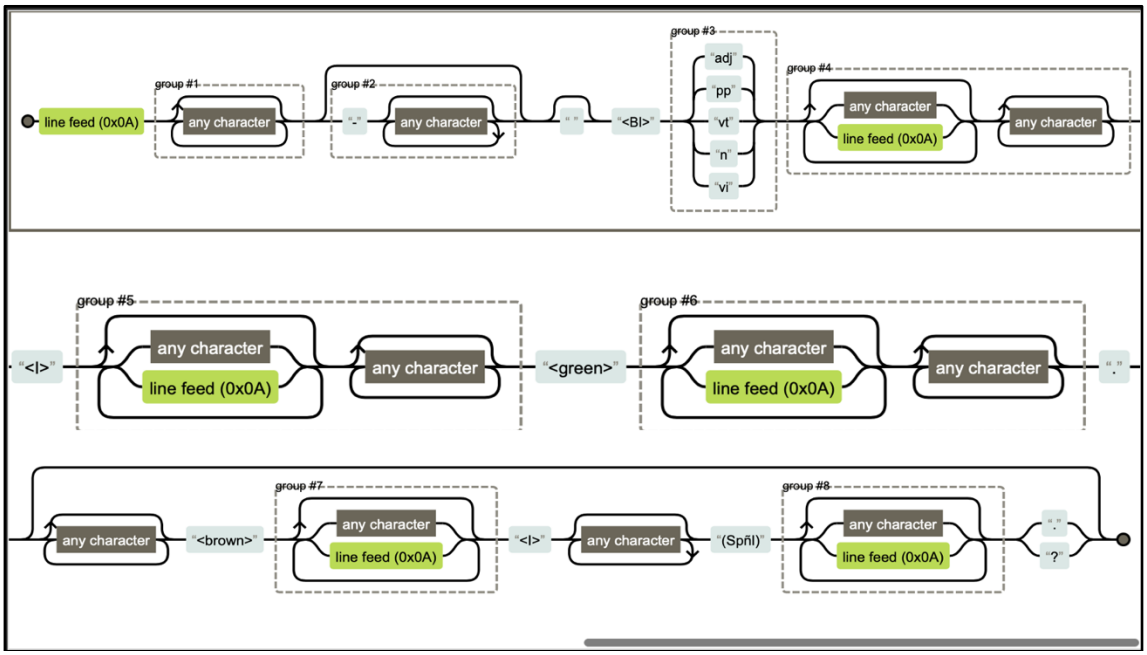

Ilustración 22- Representación gráfica del total de la expresión regular para la extracción de todos los datos precisos. Fuente: Regexper

En los siguientes puntos se desglosa en sí cada grupo de datos que se han querido extraer y su representación según regexper para su mejor comprensión.

- Entradas del diccionario: Pretendiéndose escoger todas y cada una de las entradas del diccionario que hacen relación a la palabra en inglés. Esta entrada del diccionario en inglés comienza con un salto de línea, le siguen caracteres. Puede tener un grupo opcional de acrónimo, empezando este siempre con un guion. Para saber hasta dónde llega esta entrada, se sabe que termina con un <BI>. ***Código: \n(.*?)(-.*)?***

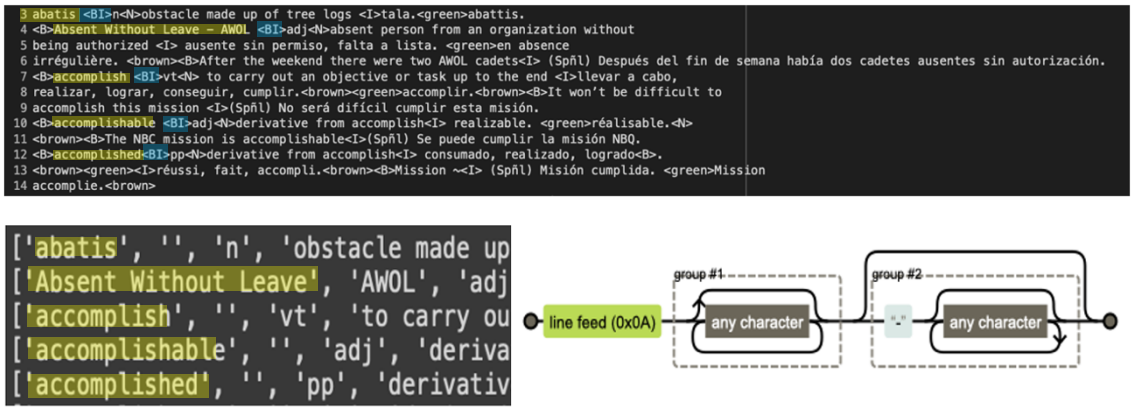

Ilustración 23- Obtención de entrada en inglés con expresiones regulares. Desde archivo txt se emplean expresiones regulares para obtener diferentes grupos (esquema ilustrativo de regexper). Fuente: elaboración propia

- Tipo de palabra: obteniéndose los diferentes tipos de palabras que pueden aparecer según la entrada del diccionario. Estas sólo pueden ser un verbo transitivo (vt), verbo intransitivo (vi), un nombre (n), un adjetivo (adj) o past participle (pp). Para ello se indica que obtenga aquello que coincida con "<BI>" más uno de los tipos señalados, significando "|" que puede ser cualquiera de lo citado. ***Código: <BI>(adj|pp|vt|n|vi)***

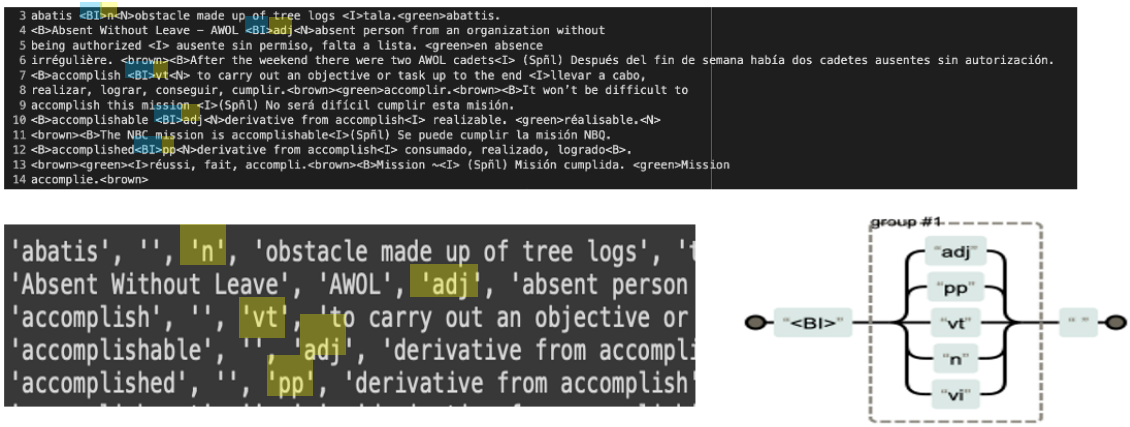

Ilustración 24- Obtención de tipo de palabra con expresiones regulares. Desde archivo txt se emplean expresiones regulares para obtener diferentes grupos (esquema ilustrativo de regexper). Fuente: elaboración propia

- Definición de la entrada: obteniéndose la propia definición de la entrada (en inglés, como viene en el diccionario) estando formada esta por un grupo de palabras y/o signos, existiendo a veces saltos de línea en una misma definición. ***Código:*** ***((?:.|\n)*?.*?)***

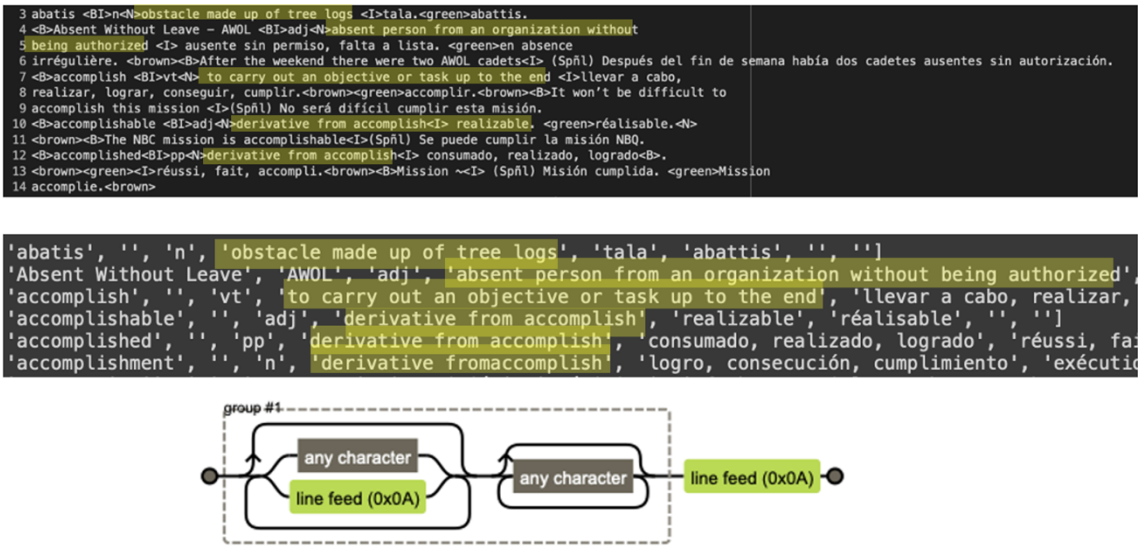

Ilustración 25- Obtención de la definición (inglés) con expresiones regulares. Desde archivo txt se emplean expresiones regulares para obtener diferentes grupos (esquema ilustrativo de regexper). Fuente: elaboración propia

- Traducción al español: siendo esta parte de expresión igual que para obtener la propia definición, pero diferenciándose en el indicador de "<I>", consiguiendo obtener el texto que queremos que viene después de ese indicador. ***Código:*** ***<I>((?:.|\n)*?.*?)***

