ET8: Código fuente y ejecutables.

Plan de impulso de las Tecnologías del Lenguaje

Àlex Bravo, Horacio Saggion y Pablo Accuosto

07 2018

GOBIERNO DE ESPAÑA
MINISTERIO DE ECONOMÍA Y EMPRESA
SECRETARÍA DE ESTADO PARA EL AVANCE DIGITAL
ontsi observatorio nacional de las telecomunicaciones y de la SI
red.es
Plan TL
Plan de Impulso de las Tecnologías del Lenguaje

Este estudio ha sido realizado dentro del ámbito del Plan de Impulso de las Tecnologías del Lenguaje con financiación de la Secretaría de Estado para el Avance Digital y Red.es, que no comparten necesariamente los contenidos expresados en el mismo. Dichos contenidos son responsabilidad exclusiva de sus autores.

ÍNDICE

1. INTRODUCCIÓN 4

2. CORPORA PROCESADO 5

3. IXA PIPES 6

4. SCISUMMSERVICES 7

4.1 Componente OpenMINTED 8

4.1 Imagen Docker 8

5. METAMAP 10

6. UMLS MAPPER 13

7. MODELOS PROBABILÍSTICOS DE N-GRAMAS 14

8. MODELOS DE WORD EMBEDDINGS 15

9. REFERENCIAS 16

10. GLOSARIO DE SIGLAS Y ACRÓNIMOS 17

Estudio de viabilidad de una versión en español del sistema
Entregable 8: Código fuente y ejecutables

1. INTRODUCCIÓN

En este entregable se van a listar todas las herramientas que se han utilizado durante el estudio de viabilidad de una versión en español del sistema UMLS, indicando donde se obtienen y cómo se han ejecutado. También, se comparten todas las implementaciones que se han utilizado para ejecutar las herramientas anteriores, procesar corpora o generar resultados adicionales (como por ejemplo los modelos de lenguaje basados en Word Embeddings). Además, se comparten todos los resultados que se han obtenido durante el estudio. La descripción detallada de todos las implementaciones realizadas y compartidas en este estudio, se incluirán como fichero LEEME en el mismo repositorio.

2. CORPORA PROCESADO

Para este estudio, se han examinado y evaluado varios corpus de dominio biomédico (ver Entregable 1). En el Entregable 3 y 4 se procesaron una serie de corpora de fácil acceso desde la web de MedSpen[1]. Para obtener la información necesaria de cada corpora, se implementaron una serie de scripts en Python para transformarlos de sus formatos originales a ficheros de texto. Estos scripts se han compartido mediante un repositorio de GitHub[2] y se describen a continuación:

- COPPA_processing.py: Este script extrae todos los resúmenes de las patentes relacionadas con el campo de la biomedicina. Dada la ruta donde se encuentran los documentos del corpus tanto en inglés (p.e. /home/user/Lote1/tei/Xml/en/) como en español (p.e. /home/user/Lote1/tei/Xml/es/), crea un fichero con el volcado del texto de todas las patentes relacionadas con el campo biomédico. El resultado de este procesamiento está compartido en un directorio de Google Drive campo biomédico[3].

- MedlinePlus_processing.py: Este script extrae todos los artículos que componen el corpus de MedlinePlus. Dada la ruta donde se encuentran los documentos del corpus tanto en inglés (p.e. /home/user/MedlinePlus-TEI-Sp-En/TEI_EN/) como en español (p.e. /home/user/MedlinePlus-TEI-Sp-En/TEI_ES/), crea una carpeta para cada tipo de artículo y extrae el texto de todos los artículos en ficheros separados. El resultado de este procesamiento está compartido en un directorio de Google Drive campo biomédico[4].

- Scielo_processing.py: Este script extrae todos los artículos en inglés y en español que componen el corpus de Scielo. Dada la ruta donde se encuentran los documentos del corpus,un fichero con el volcado del texto de todos los artículos del corpus. El resultado de este procesamiento está compartido en un directorio de Google Drive campo biomédico[5].

