Examen para la evaluación de la asignatura Procesamiento del Lenguaje Natural

Máster en Lógica, Ciencias de la Computación e Inteligencia Artificial

Convocatoria Junio 2022

Consideraciones generales

El sistema de evaluación de la asignatura PLN utilizará los siguientes criterios de evaluación.

La puntuación en la asignatura se basará en los siguientes tres elementos:

1. **Presentación de un tópico de profundización (Valoración final: 2 puntos sobre 10)**
    - Esta tarea puede ser realizada en grupos de 1 a 4 estudiantes.
    - Para la preparación del tópico de profundización se ofrecen dos alternativas:
        - Alternativa 1: Selección de uno de los tópicos de profundización descritos en la sección C de contenidos. El trabajo consistirá en la elaboración de una presentación de dicho tópico.
        - Alternativa 2: Selección por parte del alumno(s) de un tópico, que deberá ser propuesto al profesor y consensuado.
    - Teniendo en cuenta que uno de los objetivos del Máster es la preparación a la investigación, es relevante que en la preparación de esta presentación, se aborden tanto los fundamentos del tópico como las líneas más actuales de investigación en torno al mismo.
    - Para la elaboración de esta tarea, se deberá preparar un documento con el resumen del trabajo realizado, y se deberá hacer una presentación del mismo en clase (via de forma presencial u online) según las circunstancias sanitarias y las posibilidades de cada grupo de alumnos. Puesto que se pueden formar grupos, tanto la extensión como el tiempo de presentación se adecuarán al número de alumnos, aconsejándose un resumen del trabajo de entre 1 o 2 caras de folio por alumno y una duración de la presentación de entre 10 a 15 minutos por alumno.
    - La presentación de los tópicos se debe realizar durante el período lectivo de la asignatura, así que se deberá consensuar con el profesor el día y hora de presentación.

2. **Ejercicio teórico-práctico (Valoración final: 5 puntos sobre 10)**
    - Este examen constará de un conjunto de ejercicios teóricos y prácticos correspondientes a los contenidos centrales de la asignatura (Bibliografía básica de los Temas 1 a 10, del Bloque B Puntos clave del Procesamiento del Lenguaje Natural).
    - Es decir, se propondrán una serie de ejercicios prácticos basados en dichos contenidos así como otros de carácter práctico. También se propondrán otros ejercicios que supongan aplicar de forma simultánea contenidos y técnicas correspondientes a la bibliografía básica del temario.
    - Para la realización de este examen los alumnos dispondrán de un período de 10 días mínimo.
    - Para la convocatoria de Julio de 2022, el enunciado de este ejercicio se entregará el día 1 de junio, y el período para la realización y entrega del ejercicio se extenderá hasta el día 20 de junio (incluido).

3. **Examen (Valoración final: 3 puntos sobre 10)**
    - Se realizará un examen (inicialmente previsto en formato presencial), de 2 horas de duración.
    - El contenido del examen será la bibliografía básica (obligatoria) de los temas 1 a 10 del Bloque B de la Asignatura. El examen estará formado por 4 preguntas de las que cada alumno deberá seleccionar 3. Serán preguntas de redacción, orientadas a reflejar el conocimiento de las ideas principales del temario.
    - Este examen se realizará el día 27 de junio, en horario de 18:30 a 20:30, según el calendario de evaluación de la Escuela de Informática.

Ejercicio teórico-práctico

Declaración de integridad académica

Para la entrega del ejercicio completo se debe preparar un documento (PDF) siguiendo las siguientes directrices:

Página 1: Identificación y declaración de integridad académica

Ejercicio: Como se indica en la especificación del ejercicio práctico, hasta un máximo de 7 páginas (caras) correspondientes a cada uno de los apartados del ejercicio.

El examen se realiza en modo no presencial. Para realizar el ejercicio se puede utilizar cualquier material disponible en Internet. Podéis leer y utilizar los materiales del curso, leer documentación y proyectos desarrollados, etc. Lo único que no es académicamente aceptable es que una u más personas, bien de forma directa o indirecta, realice las tareas correspondientes a este examen. Lo que debéis entregar como solución al examen deberá ser desarrollo vuestro, y cuando utilicéis material disponible en Internet o leído en distintos materiales bibliográficos, debéis indicar claramente la fuente. Por ejemplo, si os habéis inspirado en un determinado algoritmo debéis indicar la fuente correspondiente (libro, web, etc.)