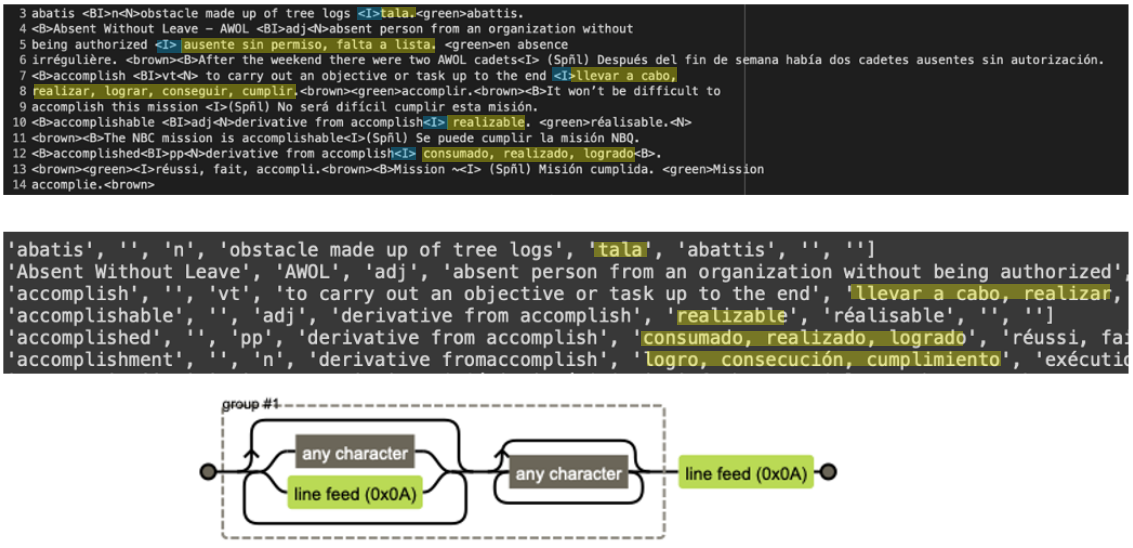

Ilustración 26- Obtención de la traducción al español con expresiones regulares. Desde archivo txt se emplean expresiones regulares para obtener diferentes grupos (esquema ilustrativo de regexper). Fuente: elaboración propia

- Traducción al francés: siendo esta parte de expresión igual que para obtener la propia definición o la traducción al español, pero diferenciándose en el indicador de "<green>", consiguiendo obtener el texto que queremos que viene después de ese indicador.***Código:*** ***<green>((?:.|\n)*?.*?)***

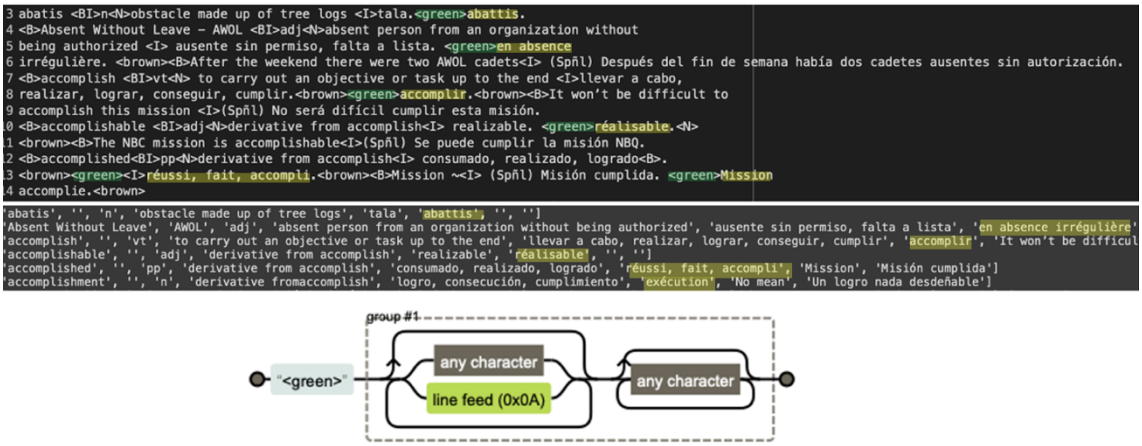

Ilustración 27- Obtención de la traducción al francés con expresiones regulares. Desde archivo txt se emplean expresiones regulares para obtener diferentes grupos (esquema ilustrativo de regexper). Fuente: elaboración propia

- Frase de ejemplo en inglés: para obtener esta frase cabe recalcar que es un elemento opcional, es decir, hay entradas que tienen frases de ejemplo y otras que no, ¿por lo que se utiliza el término regular de "?:". En este caso concreto buscamos patrones que empiecen en un punto dado que, toda entrada traducida al francés acaba en un punto y le sigue (o no) un espacio con <Brown>. ***Código:*** ***\.(?:.*?<brown>(?:.|\n)*?)<I>.****

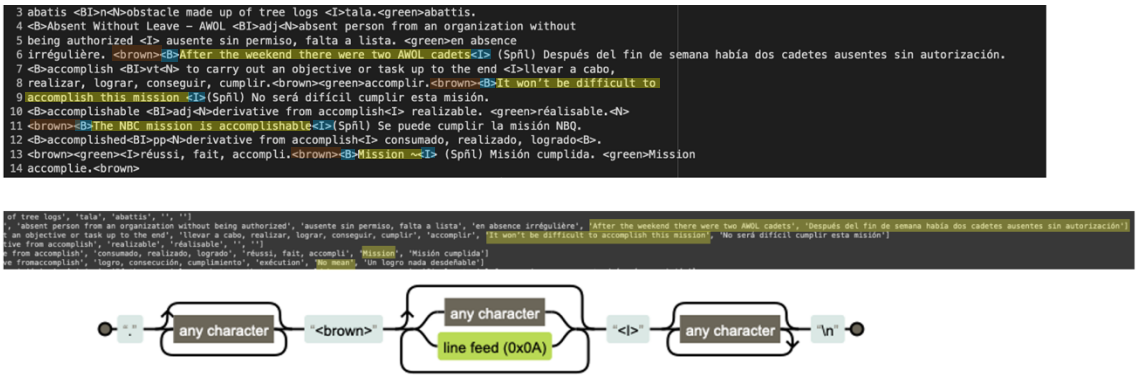

Ilustración 28- Obtención de la frase de ejemplo en inglés con expresiones regulares. Desde archivo txt se emplean expresiones regulares para obtener diferentes grupos (esquema ilustrativo de regexper). Fuente: elaboración propia

- Frase de ejemplo en español: este código es muy similar al anterior, con la particularidad de que, en este caso, se busca que la frase contenga al principio "(Spñl)". ***Código: (Spñl)((?:.|\n)*?)(?:\.|?))?***

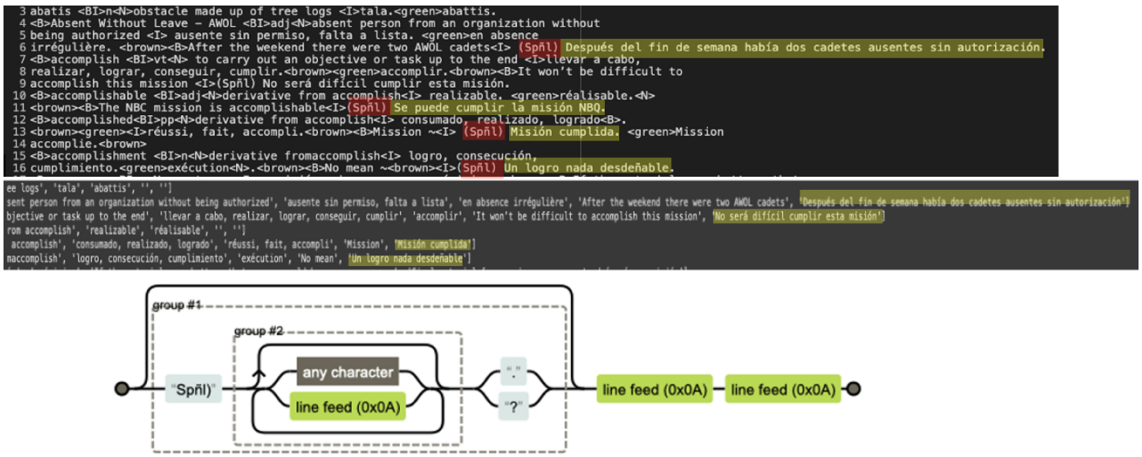

Ilustración 29- Obtención de la frase de ejemplo en español con expresiones regulares. Desde archivo txt se emplean expresiones regulares para obtener diferentes grupos (esquema ilustrativo de regexper). Fuente: elaboración propia

4.3.5 Limpieza de datos

La parte final del código para el manejo de los datos consiste en, una vez obtenida la información que hemos requerido mediante la expresión regular, "limpiar" esta eliminando signos de puntuación, espacios e indicadores que no aportan información y no permiten una lectura clara de los datos, sino que sólo ayudaban a la propia extracción y/o generación de patrones.

El código empleado para ello es el siguiente:

```
for j in range(0,len(x[i]),1):
  x[i][j] = re.sub("\n","",x[i][j])
  x[i][j] = re.sub("<N>","",x[i][j])
  x[i][j] = re.sub("\(a\)","",x[i][j])
  x[i][j] = re.sub("\(b\)","",x[i][j])
  x[i][j] = re.sub("<brown>","",x[i][j])
  x[i][j] = re.sub("(Spñl)","",x[i][j])
  x[i][j] = re.sub("<~>","",x[i][j])
  x[i][j] = re.sub("<I>","",x[i][j])
  x[i][j] = re.sub("<BI>","",x[i][j])
  x[i][j] = re.sub("\(Am Eng\)","",x[i][j])
  x[i][j] = re.sub("\(c\)","",x[i][j])
  x[i][j] = re.sub("\(Br Eng\)","",x[i][j])
  x[i][j] = re.sub("\[plural\]","",x[i][j])
  x[i][j] = re.sub("<green>","",x[i][j])
  x[i][j] = re.sub("87 ®MILGLOSS-AGM-DPTO","",x[i][j])
  x[i][j] = re.sub("<B>","",x[i][j])
  x[i][j] = re.sub("\.","",x[i][j])
  x[i][j] = re.sub("\~","",x[i][j])
  if j==1:
    x[i][j] = re.sub('- ','',x[i][j])

  x[i][j] =x[i][j].strip()
```

Ilustración 30- Función re.sub que permite limpiar los datos obtenidos con expresiones regulares, sustituyendo términos por otros

Todo el código anteriormente desarrollado tiene la finalidad ya mencionada, ordenar y limpiar los datos que nos proporciona el diccionario para poder utilizar estos como una matriz de dos dimensiones permitiendo trabajar con ella más fácilmente a la hora de definir cada posición de un vector ([i],[j]) en estándar Ontolex con la función “for” en Python.