[1] *http://temu.bsc.es/mespen/*

[2] *https://github.com/abravo84/PlanImpulso*

[3] *https://drive.google.com/drive/u/0/folders/18SE4Hlyp9Q8sheTxwrOTFZtBQ5Z7Fser?ogsrc=32*

[4] *https://drive.google.com/drive/u/0/folders/1F2hIONk84HfcWwkfYfZTOb79fT4wa6nD?ogsrc=32*

[5] *https://drive.google.com/drive/u/0/folders/1cTTn2xp98ZpgoR-nuxl88JRZ_0jS8edU?ogsrc=32*

3. IXA PIPES

IXA Pipes [1] es una herramienta utilizada para el análisis lingüístico para varios idiomas, incluidos el inglés, español, euskera y gallego. IXA Pipes ha sido desarrollado por el Grupo IXA NLP[6] de la Universidad del País Vasco[7] (ver Entregable 3 y 4 para más información).

La ejecución de IXA pipes fue detalladamente descrita en el Entregable 3 y 4. En este entregable, se comparte los ficheros de Shell Script que se utilizaron para procesar el corpus de MedlinePlus en inglés (ixa_en.sh) y en español (ixa_es.sh) con IXA Pipes. Ambos están accesibles en el repositorio de GitHub[8]. En ellos se describen las ejecuciones y configuraciones que se utilizaron. Los resultados obtenidos fueron compartidos mediante una carpeta de Google Drive[9].

[6] *http://ixa.si.ehu.es/*

[7] *http://www.ehu.es/*

[8] *https://github.com/abravo84/PlanImpulso*

[9] *https://drive.google.com/drive/u/0/folders/1Kl5NTwWSYsEZu4f86x0maLFphx1ICpCd*

4. SCISUMMSERVICES

SciSummServices es una herramienta de PLN que permite analizar artículos científicos basado en dos componentes principales: el procesamiento del analizador de artículos Dr. Inventor (DRI) [10] y el enriquecimiento de información a las oraciones con representaciones vectoriales. Tanto la herramienta SciSummServices como sus principales componentes están basados en GATE, por lo tanto, la entrada de un artículo será transformada al formato GATE así como los resultados procesados. El formato de entrada de SciSummServices es un directorio conteniendo un conjunto de artículos en formato XML (aunque DRI está preparado para transformar PDFs en XML, se ha limitado esta funcionalidad). Este formato de artículos científicos se pueden encontrar fácilmente en muchos repositorios del campo biomédico, por ejemplo: PubMed Central[11], Europe PMC[12] o Plos ONE Medicine[13].

El código SciSummServices está disponible en un repositorio de GitHub[14], bajo una licencia Creative Commons Attribution 4.0. Más información en el fichero LICENSE.TXT del repositorio de GitHub[15]. Además, el código incluye tres librerias (https://github.com/abravo84/scisumservices/tree/master/lib) bajo una licencia Attribution-NonCommercial 4.0 International[16]: SUMMA[17], DRINVENTOR[18] y SOCRECOMPUTATION[19].

[10] *http://backingdata.org/dri/library/*

[11] *https://www.ncbi.nlm.nih.gov/pmc/*

[12] *https://europepmc.org/*

[13] *https://journals.plos.org/plosmedicine/*

[14] *https://github.com/abravo84/scisumservices*

[15] *https://github.com/abravo84/scisumservices/blob/master/LICENSE.TXT*

[16] *https://creativecommons.org/licenses/by-nc/4.0/legalcode*

[17] *https://github.com/abravo84/scisumservices/blob/master/lib/LICENSE_SUMMA_UPF.txt*

[18] *https://github.com/abravo84/scisumservices/blob/master/lib/LICENSE_DRINVENTOR_LIB400b.txt*

[19] *https://github.com/abravo84/scisumservices/blob/master/lib/LICENSE_SCORECOMPUTATION.txt*

SciSummServices se desarrolló para la plataforma OPENMINTED[20] y es accesible mediante la propia plataforma de OPENMINTED[21] y como imagen Docker registrada en Docker Hub[22]. Los resultados obtenidos con el procesamiento de SciSummServices están compartidos a través de un directorio de Google Drive[23]. A continuación se describe el acceso y funcionamiento de SciSummServices en sus diferentes plataformas.