A continuación, se muestra el uso de estas funciones para la limpieza de los datos. Se compara la matriz antes y después de su data cleaning mediante el empleo del código mostrado en la anterior ilustración.

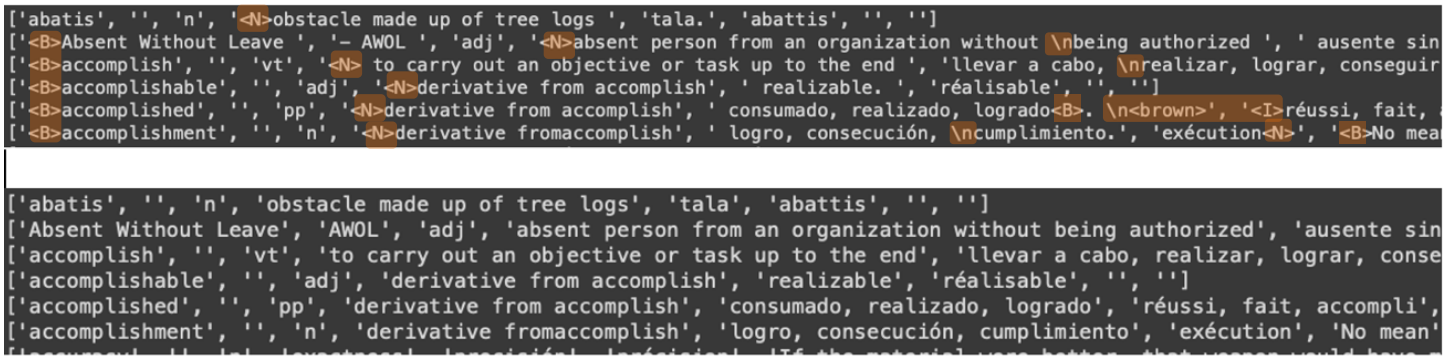

Ilustración 31- Comparativa de antes y después de la limpieza de datos sobre la matriz extraída del archivo txt. Fuente: elaboración propia

4.3.6 Definición de los datos en estándar Ontolex

Se busca con ello definir cada palabra en inglés, en español y en francés por separado, así como la relación que guardaría una palabra en inglés con su homónima en español. Para definir estas palabras con Ontolex se genera un código que permite realizar esta acción de forma automática, ya que, para la definición de estas, siguen todas, un mismo patrón en el que sólo varía la entrada en sí del diccionario. Mediante el desarrollo de código, se recorre de forma iterativa la matriz de datos extraídos del diccionario tanto en vertical (entradas del diccionario) como en horizontal (diferentes apartados de una misma entrada).

A continuación, se muestra un ejemplo del código empleado (para la entrada *accomplish*) donde se observa el texto pre-definido al que se le inserta el o los elementos necesarios de cada entrada de la matriz para su definición, generando todo ello un archivo .txt que dará como resultado final del proyecto un total de cuatro archivos: las palabras en inglés, en español, en francés y la relación entre el español e inglés.

- Palabra en inglés:

```
[9] import os

    f = open('lexEN.txt', 'w') #crear fichero ENGLISH
    f.write("@prefix milgloss: <https://ejercito.defensa.gob.es/unidades/Zaragoza/agm/> ." + os.linesep)
    f.write("@prefix dc: <http://purl.org/dc/elements/1.1/> ." + os.linesep)
    f.write("@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> ." + os.linesep)
    f.write("@prefix foaf: <http://xmlns.com/foaf/0.1/> ." + os.linesep)
    f.write("@prefix lexinfo: <http://www.lexinfo.net/ontology/2.0/lexinfo#> ." + os.linesep)
    f.write("@prefix xsd: <http://www.w3.org/2001/XMLSchema#> ." + os.linesep)
    f.write("@prefix owl: <http://www.w3.org/2002/07/owl#> ." + os.linesep)
    f.write("@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> ." + os.linesep)
    f.write("@prefix ontolex: <http://www.w3.org/ns/lemon/ontolex#> ." + os.linesep)
    f.write("@prefix lime: <http://www.w3.org/ns/lemon/lime#> ." + os.linesep)
    f.write("@prefix : <https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss/lexiconEN/> ." + os.linesep)
    f.write("")

    for i in range(0,len(x),1):
      f.write("<http://https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss/lexiconEN> lime:entry :"+x[i][0]+"-"+x[i][3]+"-en ." + os.linesep)
      f.write(":"+x[i][0]+"-"+x[i][3]+"-en a ontolex:LexicalEntry;" + os.linesep)
      f.write(" lexinfo:morphosyntacticProperty milgloss:"+x[i][3]+";" + os.linesep)
      f.write("ontolex:lexicalForm :"+x[i][0]+"-"+x[i][3]+"-en-form;" + os.linesep)
      f.write(" dc:source <https://github.com/milgloss/apertium-trunk.git> . " + os.linesep)
      f.write(":"+x[i][0]+"-"+x[i][3]+"-en-form a ontolex:Form; " + os.linesep)
      f.write('contolex:writtenRep "'+x[i][0]+'"@en .' + os.linesep)
      f.write(os.linesep)

    f.close()
```

```
<http://https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss/lexiconEN> lime:entry :accomplish-to carry out an objective or task up to the end-en .
:accomplish-to carry out an objective or task up to the end-en a ontolex:LexicalEntry;
 lexinfo:morphosyntacticProperty milgloss:to carry out an objective or task up to the end;
ontolex:lexicalForm :accomplish-to carry out an objective or task up to the end-en-form;
 dc:source <https://github.com/milgloss/apertium-trunk.git> .
:accomplish-to carry out an objective or task up to the end-en-form a ontolex:Form;
contolex:writtenRep "accomplish"@en .
```

Ilustración 32- En la parte superior, código para la definición con Ontolex de las entradas en inglés y en la parte inferior, ejemplo de la obtención en archivo txt de la palabra accomplish. Fuente: elaboración propia.

- Palabra en español:

```
import os

f = open('lexES.txt', "w") #crear fichero ESPAÑOL
f.write("@prefix milgloss: <https://ejercito.defensa.gob.es/unidades/Zaragoza/agm/> ." + os.linesep)
f.write("@prefix dc: <http://purl.org/dc/elements/1.1/> ." + os.linesep)
f.write("@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> ." + os.linesep)
f.write("@prefix foaf: <http://xmlns.com/foaf/0.1/> ." + os.linesep)
f.write("@prefix lexinfo: <http://www.lexinfo.net/ontology/2.0/lexinfo#> ." + os.linesep)
f.write("@prefix xsd: <http://www.w3.org/2001/XMLSchema#> ." + os.linesep)
f.write("@prefix owl: <http://www.w3.org/2002/07/owl#> ." + os.linesep)
f.write("@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> ." + os.linesep)
f.write("@prefix ontolex: <http://www.w3.org/ns/lemon/ontolex#> ." + os.linesep)
f.write("@prefix lime: <http://www.w3.org/ns/lemon/lime#> ." + os.linesep)
f.write("@prefix : <https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss/lexiconES/> ." + os.linesep)
f.write("")

for i in range(0,len(x),1):

  f.write("<http://https://ejercit  .gob.es/unidades/Zaragoza/agm//id/milgloss/lexiconEN> lime:entry :"+x[i][4]+"-"+x[i][3]+"-es ." + os.linesep)
  f.write(":"+x[i][4]+"-"+x[i][3]+  tolex:LexicalEntry;" + os.linesep)
  f.write(" lexinfo:morphosyntacti  milgloss:"+x[i][4]+";" + os.linesep)
  f.write("ontolex:lexicalForm :"+x[i][4]+"-"+x[i][3]+"-es-form;" + os.linesep)
  f.write(" dc:source <https://github.com/milgloss/apertium-trunk.git> . " + os.linesep)
  f.write(":"+x[i][4]+"-"+x[i][3]+"-es-form a ontolex:Form; " + os.linesep)
  f.write("contolex:writtenRep "+x[i][0]+"@es ." + os.linesep)
  f.write("" + os.linesep)

f.close()
```

int: i
1417

```
<http://https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss/lexiconEN> lime:entry :llevar a cabo, realizar, lograr, conseguir, cumplir-to carry out an objective or task up to the end-es .
:llevar a cabo, realizar, lograr, conseguir, cumplir-to carry out an objective or task up to the end-es a ontolex:LexicalEntry;
 lexinfo:morphosyntacticProperty milgloss:llevar a cabo, realizar, lograr, conseguir, cumplir;
ontolex:lexicalForm :llevar a cabo, realizar, lograr, conseguir, cumplir-to carry out an objective or task up to the end-es-form;
 dc:source <https://github.com/milgloss/apertium-trunk.git> .
:llevar a cabo, realizar, lograr, conseguir, cumplir-to carry out an objective or task up to the end-es-form a ontolex:Form;
contolex:writtenRep accomplish@es .
```

Ilustración 33- En la parte superior, código para la definición con Ontolex de las entradas en español y en la parte inferior, ejemplo de la obtención en archivo txt de la palabra lograr. Fuente: elaboración propia.

- Palabra en francés:

```
import os

f = open('lexFR.txt', "w") #crear fichero FRANÇAIS
f.write("@prefix milgloss: <https://ejercito.defensa.gob.es/unidades/Zaragoza/agm/> ." + os.linesep)
f.write("@prefix dc: <http://purl.org/dc/elements/1.1/> ." + os.linesep)
f.write("@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> ." + os.linesep)
f.write("@prefix foaf: <http://xmlns.com/foaf/0.1/> ." + os.linesep)
f.write("@prefix lexinfo: <http://www.lexinfo.net/ontology/2.0/lexinfo#> ." + os.linesep)
f.write("@prefix xsd: <http://www.w3.org/2001/XMLSchema#> ." + os.linesep)
f.write("@prefix owl: <http://www.w3.org/2002/07/owl#> ." + os.linesep)
f.write("@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> ." + os.linesep)
f.write("@prefix ontolex: <http://www.w3.org/ns/lemon/ontolex#> ." + os.linesep)
f.write("@prefix lime: <http://www.w3.org/ns/lemon/lime#> ." + os.linesep)
f.write("@prefix : <https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss/lexiconFR/> ." + os.linesep)
f.write("")

for i in range(0,len(x),1):


  f.write("<http://https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss/lexiconFR> lime:entry :"+x[i][5]+"-"+x[i][3]+"-fr ." + os.linesep)
  f.write(":"+x[i][5]+"-"+x[i][3]+"-fr a ontolex:LexicalEntry;" + os.linesep)
  f.write(" lexinfo:morphosyntacticProperty milgloss:"+x[i][5]+";" + os.linesep)
  f.write("ontolex:lexicalForm :"+x[i][5]+"-"+x[i][3]+"-fr-form;" + os.linesep)
  f.write(" dc:source <https://github.com/milgloss/apertium-trunk.git> . " + os.linesep)
  f.write(":"+x[i][5]+"-"+x[i][3]+"-fr-form a ontolex:Form; " + os.linesep)
  f.write('contolex:writtenRep "'+x[i][0]+'"@fr .' + os.linesep)
  f.write("" + os.linesep)
f.close()
```

```
<http://https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss/lexiconFR> lime:entry :accomplir-to carry out an objective or task up to the end-fr .
:accomplir-to carry out an objective or task up to the end-fr a ontolex:LexicalEntry;
 lexinfo:morphosyntacticProperty milgloss:accomplir;
ontolex:lexicalForm :accomplir-to carry out an objective or task up to the end-fr-form;
 dc:source <https://github.com/milgloss/apertium-trunk.git> .
:accomplir-to carry out an objective or task up to the end-fr-form a ontolex:Form;
contolex:writtenRep "accomplish"@fr .
```

Ilustración 34- En la parte superior, código para la definición con Ontolex de las entradas en francés y en la parte inferior, ejemplo de la obtención en archivo txt de la palabra lograr. Fuente: elaboración propia.

- Relación de palabras inglés-español:

```
import os

f = open('lexTRD_ES_EN.txt', "w") #crear fichero relación ES-EN
f.write("@prefix milgloss: <https://ejercito.defensa.gob.es/unidades/Zaragoza/agm/> ." + os.linesep)
f.write("@prefix dc: <http://purl.org/dc/elements/1.1/> ." + os.linesep)
f.write("@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> ." + os.linesep)
f.write("@prefix foaf: <http://xmlns.com/foaf/0.1/> ." + os.linesep)
f.write("@prefix lexinfo: <http://www.lexinfo.net/ontology/2.0/lexinfo#> ." + os.linesep)
f.write("@prefix xsd: <http://www.w3.org/2001/XMLSchema#> ." + os.linesep)
f.write("@prefix owl: <http://www.w3.org/2002/07/owl#> ." + os.linesep)
f.write("@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> ." + os.linesep)
f.write("@prefix ontolex: <http://www.w3.org/ns/lemon/ontolex#> ." + os.linesep)
f.write("@prefix lime: <http://www.w3.org/ns/lemon/lime#> ." + os.linesep)
f.write("@prefix vartrans: <http://www.w3.org/ns/lemon/vartrans#> ." + os.linesep)
f.write("@prefix : <https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss/tranSetES-EN/> ." + os.linesep)
f.write("")
f.write("@prefix src: <https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss/lexiconES/> . " + os.linesep)
f.write("")
f.write("@prefix tgt: <https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/apertium/lexiconEN/> ." + os.linesep)
f.write("")

for i in range(0,len(x),1):



  #HASTA AQUÍ. DESDE AQUÍ SI:
  f.write("<http://https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss/tranSetES-EN> " + os.linesep)
  f.write("vartrans:trans :"+x[i][4]+'_'+x[i][0]+'-'+x[i][3]+"es-sense-"+x[i][0]+'_'+x[i][4]+'-'+x[i][3]+"-en-sense-trans ." + os.linesep)
  f.write(":"+x[i][4]+'_'+x[i][0]+'-'+x[i][3]+"-es-sense-"+x[i][0]+'_'+x[i][4]+'-'+x[i][3]+"-en-sense-trans a vartrans:Translation;" + os.linesep)
  f.write("vartrans:source :"+x[i][4]+'_'+x[i][0]+'-'+x[i][3]+"-es-sense;" + os.linesep)
  f.write("vartrans:target :"+x[i][0]+'_'+x[i][4]+'-'+x[i][3]+"-en-sense ." + os.linesep)
  f.write(":"+x[i][4]+'_'+x[i][0]+'-'+x[i][3]+"es-sense a ontolex:LexicalSense;" + os.linesep)
  f.write("ontolex:isSenseOf src:"+x[i][4]+"-"+x[i][3]+"-es ." + os.linesep)
  f.write(":"+x[i][0]+'_'+x[i][4]+'-'+x[i][1]+"-en-sense a ontolex:LexicalSense;" + os.linesep)
  f.write("ontolex:isSenseOf tgt:"+x[i][0]+'-'+x[i][3]+"-en ." + os.linesep)
  f.write("" + os.linesep)
f.close()
```

```
<http://https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss/tranSetES-EN>
vartrans:trans :llevar a cabo, realizar, lograr, conseguir, cumplir_accomplish-to carry out an objective or task up to the endes-sense-accomplish_llevar a cabo, realizar, lograr, conseguir, cumplir-to carry out an objective or task up to the end-en-sens
:llevar a cabo, realizar, lograr, conseguir, cumplir_accomplish-to carry out an objective or task up to the end-es-sense-accomplish_llevar a cabo, realizar, lograr, conseguir, cumplir-to carry out an objective or task up to the end-en-sense-trans a vart
vartrans:source :llevar a cabo, realizar, lograr, conseguir, cumplir_accomplish-to carry out an objective or task up to the end-es-sense;
vartrans:target :accomplish_llevar a cabo, realizar, lograr, conseguir, cumplir-to carry out an objective or task up to the end-en-sense .
:llevar a cabo, realizar, lograr, conseguir, cumplir_accomplish-to carry out an objective or task up to the endes-sense a ontolex:LexicalSense;
ontolex:isSenseOf src:llevar a cabo, realizar, lograr, conseguir, cumplir-to carry out an objective or task up to the end-es .
:accomplish_llevar a cabo, realizar, lograr, conseguir, cumplir-en-sense a ontolex:LexicalSense;
ontolex:isSenseOf tgt:accomplish-to carry out an objective or task up to the end-en .
```

Ilustración 35- Definición en Ontolex de la relación entre las entradas (inglés) y su traducción al español. En la parte superior aparece el código para generar su definición y en la inferior el archivo txt generado. Fuente: elaboración propia.

5 CONCLUSIONES, LIMITACIONES Y LÍNEAS FUTURAS

Para finalizar el trabajo se van a explicar las conclusiones obtenidas durante su realización y las líneas futuras que deja abiertas para acciones futuras.

Como objetivo principal del trabajo se tenía la extracción, organización y definición de los datos obtenidos de MILGLOSS. Otros objetivos eran la obtención en red y adaptación de un código que permita procesar el pdf para su manipulación y futura extracción de datos, definición de los requisitos, así como proponer un posible modelo de aplicación para futuras acciones.

Estos objetivos enlazan con líneas futuras de cara a poder generar datos interoperables por diferentes sistemas que pretendan emplearlos en traductores ya existentes.

5.1 CONCLUSIONES

- Sobre la obtención en red de un código y adaptación del mismo que permitiese procesar el pdf se puede concluir que, tras una búsqueda de códigos existentes para procesar pdf (ya mencionado en apartados anteriores) se tuvo que proceder a su adaptación particular. Se encontraron ciertas dificultades a la hora de ejecutar el código y obtener el archivo txt del MILGLOSS ya que, los patrones que a simple vista se observaban, en el txt se perdían. Se tuvo que incluir código que permitiese hacer patrones según los colores del pdf y los tipos de letras. La elaboración de esta parte del proyecto, como ya se ha mencionado anteriormente, fue gracias a la contribución de la Dra. Lacramioara Dranca. Una vez solventado el problema de los patrones, se pudo continuar con el desarrollo del proyecto.

- Sobre la organización de los datos se procedió con expresiones regulares. Sin gran dificultad y tras varias pruebas, se consiguió la expresión regular que permite extraer los datos que se catalogaron como de interés dentro del MILGLOSS. Cierto es que, al no ser un documento 100% homogéneo en todas sus entradas, por la existencia de patrones diferentes en algunas entradas, la extracción de datos tiene, en alguna entrada, pequeñas excepciones por la presencia de algún símbolo no deseado. Este hecho no niega la consecución de los objetivos dado que permite la comprensión de los mismos y su empleo.

- Para la definición de los datos obtenidos a un estándar se cumplen los objetivos de definición de las palabras en inglés, español, francés y la relación inglés-español, como se perseguía en los objetivos. Tratándose de definir las frases y su relación, no se ha conseguido obtener un buen producto, por lo que este objetivo surgido durante la elaboración del proyecto, no se ha conseguido alcanzar, dejando paso en líneas futuras a su elaboración.

- Dado que el proceso de entrenamiento y ajuste de un modelo DL puede ser muy intensivo en términos de recursos computacionales y tiempo, se ha optado por hablar sobre cómo contribuirían los datos del MILGLOSS a un modelo de aplicación o traductor existentes

5.2 LIMITACIONES

Entre las limitaciones encontradas durante la realización del trabajo, las más destacables han sido: la poca adecuación del destino de prácticas con el tema del trabajo y la dificultad para una adecuada programación en tiempo de hitos y actividades alcanzables.

En primer lugar, destacar que durante una primera etapa del proyecto, se trabaja destinado en unidades del ET tanto para la recopilación de información como para el trato y conocimiento de primera mano, tanto de forma teórica como práctica, del tema a abordar durante el trabajo; en el destino de prácticas que se asigna este trabajo no se cuenta con personal ni medios que sirva de apoyo a tal proyecto y esto va en deterioro de su realización, tanto en materia de contenido como de programación temporal.