4.1 Componente OpenMINTED

SciSumm Services está disponible en la plataforma de OpenMinted en la siguiente dirección: https://test.openminted.eu/landingPage/component/9503494d-c09e-4b9a-80b8-10945c38ef6d

Concretamente, el componente de SciSummServices fue registrado con la siguiente identificación dentro de la plataforma (OpenMINTED ID):

OMTD: 378df0f8-1cb7-a8e4-df3b-80c0ba47851d

Para utilizar el servicio con esta plataforma, se realiza mediante web, donde se puede crear un corpus de artículos y pasar como entrada al componente de SciSummServices. Un tutorial de como hacer funcionar esta herramienta mediante OpenMINTED ha sido descrito en el ANEXO V del Entregable 3 y 4.

4.1 Imagen Docker

SciSummServices también está registrado y empaquetado como una imagen Docker en su propio Hub. Esta versión contiene toda la información fuente y código necesaria para que pueda ser compilado en un fichero JAR (mediante Maven), y posteriormente ejecutado. Se puede acceder a la imagen Docker desde la siguiente dirección:

https://hub.docker.com/r/abravp/openminted_scisumservices:1.0.0

Para ejecutar este componente se utiliza el siguiente comando:

[20] http://openminted.eu/

[21] https://test.openminted.eu/landingPage/component/9503494d-c09e-4b9a-80b8-10945c38ef6d

[22] https://hub.docker.com/r/abravp/openminted_scisumservices/

[23] https://drive.google.com/drive/u/0/folders/1C63FApdWvUEXmGWDLo6C2Y8vRvqMITcS?ogsrc=32

```
docker run -v /var/data/plos_one:/input -ti -v /var/data/plos_one_output:/output abravp/openminted_scisumservices:1.0.0 process.sh --input /input --output /output
```

Donde `/var/data/plos_one` es la ruta del directorio que contiene los artículos y `/var/data/plos_one_output` es la ruta del directorio de salida donde se guardaran los artículos procesados.

5. METAMAP

Metamap es probablemente el anotador biomédico más conocido y utilizado para la extracción de terminología biomédica. Ha sido desarrollado por la National Library of Medicine (NLM) para detectar terminología biomédica en textos e identificarla con sus correspondientes conceptos en el UMLS Metathesaurus. Además, cada anotación incluye un score que refleja la relevancia que existe entre el término encontrado en el texto y el concepto biomédico.

Para utilizar MetaMap se tiene que tener activada una cuenta UTS[24] (UMLS Terminology Services). Hay diferentes versiones de Metamap dependiendo de cómo se va a utilizar. En nuestro caso, vamos a utilizar la API de JAVA[25] de Metamap para procesar documentos biomédicos.

La API de Java de MetaMap permite que los programas Java accedan al mapeo de MetaMap, es decir, actúan como un cliente. Por lo tanto, se ha de instalar la parte de la API de Metamap que actúa de servidor y que explicaremos a continuación. Para resumir, este fichero contiene el conjunto de clases Java que comprenden la API y el servidor de mapeo de MetaMap.

La API de Java de MetaMap tiene dos requerimientos principales:

- La versión 1.8 de Java Software Development Kit (SDK) o superior es necesaria para utilizar la API de Java. El Java Runtime Environment (JRE 1.8) o superior es el entorno mínimo requerido para ejecutar esta API.

- Se debe descargar la versión completa de MetaMap en la web de descarga MetaMap Full Download [26]e instalarse antes de instalar la distribución de la API de Java. Para este proyecto se ha descargado la versión MetaMap 2016v2 Linux Version (64-bit - Bzip2 Tar - 2.69 GB) [27].

Una vez descargado y extraído el fichero de Metamap y con el Java requerido instalado, se puede instalar la API de Java de MetaMap. Para ello descargamos el fichero comprimido conteniendo la API.

[24] *https://uts.nlm.nih.gov/*

[25] *https://metamap.nlm.nih.gov/JavaApi.shtml*

[26] *https://metamap.nlm.nih.gov/MainDownload.shtml*

[27] *https://metamap.nlm.nih.gov/download/public_mm_linux_main_2016v2.tar.bz2*

En este proyecto se ha descargado la versión MetaMap Java API Release for Linux (Bzip2 Tar - 8 MB) [28].