La dificultad en la planificación temporal de los hitos y actividades a realizar nace del problema tratado en primer lugar sumándole la dificultad de compaginar el régimen de vida y actividades de los diferentes centros en los que se ha trabajado, AGM y Academia de Infantería (ACINF).

5.3 LÍNEAS FUTURAS

Recordar que el fin global del proyecto es contribuir a la generación de datos (de índole militar, provenientes del MILGLOSS) para su futuro empleo en desarrollo y/o entreno de traductores dentro de un dominio militar. Por tanto, las líneas futuras del proyecto deben ir encaminadas a tal hecho y en concordancia con los objetivos alcanzados.

- La primera línea futura que surge del proyecto es la depuración total de los datos obtenidos. La depuración total de los datos ya extraídos consta de ir entrada por entrada analizando las anomalías. El código que habría que generar para la limpieza total debe ir enfocado a las excepciones que surgen en el empleo del código ya existente. La función más empleada en este tipo de data cleaning es re.sub() (en este proyecto), por lo que la línea futura y sus acciones irán en esta dirección.

- El diccionario MILGLOSS tiene más información de interés que la mostrada durante el proyecto por lo que, sería de interés obtener el total de la información que este contiene. En el proyecto se ha extraído la que se considera de mayor interés y que más valor aporta al proyecto, pero sería conveniente poder extraer el total de la información, así como en futuros proyectos, obtener documentos de índole militar que daten en dos idiomas, inglés y español, para que puedan contribuir en la recopilación de datos de interés.

- La definición de frases en inglés, español y su relación, objetivo intermedio surgido durante la elaboración del proyecto, sería una contribución futura siguiendo el mismo esquema que la definición llevada a cabo mediante estándar Ontolex.

- Como ya se ha comentado, la línea futura en general es que este proyecto sirva tanto como motivación para futuros traductores en dominio militar como un puente para los mismos, sirviendo parte del producto obtenido en el desarrollo del proyecto como input para futuras acciones y/o elaboración de un traductor.

6 REFERENCIAS BIBLIOGRÁFICAS

Ávila, R., s.f. *Opena Knowledge Fundation.* [En línea]
Available at: https://opendatahandbook.org/glossary/en/terms/machine-readable/
[Último acceso: 2023].
Abril, C. N. y. P., 2017. Traducción automática Neuronal. *Revista Tradumàtica,* Issue 15, pp. 66-74.
Anon., 2018. *Open knowledge fundation.* [En línea]
Available at: https://opendatahandbook.org/glossary/en/terms/machine-readable/
[Último acceso: 2023].
Anon., 2019. *Alura Latam.* [En línea]
Available at: https://www.aluracursos.com/blog/google-colab-que-es-y-como-usarlo
[Último acceso: diciembre 2022].
Anon., 2021. *Solver Intelligence Analytics.* [En línea]
Available at: https://iasolver.es/que-es-la-inteligencia-artificial/
Avallone, J., 2021. *Regexper.* [En línea]
Available at: https://regexper.com
Avallone, J., s.f. *regexper.* [En línea]
Available at: https://regexper.com
Bagnato, J. I., 2019. *aprende machine learning.* [En línea]
Available at: aprendemachinelearning.com
Bahdanau, D. ,. C. K. y. B. Y., 2014. *Neural machine translation.* s.l.:s.n.
Briva-Iglesias, V., 2021. Mutatis Mutandi. *Revista Latinoamericana de Traducción,* 14(22).
cañadas, r., 2022. *AbDatum.* [En línea]
Available at: https://abdatum.com/tecnologia/redes-neuronales-recurrentes
Defensa, M. d., 2016. *Defensa.gob.* [En línea]
Available at:
https://ejercito.defensa.gob.es/unidades/Granada/madoc/Noticias/2016/045.html
[Último acceso: diciembre 2022].
Edunov, S., 2018. Understanding Back - Translation at scale.
Escrivá, J. V. S., 2021. *Utilidad de las nuevas tecnologías en la mejora de la comunicación médico-paciente en el área de salud mental: aportaciones de la inteligencia artificial y el procesamiento del lenguaje natura.* s.l.:s.n.
F., D. G., 2022. *Metodologías de control de calidad y pruebas para soluciones tecnológicas basadas en AI/ML y Big Data..* s.l.:s.n.
Google, 2023. *Colab.* [En línea]
Available at: https://colab.research.google.com/?hl=es
Goyvaerts, J. y. S. L., 2020. *Regular Expressions Cookbook.* 2da ed. s.l.:s.n.
Hermida, J. &. Q. L., 2019. La hermenéutica como método de interpretación de textos en la investigación psicoanalítica. *Revista de psicología y ciencias afines,* 16(2), pp. 73-80.
Hernández-Sosa, D. &. F.-O., 2022. *Deep Learning for Diagonal Earlobe Crease Detection.* s.l.:s.n.
Jeff, L., 2021. s.l.:s.n.
Jorge Garcia, F. K., 2022. *W3C.* [En línea]
Available at: https://www.w3.org/community/ontolex/
[Último acceso: enero 2023].
Koehn, 2009. *no lo se.* s.l.:s.n.
MADOC, 2017. Tendencias según especialidades. *tendencias ,* Volumen II.

Peris, F. C. y. Á., 2017. Traducción Autonática Neuronal. *Tradumàtica,* Issue 15, pp. 66-74.
Python Software Fundation, 2023. *Biclioteca estándar de Python.* [En línea]
Available at: https://docs.python.org/es/3/library/index.html
[Último acceso: enero 2023].
Quijano, J., 2013. *Genbeta.* [En línea]
Available at: https://www.genbeta.com/desarrollo/regexper-ver-la-expresiones-regulares-de-forma-mas-humana
[Último acceso: Diciembre 2022].
Rockikz, A., 2022. *PythonCode.* [En línea]
Available at: https://www.thepythoncode.com/article/extract-text-from-pdf-in-python?utm_content=cmp-true
[Último acceso: octubre 2022].
Saunders, D., 2022. Domain adaptaiton and multi-domain adaptation for neural machine translation. *Artificial Inteligence,* Volumen 75, pp. 351-424.
SITE, N. P., s.f. *NATO STANDARIZATION OFFICE.* [En línea]
Available at: https://nso.nato.int/nso/nsdd/main/list-promulg
[Último acceso: Diciembre 2022].
Software, A., 2022. *PyMuPDF.* [En línea]
Available at: https://pymupdf.readthedocs.io/en/latest/
Soriano, M. A., 2019. *Estudio comparativo de traductores.* Madird: s.n.
Soriano, M. A., 2019. *Estudio comparativo de traductores.* Madrid: s.n.
Sorolla, P. V., 2018. *La evaluación de las herramientas de la Traducicón automática.* Soria: s.n.
Thompson, B. G. J. K. H. D. K. &. K. P., 2019. Overcoming catastrophic forgetting during domain adaptation of neural machine translation. *Conference of the North American Chapter of the Association for Computational Linguistics,* Volumen 1, pp. 2062-2068.
Van der Wees, M., 2017. *What's in a domain?: Towards fine-grained adaptation for machine translation.* Amsterdam: s.n.
W3School, 2022. *Python RegEx.* [En línea]
Available at: https://www.w3schools.com/python/python_regex.asp

7 ANEXOS

7.1 ANEXO I

La composición del anexo I data sobre las entrevistas realizadas a personal de las FAS destinados en el RI Canarias Nº50 ya que se precisa conocer y marcar los requisitos que la aplicación debe poseer para satisfacer las necesidades de los futuros usuarios.

Dejando a un lado los requisitos técnicos que un software debe cumplir, se ha realizado una encuesta informativa para conocer de primera mano las necesidades y requerimientos que el personal de las FAS solicita a este proyecto. La muestra ha sido seleccionada de la compañía de Mando y Apoyo del RI Canarias Nº50, considerada significativa y representativa del conjunto de las FAS por los motivos expuestos en el apartado 1.2. Esta encuesta cuenta con un total de 35 encuestados, siendo las preguntas de la misma las siguientes:

- Indicar la escala a la que pertenecen
- Indicar la especialidad fundamental a la que pertenecen
- Indicar si ha sido desplegado en zona de operaciones
- Indicar si han realizado ejercicios conjuntos con otros ejércitos
- Elegir qué idiomas se consideran más importantes para ser adquiridos, pudiendo elegir más de uno entre francés, inglés, alemán, ruso, árabe y chino.
- Indicar si han encontrado deficiencias al haber usado un traductor para la traducción de documentos de índole militar
- Indicar qué requisitos cree que debería tener este traductor, siendo esta pregunta de carácter abierto, permitiendo a los encuestados escribir lo que requieran.

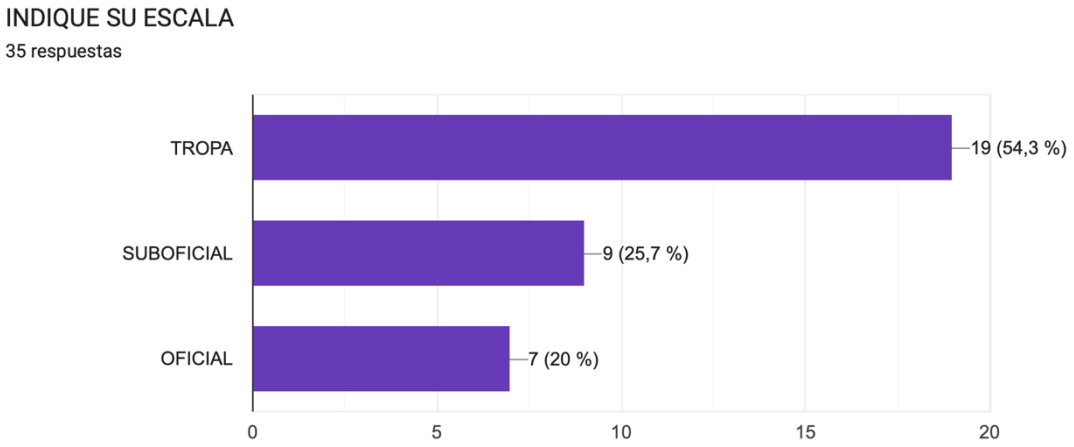

Ilustración 36- Resultado gráfico de pregunta realiazda en encuesta online a personal del RI Canarias Nº50. Fuente: elaboración propia a partir de google drive encuestas.

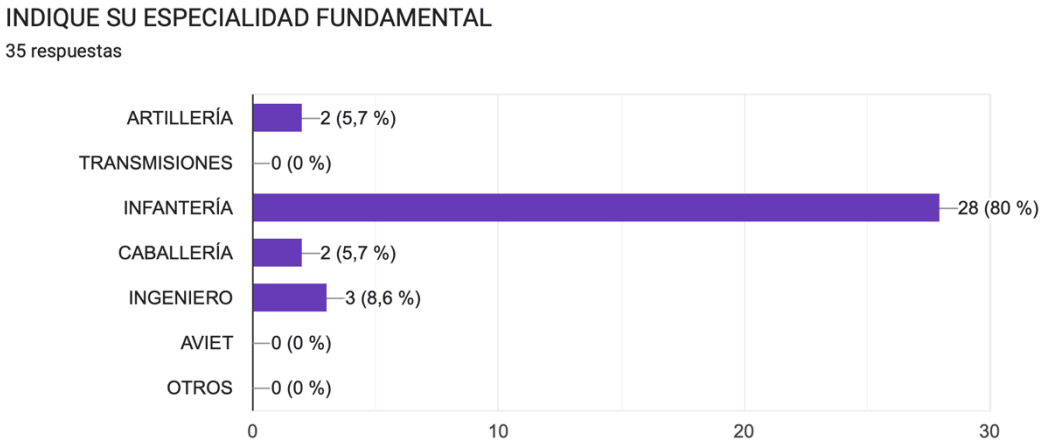

Ilustración 37- Resultado gráfico de pregunta realiazda en encuesta online a personal del RI Canarias Nº50. Fuente: elaboración propia a partir de google drive encuestas

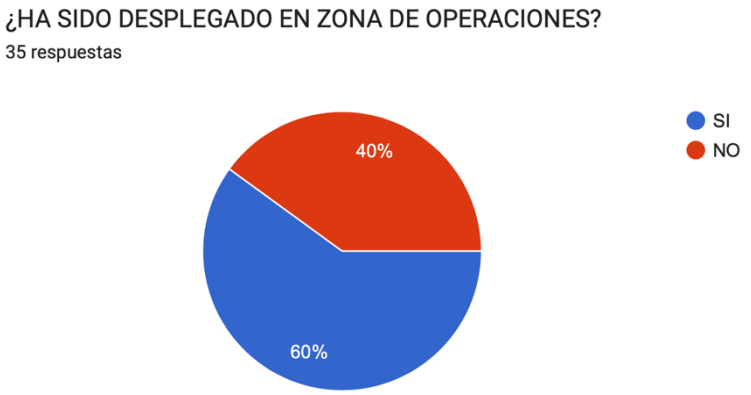

Ilustración 38- Resultado gráfico de pregunta realiazda en encuesta online a personal del RI Canarias Nº50. Fuente: elaboración propia a partir de google drive encuestas

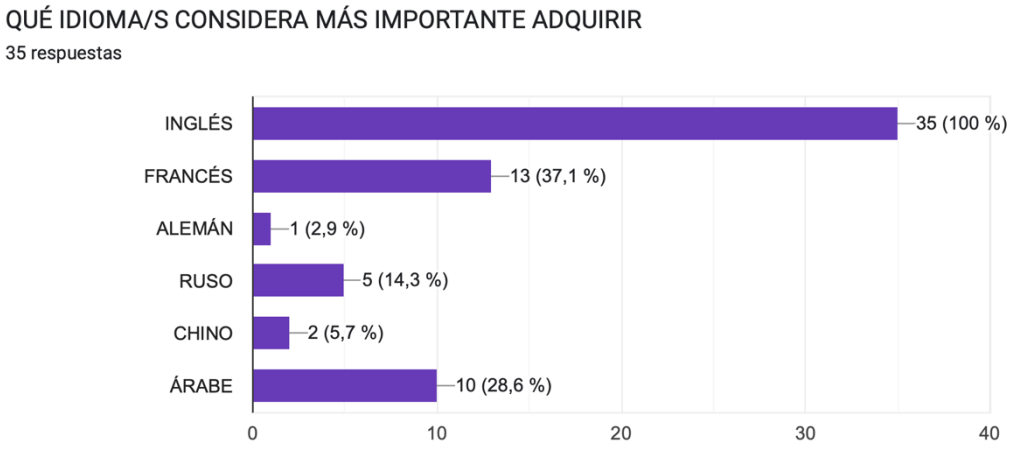

Ilustración 39- Resultado gráfico de pregunta realiazda en encuesta online a personal del RI Canarias Nº50. Fuente: elaboración propia a partir de google drive encuestas

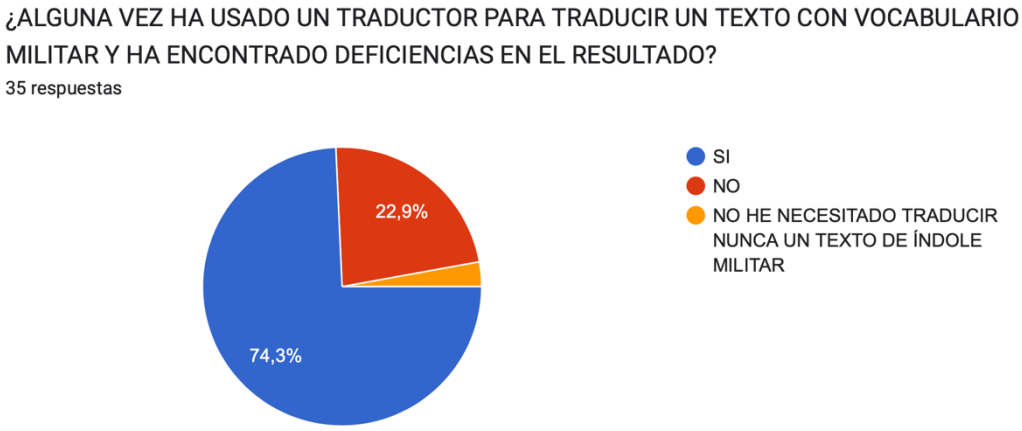

Ilustración 40- Resultado gráfico de pregunta realiazda en encuesta online a personal del RI Canarias Nº50. Fuente: elaboración propia a partir de google drive encuestas

En resumen, en la encuesta la compone un 20% de oficiales, un 25% de suboficiales y un 55% de personal de tropa. Dentro de este personal, un 80% pertenece a la especialidad fundamental de infantería, un 6% a la de artillería, un 6% a la de caballería y un 8% a la de ingenieros. El 60% del personal ha sido desplegado alguna vez en zona de operaciones y el 75% ha participado en algún ejercicio conjunto con otros ejércitos.

Dato muy curioso es que, de los posibles idiomas a elegir, el 100% ha elegido entre sus opciones el inglés y como otras opciones relevantes se obtiene que un 37% selecciona el francés, un 28% el árabe y un 14% el ruso, quedando el chino y alemán en última posición con un 6% y 3% respectivamente.

El 75% de los encuestados han encontrado algún problema o deficiencia en el resultado de la traducción al haber empleado algún traductor para textos con terminología militar, un 23% no han tenido este problema y un 2% nunca han tenido la necesidad de traducir un documento de índole militar.

Para la última pregunta “¿QUÉ REQUISITOS CREE QUE DEBERÍA TENER UN TRADUCTOR PARA QUE FUESE ÚTIL PARA DEFENSA?” se muestra una breve síntesis de los resultados obtenidos remarcando que 27 encuestados han escrito qué requisitos solicitan:

- 16 encuestados requieren que el traductor tenga un dominio específico de carácter militar (en vocabulario, frases, terminología, etc.).
- 4 encuestados requieren que cuente también con siglas OTAN.
- 4 encuestados requieren que pueda traducir también por voz
- En menos proporción (dos o menos encuestados) se han obtenido otros requisitos no significativos para el proyecto.

Con todos los datos obtenidos, se llega a la conclusión esperada: los usuarios piden la especificación de un traductor a terminología militar, por lo que se precisa realizar una recopilación de datos de índole militar (con traducción al inglés y francés como mínimo, ya que son los más requeridos) para que sirvan como apoyo para futuros proyectos.

7.2 ANEXO II

Este anexo muestra el código empleado para poder procesar el documento pdf de MILGLOSS. Ha sido extraído de la red (ilustración 42) y, posteriormente, modificado en parte por la Dra. Lacramioara Dranca, para poder adaptarlo a las necesidades particulares del trabajo (mostrado en la ilustración 43).

Sitio web: https://www.thepythoncode.com/article/extract-text-from-pdf-in-python

```
def extract_text(input_file, output_file, pages):
    ''' extract the text content of a pdf file
    :param input_file: pdf file to be processed
    :param output_file: txt file to write the result
    :param page: -1 to process all pages, or list with the pages to be processed, ex: [0,1,3]
    '''
    pdf          = fitz.open(input_file)
    if pages==-1:
        # if pages is not set, default is all pages of the input PDF document
        pages = list(range(pdf.page_count))

    output_files = output_file
    if output_file is not sys.stdout:
            # a single file, open it
            output_file = open(output_file, "w")
            output_files = { pn: output_file for pn in pages }
    else:
            # if output file is standard output, do not open
            output_files = { pn: output_file for pn in pages }
    # iterate over pages
    for pg in range(pdf.page_count):
        if pg in pages:
            # get the page object
            page = pdf[pg]
            # extract the text of that page and split by new lines '\n'
            page_lines = page.get_text().splitlines()
            # get the output file
            file = output_files[pg]
            # get the number of lines
            n_lines = len(page_lines)
            for line in page_lines:
                # remove any whitespaces in the end & beginning of the line
                #line = line.strip()
                # print the line to the file/stdout
                print(line, file=file)
            print(f"[*] Wrote {n_lines} lines in page {pg}")
    # close the files
    for pn, f in output_files.items():
        if f is not sys.stdout:
            f.close()
```