Una vez descargado, se extrae dentro del mismo directorio donde hemos extraído anteriormente nuestra versión completa de MetaMap (usualmente el directorio se llama "public_mm"). Para más información ver la página web dedicada a la instalación y funcionamiento de la API[29].

Una vez descargado y extraído, dentro del directorio de Metamap "public_mm" hay una carpeta bin, dentro la cual hay varios ejecutables para realizar la instalación del servidor de Metamap. En nuestro caso, bajo Linux, utilizaremos el comando siguiente para la instalación:

$ ./bin/install.sh

Durante la instalación se preguntará sobre la localización de dos directorios. El primero es la ubicación donde hemos extraído nuestra copia completa de Metamap (carpeta llamada "public_mm"). El segundo directorio es la localización donde el Java Runtime Environment (JRE) está instalado. Una vez realizado esto, Metamap está listo para iniciarse como un servidor.

Para utilizar el servicio de Metamap (mmserverXX), antes hay que ejecutar una serie de servidores de apoyo. Primero, si el SKR/Medpost Tagger no se está ejecutando, hay que iniciar su servicio mediante el comando:

$ ./bin/skrmedpostctl start

Si se desea el servicio de Metamap para la desambiguación de palabras (WSD), se tendrá que iniciar también este servicio mediante el comando:

$ ./bin/wsdserverctl start

Una vez se han iniciado los servidores de apoyo, ya se puede iniciar el servidor principal de Metamap que se comunicará con la API de Java con el siguiente comando:

$ ./bin/mmserver{two-digit-year}

[28] https://metamap.nlm.nih.gov/download/public_mm_linux_javaapi_2016v2.tar.bz2

[29] https://metamap.nlm.nih.gov/Docs/README_javaapi.shtml

Para comprobar que todo es correcto, se puede realizar una pequeña prueba de ejecución. Usando otro terminal, puede verificar que api se está ejecutando usando el script testapi.sh, el cual toma una consulta como un argumento (el resultado debe de ser parecido al expuesto en la web relacionada[30]):

$ ./bin/testapi.sh cultura de laboratorio

Una vez se ha comprobado que el servidor de Metamap está funcionando, se puede implementar un programa en Java para procesar un conjunto de datos. En este entregable se comparte la clase de Java que utilizamos como cliente para conectar al servidor. Esta clase de Java, se ha compilado en un fichero JAR que también se comparte. Este fichero le entra como parámetros un directorio conteniendo documentos de texto, un directorio para que MetaMap escriba los resultados y todos los parámetros de configuración. Para ayudar a su ejecución, se ha implementado un Shell Script que ejecuta el fichero JAR con diferentes corpus y configuraciones. Todos los ficheros que contienen código, se han compartido en el repositorio de GitHub[31]. Además, los resultados obtenidos de procesar los corpus de COPPA, MedlinePlus y el subconjunto de Scielo han sido compartidos en un directorio de Google Drive[32].

[30] *https://metamap.nlm.nih.gov/Docs/README_javaapi.shtml#Testing%20the%20API*

[31] *https://github.com/abravo84/PlanImpulso*

[32] *https://drive.google.com/drive/u/0/folders/1aiCstxALXAO_X9pxe8Jw7qEGcv6QqpUb?ogsrc=32*

6. UMLS MAPPER

UMLS Mapper es una herramienta que identifica términos médicos en texto libre en español realizado por el grupo Vicomtech[33]. Además mapea esta terminología encontrada a conceptos UMLS. El mapeo se basa en técnicas de recuperación de información basado en la indexación de estas terminologías con Apache Lucene. Se usaron acrónimos y abreviaturas [2] y el IXA Pipes [1] para procesar los textos de entrada. La desambiguación de términos (WSD) con más de un mapeo posible se basa en la herramienta UKB [3]. Teniendo en cuenta el trabajo de la tarea 4, esta herramienta se basa en el procesamiento del análisis lingüístico de IXA Pipes (proceso incluido en la Tarea 3) para la identificación y extracción de terminología UMLS.