Ilustración 41- Muestra de código para el procesamiento del pdf MILGLOSS. Fuente: elaboración propia desde el ejecutable colab.

```
def extract_text_dict(input_file, output_file, pages):
    ''' extract the text content of a pdf file
    :param input_file: pdf file to be processed
    :param output_file: txt file to write the result
    :param page: -1 to process all pages, or list with the pages to be processed, ex: [0,1,3]
    '''
    pdf          = fitz.open(input_file)
    if pages==-1:
        # if pages is not set, default is all pages of the input PDF document
        pages = list(range(pdf.page_count))

    output_files = output_file
    if output_file is not sys.stdout:
            # a single file, open it
            output_file = open(output_file, "w")
            output_files = { pn: output_file for pn in pages }
    else:
            # if output file is standard output, do not open
            output_files = { pn: output_file for pn in pages }

    color_code = {}
    color_code[9914117]="<brown>"
    #color_code[8031]="<blue>"
    color_code[44880]="<green>"
    color = 0
    c = False
    font = "N"
    # iterate over pages
    for pg in range(pdf.page_count):
        if pg in pages:
            # get the page object
            page = pdf[pg]
            # get the dict structure of the page
            page_dict = page.get_text("dict",sort=True)
            # get the output file
            file = output_files[pg]
            #for each block in the page
            for bl in page_dict["blocks"]:
              line = ""
              #only for text text
              if "lines" in bl:
                for span in bl["lines"]:
                  #for each span in the line
                  for txt in span["spans"]:
                      #detect color has changed
                      if txt["color"] != c:
                        # if color is one of the monitored colors (see color_code definition)
                        if txt["color"] in color_code:
                          #concatenate color annotation
                          line = line + color_code[txt["color"]]
                        c = txt["color"]
                      #if text is a blank char
                      if txt["text"]==' ':
                        continue
                      #if text font contains italic or bold
                      elif "Bold" in txt["font"] and "Italic" in txt["font"]:
                        font_txt = "BI"
                      # if text font contains bold
                      elif "Bold" in txt["font"]:
                        font_txt = "B"
                      #if text font contains italic
                      elif "Italic" in txt["font"]:
                        font_txt = "I"
                      else:
                        font_txt = "N"
                      #detect font has changed
                      if font != font_txt:
                        #update current font
                        font = font_txt
                        #concatenate font annotation
                        line = line + "<"+font_txt+">"
                      line = line + txt["text"]
                # print line to output file
                print(line,file=file)
    # close the files
    for pn, f in output_files.items():
        if f is not sys.stdout:
            f.close()
```

Ilustración 42- Muestra de código para el procesamiento del pdf MILGLOSS. Fuente: elaboración propia desde el ejecutable colab.

```
#extract text from list "pag" and save text in a text file "text.txt"
pag = []
for i in range(18,126,1):
  pag.append(i)
print(pag)
extract_text_dict('MILGLOSS 21 FEB 12 (1).pdf','text.txt',pag)
```

Ilustración 43- Muestra de código para el procesamiento del pdf MILGLOSS. Fuente: elaboración propia desde el ejecutable colab.

7.3 ANEXO III

Tras el procesado del archivo pdf para poder trabajar cómodamente con expresiones regulares se deben encontrar los patrones que definen la aparición sistemática de los datos.

Tras encontrar ciertas dificultades en ellos, se genera un archivo txt editado y manipulado a molde de las necesidades para poder inferir en él las expresiones regulares que nos facilitaran el acceso a los datos requeridos. Aquí se muestra un fragmento de cómo es físicamente este archivo.

Ilustración 44- Muestra del archivo txt generado a partir del pdf MILGLOSS. Fuente: elaboración propia a partir del ejecutable colab.

7.4 ANEXO IV

Este anexo es una muestra de cómo es la matriz que se ha generado a partir del archivo de texto text. Se aprecia que las entradas aparecen extraídas en fila y la información de cada entrada por columnas. Esta matriz se corresponde con la información del archivo que aparece en la ilustración 45 (anexo III).

Ilustración 45- Muestra de la matriz de datos extraídos del archivo txt. Fuente: elaboración propia a partir del ejecutable colab.