El acceso a esta herramienta no es libre, por ese motivo, nos hemos puesto en contacto con el grupo Vicomtech para tener acceso a su herramienta para nuestro estudio. posteriormente, se nos concedió un acceso a esta herramienta de manera temporal y restringida (solo para español) mediante Web Service. Aún así, se han podido identificar las opciones de configuración de la herramienta que se detallan en el ANEXO III del Entregable 3 y 4.

El UMLS Mapper fue utilizado para identificar conceptos UMLS en el mismo set de documentos utilizado en MetaMap pero en la lengua española: COPPA, MedlinePlus y el subconjunto de Scielo. Los resultados han sido compartidos en un directorio de Google Drive[34].

[33] *http://www.vicomtech.org/*

[34] *https://drive.google.com/drive/u/0/folders/1QunFYI1BFqai5cY8QFAC7QZL7v0aE-50?ogsrc=32*

7. MODELOS PROBABILÍSTICOS DE N-GRAMAS

En el Entregable 5 se implementó un script en Python para crear un modelo probabilístico basado en trigramas. Este modelo captura la probabilidad de una palabra basada en sus dos anteriores. En otras palabras, la probabilidad de la palabra Wi dadas las palabras Wi-1 y Wi-2 se define como P(Wi | Wi-1 , Wi-2).

Para probar el potencial de un modelo probabilístico basado en trigramas, se ha implementado un script de Python (utilizando la librería NLTK[35]) para calcular las probabilidades de cada palabra teniendo en cuenta las dos anteriores. Con este script se han creado dos modelos probabilísticos basados en trigramas:

- Primero, se han seleccionado todas las frases del corpus de MedlinePlus en inglés para poder modelarlas como se ha comentado anteriormente. Para mostrar el potencial del modelo, se han generado 50 frases aleatorias, basadas en las probabilidades aprendidas.
- El segundo experimento, ha sido realizado utilizando toda la terminología UMLS en inglés y se han generado 70 términos aleatorios.

El script de Python utilizado para implementar el modelo probabilístico de trigramas, se ha compartido en el repositorio de GitHub[36].

[35] *https://www.nltk.org/*

[36] *https://github.com/abravo84/PlanImpulso*

8. MODELOS DE WORD EMBEDDINGS

Para este entregable se va a utilizar una versión mejorada del modelo popular de Word2Vec llamada FastText [4]. La herramienta de FastText está disponible en su repositorio de GitHub[37]. Para una mejor documentación sobre la herramienta se puede consultar su página web principal[38].

Para la modelización del lenguaje médico usando FastText se han tenido en cuenta diferentes corpus que son tomados como base del aprendizaje.

Para modelar el vocabulario biomédico en español, se han seleccionado todas las frases de los siguientes corpora: Scielo, MedlinePlus, COPPA y IBECS. Y para modelar el vocabulario biomédico en inglés, se utilizó los documentos en inglés de los corpora Scielo, MedlinePlus y COPPA y además se seleccionaron unos 55.000 resúmenes actuales de PubMed como un nuevo corpus.

De cada corpus, se han extraído todas las frases y se han compilado en un fichero. A partir de cada conjunto de frases se ha ejecutado FastText para generar los embeddings para todo el vocabulario de cada corpus (compartidos en Google Drive[39]). Concretamente se han creado dos tipos de embeddings para cada corpus: uno basado en palabras y otro en lemmas.

Los embeddings se han generado en dos ficheros diferentes: un fichero binario (con extensión .bin) y un fichero de texto (con extensión .ext). La ejecución de FasText ha sido llevada a cabo con la siguiente configuración (en el ANEXO III del Entregable 5, se puede ver todos los parámetros de configuración).

> fasttext skipgram -ws 3 -epoch 20 -minCount 3 -neg 10 -input input_sentences.txt -output output_filename -dim 300

Los embeddings generados en este estudio han sido compartidos en un directorio de Google Drive[40].

[37] *https://github.com/facebookresearch/fastText*

[38] *https://fasttext.cc*

[39] *https://drive.google.com/drive/u/0/folders/1ys_kTaREVSg1_Y8oV688Jy0aRGK_g-qT?ogsrc=32*

[40] *https://drive.google.com/drive/u/0/folders/1ys_kTaREVSg1_Y8oV688Jy0aRGK_g-qT?ogsrc=32*

9. REFERENCIAS

[1] Agerri, R., Bermudez, J., & Rigau, G. (2014, May). IXA pipeline: Efficient and Ready to Use Multilingual NLP tools. In LREC (Vol. 2014, pp. 3823-3828).

[2] Montoya, I. (2017). Análisis, normalización, enriquecimiento y codificación de historia clínica electrónica (HCE). Tesis del Máster Universitario Konputazio Ingeniaritza eta Sistema Adimentsuak, (UPV/EHU).

[3] Agirre, E., & Soroa, A. (2009, March). Personalizing pagerank for word sense disambiguation. In Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (pp. 33-41). Association for Computational Linguistics.

[4] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606.

10. GLOSARIO DE SIGLAS Y ACRÓNIMOS

ACTH	Hormona adrenocorticotropa
AEMPS	Agencia Española de Medicamentos y Productos Sanitarios
AUI	Atom Unique Identifiers
BARR	Biomedical Abbreviation Recognition and Resolution
BNCS	Biblioteca Nacional de Ciencias de la Salud
CAP	College of American Pathologists
CBOW	Continuous Bag-of-Words
ChEBI	Chemical Entities of Biomedical Interest
CIMA	Centro de información online de medicamentos de la AEMPS
CMS	Centers for Medicare and Medicaid Service
CoNLL	The SIGNLL Conference on Computational Natural Language Learning
CPCSPA	ICPC Spanish
CPTSP	Current Procedural Terminology Spanish
CRF	conditional random field
CTV3	Clinical Terms Version 3
CUI	Concept Unique Identifiers
CUIDEN	Base de Datos Bibliográfica de la Fundación Index
DRI	Dr. Inventor
EHR	Electronic health record
EMEA	European Medicines Agency
ENFISPO	Biblioteca de la Facultad de Enfermería, Fisioterapia y Podología de la Universidad Complutense de Madrid

FECYT	Fundación Española para la Ciencia y la Tecnología
HUGO	Human Genome Organisation
IBECS	Índice Bibliográfico Español de Ciencias de la Salud
ICD	International Classification of Diseases
ICPC	International Classification of Primary Care
ICPCBAQ	ICPC Basque
IHTSDO	International Health Terminology Standards Development Organisation
IME	Índice Médico Español
IPC	International Patent Classification
ISCIII	Instituto de Salud Carlos III
ISSN	International Standard Serial Number
IULA	Institut de Lingüística Aplicada
JSON	JavaScript Object Notation
LNC-ES-AR	LOINC Linguistic Variant - Spanish, Argentina
LNC-ES-CH	LOINC Linguistic Variant - Spanish, Switzerland
LNC-ES-ES	LOINC Linguistic Variant - Spanish, Spain
LREC	Language Resources and Evaluation Conference
LUI	Lexical (term) Unique Identifiers
MDRSPA	MedDRA Spanish
MedDRA	Medical Dictionary for Regulatory Activities
MEDES	MEDicina en ESpañol
MeSH	Medical Subject Headings
MSHSPA	MeSH Spanish

MSSSI	Ministerio de Sanidad, Servicios Sociales e Igualdad
NCBI	National Center for Biotechnology Information
NCBO	National Center for Biomedical Ontology
NHS	National Health Service
NLM	National Library of Medicine
NLTK	Natural Language Toolkit
PLN	Procesamiento del Lenguaje Natural
POS	Part-of-speech
SciELO	Scientific Electronic Library Online
SCN	Nombre Científico
SCTSPA	SNOMED CT Spanish Edition
Snomed RT	Snomed Reference Terminology
SNS	Sistema Nacional de Salud
SUI	String Unique Identifiers
TUI	Type Unique Identifier
UIMA	Unstructured Information Management Architecture
UMLS	Unified Medical Language System
UTS	Servicios de Terminología UMLS
WHOART	WHO Adverse Drug Reaction
WHOSPA	WHOART Spanish
WSD	Word Sense Disambiguation
XML	Extensible Markup Language