7.5 ANEXO V

Por último, se adjunta como anexo final, el total del código que se ha empleado durante la elaboración del proyecto. Tanto para el procesamiento del pdf, la extracción de datos con expresiones regulares y la definición a un estándar Ontolex.

```
pip install PyMuPDF==1.20.2
```

```
!gdown '1LYCHgqGHWTb5rjWPo33LmTamFeC4kb_m'
```

```
import fitz
import argparse
import sys
import os
from pprint import pprint
```

```
def extract_text(input_file, output_file, pages):
    ''' extract the text content of a pdf file
    :param input_file: pdf file to be processed
    :param output_file: txt file to write the result
    :param page: -1 to process all pages, or list with the pages to
be processed, ex: [0,1,3]
    '''
    pdf          = fitz.open(input_file)
    if pages==-1:
        # if pages is not set, default is all pages of the input
PDF document
        pages = list(range(pdf.page_count))

    output_files = output_file
    if output_file is not sys.stdout:
            # a single file, open it
            output_file = open(output_file, "w")
            output_files = { pn: output_file for pn in pages }
    else:
            # if output file is standard output, do not open
            output_files = { pn: output_file for pn in pages }
    # iterate over pages
    for pg in range(pdf.page_count):
        if pg in pages:
            # get the page object
            page = pdf[pg]
            # extract the text of that page and split by new lines
'\n'
            page_lines = page.get_text().splitlines()
            # get the output file
            file = output_files[pg]
```

```
            # get the number of lines
            n_lines = len(page_lines)
            for line in page_lines:
                # remove any whitespaces in the end & beginning of
the line
                #line = line.strip()
                # print the line to the file/stdout
                print(line, file=file)
            print(f"[*] Wrote {n_lines} lines in page {pg}")
    # close the files
    for pn, f in output_files.items():
        if f is not sys.stdout:
            f.close()
```

```
def extract_text_dict(input_file, output_file, pages):
    ''' extract the text content of a pdf file
    :param input_file: pdf file to be processed
    :param output_file: txt file to write the result
    :param page: -1 to process all pages, or list with the pages to
be processed, ex: [0,1,3]
    '''
    pdf          = fitz.open(input_file)
    if pages==-1:
        # if pages is not set, default is all pages of the input
PDF document
        pages = list(range(pdf.page_count))

    output_files = output_file
    if output_file is not sys.stdout:
            # a single file, open it
            output_file = open(output_file, "w")
            output_files = { pn: output_file for pn in pages }
    else:
            # if output file is standard output, do not open
            output_files = { pn: output_file for pn in pages }

    color_code = {}
    color_code[9914117]="<brown>"
    #color_code[8031]="<blue>"
    color_code[44880]="<green>"
    color = 0
    c = False
    font = "N"
    # iterate over pages
    for pg in range(pdf.page_count):
        if pg in pages:
            # get the page object
            page = pdf[pg]
            # get the dict structure of the page
```

```
            page_dict = page.get_text("dict",sort=True)
            # get the output file
            file = output_files[pg]
            #for each block in the page
            for bl in page_dict["blocks"]:
              line = ""
              #only for text text
              if "lines" in bl:
                for span in bl["lines"]:
                  #for each span in the line
                  for txt in span["spans"]:
                      #detect color has changed
                      if txt["color"] != c:
                        # if color is one of the monitored colors
(see color_code definition)
                        if txt["color"] in color_code:
                          #concatenate color annotation
                          line = line + color_code[txt["color"]]
                        c = txt["color"]
                      #if text is a blank char
                      if txt["text"]==' ':
                        continue
                      #if text font contains italic or bold
                      elif "Bold" in txt["font"] and "Italic" in
txt["font"]:
                        font_txt = "BI"
                      # if text font contains bold
                      elif "Bold" in txt["font"]:
                        font_txt = "B"
                      #if text font contains italic
                      elif "Italic" in txt["font"]:
                        font_txt = "I"
                      else:
                        font_txt = "N"
                      #detect font has changed
                      if font != font_txt:
                        #update current font
                        font = font_txt
                        #concatenate font annotation
                        line = line + "<"+font_txt+">"
                      line = line + txt["text"]
                # print line to output file
                print(line,file=file)
    # close the files
    for pn, f in output_files.items():
        if f is not sys.stdout:
            f.close()
```

```
#extract text from list "pag" and save text in a text file
"text.txt"
pag = []
for i in range(18,126,1):
  pag.append(i)
print(pag)
extract_text_dict('MILGLOSS 21 FEB 12 (1).pdf','text.txt',pag)
```

```
import re

file = open('text.txt')
txt = file.read()

#uso de expresiones regulares para la extracción de los datos

x = re.findall('\n(.*?)(-.*)?
?<BI>(adj|pp|vt|n|vi)((?:.|\n)*?.*?)<I>((?:.|\n)*?.*?)<green>((?:.|
\n)*?.*?)\.(?:.*?<brown>((?:.|\n)*?)<I>.*\(Spñl\)((?:.|\n)*?)(?:\.|
\?))?',txt)
for i in range(0, len(x), 1):
  x[i] = list(x[i])

#función para realizar limpieza de los datos extraídos
  for j in range(0,len(x[i]),1):
    x[i][j] = re.sub("\n","",x[i][j])
    x[i][j] = re.sub("<N>","",x[i][j])
    x[i][j] = re.sub("\(a\)","",x[i][j])
    x[i][j] = re.sub("\(b\)","",x[i][j])
    x[i][j] = re.sub("<brown>","",x[i][j])
    x[i][j] = re.sub("(Spñl)","",x[i][j])
    x[i][j] = re.sub("<~>","",x[i][j])
    x[i][j] = re.sub("<I>","",x[i][j])
    x[i][j] = re.sub("<BI>","",x[i][j])
    x[i][j] = re.sub("\(Am Eng\)","",x[i][j])
    x[i][j] = re.sub("\(c\)","",x[i][j])
    x[i][j] = re.sub("\(Br Eng\)","",x[i][j])
    x[i][j] = re.sub("\[plural\]","",x[i][j])
    x[i][j] = re.sub("<green>","",x[i][j])
    x[i][j] = re.sub("87 ®MILGLOSS-AGM-DPTO","",x[i][j])
    x[i][j] = re.sub("<B>","",x[i][j])
    x[i][j] = re.sub("\.","",x[i][j])
    x[i][j] = re.sub("\~","",x[i][j])
    if j==1:
      x[i][j] = re.sub('- ','',x[i][j])

    x[i][j] =x[i][j].strip()
print(x[i])

   import os
```

```
f = open('lexEN.txt', 'w') #crear fichero ENGLISH
f.write("@prefix milgloss:
<https://ejercito.defensa.gob.es/unidades/Zaragoza/agm/> ." +
os.linesep)
f.write("@prefix dc: <http://purl.org/dc/elements/1.1/> ." +
os.linesep)
f.write("@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> ." +
os.linesep)
f.write("@prefix foaf: <http://xmlns.com/foaf/0.1/> ." +
os.linesep)
f.write("@prefix lexinfo:
<http://www.lexinfo.net/ontology/2.0/lexinfo#> ." + os.linesep)
f.write("@prefix xsd: <http://www.w3.org/2001/XMLSchema#> ." +
os.linesep)
f.write("@prefix owl: <http://www.w3.org/2002/07/owl#> ." +
os.linesep)
f.write("@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
." + os.linesep)
f.write("@prefix ontolex: <http://www.w3.org/ns/lemon/ontolex#> ."
+ os.linesep)
f.write("@prefix lime: <http://www.w3.org/ns/lemon/lime#> ." +
os.linesep)
f.write("@prefix :
<https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss
/lexiconEN/> ." + os.linesep)
f.write("")

for i in range(0,len(x),1):

f.write("<http://https://ejercito.defensa.gob.es/unidades/Zaragoza/
agm//id/milgloss/lexiconEN> lime:entry :"+x[i][0]+"-"+x[i][3]+"-en
." + os.linesep)
  f.write(":"+x[i][0]+"-"+x[i][3]+"-en a ontolex:LexicalEntry;" +
os.linesep)
  f.write(" lexinfo:morphosyntacticProperty milgloss:"+x[i][3]+";"
+ os.linesep)
  f.write("ontolex:lexicalForm :"+x[i][0]+"-"+x[i][3]+"-en-form;" +
os.linesep)
  f.write(" dc:source <https://github.com/milgloss/apertium-
trunk.git> . " + os.linesep)
  f.write(":"+x[i][0]+"-"+x[i][3]+"-en-form a ontolex:Form; " +
os.linesep)
  f.write('contolex:writtenRep "'+x[i][0]+'"@en .' + os.linesep)
  f.write(os.linesep)

f.close()
  #print(txt, file=f)
import os
```

```
f = open('lexES.txt', "w") #crear fichero ESPAÑOL
f.write("@prefix milgloss:
<https://ejercito.defensa.gob.es/unidades/Zaragoza/agm/> ." +
os.linesep)
f.write("@prefix dc: <http://purl.org/dc/elements/1.1/> ." +
os.linesep)
f.write("@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> ." +
os.linesep)
f.write("@prefix foaf: <http://xmlns.com/foaf/0.1/> ." +
os.linesep)
f.write("@prefix lexinfo:
<http://www.lexinfo.net/ontology/2.0/lexinfo#> ." + os.linesep)
f.write("@prefix xsd: <http://www.w3.org/2001/XMLSchema#> ." +
os.linesep)
f.write("@prefix owl: <http://www.w3.org/2002/07/owl#> ." +
os.linesep)
f.write("@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
." + os.linesep)
f.write("@prefix ontolex: <http://www.w3.org/ns/lemon/ontolex#> ."
+ os.linesep)
f.write("@prefix lime: <http://www.w3.org/ns/lemon/lime#> ." +
os.linesep)
f.write("@prefix :
<https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss
/lexiconES/> ." + os.linesep)
f.write("")

for i in range(0,len(x),1):


f.write("<http://https://ejercito.defensa.gob.es/unidades/Zaragoza/
agm//id/milgloss/lexiconEN> lime:entry :"+x[i][4]+"-"+x[i][3]+"-es
." + os.linesep)
  f.write(":"+x[i][4]+"-"+x[i][3]+"-es a ontolex:LexicalEntry;" +
os.linesep)
  f.write(" lexinfo:morphosyntacticProperty milgloss:"+x[i][4]+";"
+ os.linesep)
  f.write("ontolex:lexicalForm :"+x[i][4]+"-"+x[i][3]+"-es-form;" +
os.linesep)
  f.write(" dc:source <https://github.com/milgloss/apertium-
trunk.git> . " + os.linesep)
  f.write(":"+x[i][4]+"-"+x[i][3]+"-es-form a ontolex:Form; " +
os.linesep)
  f.write("contolex:writtenRep "+x[i][0]+"@es ." + os.linesep)
  f.write("" + os.linesep)

f.close()
#txt = f.read()
```

```
#print(txt, file=f)
import os

f = open('lexFR.txt', "w") #crear fichero FRANÇAIS
f.write("@prefix milgloss:
<https://ejercito.defensa.gob.es/unidades/Zaragoza/agm/> ." +
os.linesep)
f.write("@prefix dc: <http://purl.org/dc/elements/1.1/> ." +
os.linesep)
f.write("@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> ." +
os.linesep)
f.write("@prefix foaf: <http://xmlns.com/foaf/0.1/> ." +
os.linesep)
f.write("@prefix lexinfo:
<http://www.lexinfo.net/ontology/2.0/lexinfo#> ." + os.linesep)
f.write("@prefix xsd: <http://www.w3.org/2001/XMLSchema#> ." +
os.linesep)
f.write("@prefix owl: <http://www.w3.org/2002/07/owl#> ." +
os.linesep)
f.write("@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
." + os.linesep)
f.write("@prefix ontolex: <http://www.w3.org/ns/lemon/ontolex#> ."
+ os.linesep)
f.write("@prefix lime: <http://www.w3.org/ns/lemon/lime#> ." +
os.linesep)
f.write("@prefix :
<https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss
/lexiconFR/> ." + os.linesep)
f.write("")

for i in range(0,len(x),1):



f.write("<http://https://ejercito.defensa.gob.es/unidades/Zaragoza/
agm//id/milgloss/lexiconFR> lime:entry :"+x[i][5]+"-"+x[i][3]+"-fr
." + os.linesep)
  f.write(":"+x[i][5]+"-"+x[i][3]+"-fr a ontolex:LexicalEntry;" +
os.linesep)
  f.write(" lexinfo:morphosyntacticProperty milgloss:"+x[i][5]+";"
+ os.linesep)
  f.write("ontolex:lexicalForm :"+x[i][5]+"-"+x[i][3]+"-fr-form;" +
os.linesep)
  f.write(" dc:source <https://github.com/milgloss/apertium-
trunk.git> . " + os.linesep)
  f.write(":"+x[i][5]+"-"+x[i][3]+"-fr-form a ontolex:Form; " +
os.linesep)
  f.write('contolex:writtenRep "'+x[i][0]+'"@fr .' + os.linesep)
  f.write("" + os.linesep)
```

```
f.close()
#txt = f.read()
#print(txt, file=f)

import os

f = open('lexTRD_ES_EN.txt', "w") #crear fichero relación ES-EN
f.write("@prefix milgloss:
<https://ejercito.defensa.gob.es/unidades/Zaragoza/agm/> ." +
os.linesep)
f.write("@prefix dc: <http://purl.org/dc/elements/1.1/> ." +
os.linesep)
f.write("@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> ." +
os.linesep)
f.write("@prefix foaf: <http://xmlns.com/foaf/0.1/> ." +
os.linesep)
f.write("@prefix lexinfo:
<http://www.lexinfo.net/ontology/2.0/lexinfo#> ." + os.linesep)
f.write("@prefix xsd: <http://www.w3.org/2001/XMLSchema#> ." +
os.linesep)
f.write("@prefix owl: <http://www.w3.org/2002/07/owl#> ." +
os.linesep)
f.write("@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
." + os.linesep)
f.write("@prefix ontolex: <http://www.w3.org/ns/lemon/ontolex#> ."
+ os.linesep)
f.write("@prefix lime: <http://www.w3.org/ns/lemon/lime#> ." +
os.linesep)
f.write("@prefix vartrans: <http://www.w3.org/ns/lemon/vartrans#>
." + os.linesep)
f.write("@prefix :
<https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss
/tranSetES-EN/> ." + os.linesep)
f.write("")
f.write("@prefix src:
<https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/milgloss
/lexiconES/> . " + os.linesep)
f.write("")
f.write("@prefix tgt:
<https://ejercito.defensa.gob.es/unidades/Zaragoza/agm//id/apertium
/lexiconEN/> ." + os.linesep)
f.write("")

for i in range(0,len(x),1):



  #HASTA AQUÍ. DESDE AQUÍ SI:
```

```
f.write("<http://https://ejercito.defensa.gob.es/unidades/Zaragoza/
agm//id/milgloss/tranSetES-EN> " + os.linesep)
  f.write("vartrans:trans :"+x[i][4]+'_'+x[i][0]+'-'+x[i][3]+"es-
sense-"+x[i][0]+'_'+x[i][4]+'-'+x[i][3]+"-en-sense-trans ." +
os.linesep)
  f.write(":"+x[i][4]+'_'+x[i][0]+'-'+x[i][3]+"-es-sense-
"+x[i][0]+'_'+x[i][4]+'-'+x[i][3]+"-en-sense-trans a
vartrans:Translation;" + os.linesep)
  f.write("vartrans:source :"+x[i][4]+'_'+x[i][0]+'-'+x[i][3]+"-es-
sense;" + os.linesep)
  f.write("vartrans:target :"+x[i][0]+'_'+x[i][4]+'-'+x[i][3]+"-en-
sense ." + os.linesep)
  f.write(":"+x[i][4]+'_'+x[i][0]+'-'+x[i][3]+"es-sense a
ontolex:LexicalSense;" + os.linesep)
  f.write("ontolex:isSenseOf src:"+x[i][4]+"-"+x[i][3]+"-es ." +
os.linesep)
  f.write(":"+x[i][0]+'_'+x[i][4]+'-'+x[i][1]+"-en-sense a
ontolex:LexicalSense;" + os.linesep)
  f.write("ontolex:isSenseOf tgt:"+x[i][0]+'-'+x[i][3]+"-en ." +
os.linesep)
  f.write("" + os.linesep)
f.close()
#print(txt, file=f)
```