La primera página, de identificación deberá contener los siguientes datos:

- Nombre y Apellidos
- DNI o equivalente
- Email de contacto
- Asignatura: PLN – MULCIA – Convocatoria Junio 2022

Y a continuación el texto con la declaración de identidad. Se asume que la entrega del ejercicio supone la aceptación de este código ético de integridad académica. Es decir debéis copiar e incluir en dicha primera página de identificación el texto que aparece a continuación:

Afirmo que no he dado ni recibido ninguna ayuda no autorizada en este examen y que todo el trabajo ha sido mío. Cuando he utilizado un recurso no creado directamente por mí, he citado la fuente, y he descrito cómo la he utilizado.

He completado este examen de manera justa, honesta, respetuosa, responsable y confiable. Esto significa que he completado el examen como si el profesor estuviera observando todas mis acciones. He actuado de acuerdo con las instrucciones del profesor, y no he dado ni he recibido ninguna ayuda o asistencia que no sea la autorizada. Sé que la integridad de este examen y esta clase depende de mí, y me he comprometido a no tomar ninguna medida que rompa la

confianza de mis compañeros de clase o profesor, o socave la equidad de esta clase.

Certifico que la escritura contenida en este documento es mía y que cualquier cita directa ha sido identificada y citada. Además, he citado referencias en cualquier lugar donde he tomado prestadas las ideas de otra persona.

Ejercicio práctico: Predicción de puntuación

Introducción al ejercicio

Determinadas tareas del Procesamiento del Lenguaje Natural generan cadenas de palabras sin introducir signos de puntuación. Por ejemplo, un sistema de reconocimiento de voz recibe como entrada la señal acústica y genera normalmente una lista de tokens (palabras) aplicando distintos algoritmos. En la señal acústica no incluimos los signos de puntuación sino que nos valemos de pausas, la entonación de la prosodia, etc. En otro contexto, un sistema de traducción automática recibe como entrada una frase o expresión en un lenguaje y debe generar la transcripción correspondiente a la traducción. Aunque la expresión de entrada incluya signos de puntuación, la expresión de salida puede cambiar la ubicación de los signos de puntuación. Por ejemplo, entre castellano e inglés hay diferencias obvias provocadas por el uso en castellano del signo de inicio de pregunta (¿) que no se usa en inglés. Incluso la reestructuración de los elementos de la frase traducida puede necesitar una reorganización de los signos de puntuación.

Este trabajo no aborda los problemas de reconocimiento de voz o de traducción automática, sino la tarea denominada "Predicción de Puntuación". En concreto, nuestro objetivo en esta tarea será definir modelos que puedan introducir puntuación en una entrada que carece de puntuación. Para observar el impacto de esta tarea, consideremos a alguien que está dando una conferencia y termina su charla despidiéndose y preguntando si hay alguna pregunta entre la audiencia. La transcripción, con puntuación, podría ser como sigue:

> *Muchas gracias. ¿Alguna pregunta? Si no hay ninguna duda, con esto terminamos. Bien. Veo que alguien levanta la mano. Sí, por favor, ¿cuál es tu pregunta?*

Observemos qué ocurre si eliminamos mayúsculas y signos de puntuación. Intentad pronunciarlo leyendo sin más las palabras una tras otra:

> *muchas gracias alguna pregunta si no hay ninguna duda con esto terminamos bien veo que alguien levanta la mano sí por favor cuál es tu pregunta*

Incluso el uso de puntuación puede tener consecuencias muy importantes, como la propia vida de una persona. José Antonio Millán, en su libro "*Perdón imposible*" cuenta la anécdota según la cual el emperador Carlos V al recibir una condena de muerte de un preso dijo … empecemos con la forma básica sin puntuación que hubiese podido generar una transcripción básica de un reconocedor de voz:

> *perdón imposible que cumpla la condena*

Y ahora observemos dos posibles puntuaciones:

> *perdón, imposible que cumpla la condena*

o bien

perdón imposible, que cumpla la condena

Una vez introducido el problema que queremos abordar, se indica a continuación la especificación de la tarea.

Corpus (datasets)

Para desarrollar este ejercicio se van a utilizar unos corpus, disponibles junto con las instrucciones de este ejercicio.

En concreto, partimos de un dataset generado a partir de las transcripciones de presentaciones en la serie “TED talks”.

Los ficheros correspondientes a este corpus se pueden descargar desde la siguiente dirección

https://consigna.us.es/612679

Clave: M2tQ8$R4bb

Una vez descomprimido el fichero zip, se obtendrán los tres ficheros correspondientes a los corpus que se explican a continuación.

Corpus de entrenamiento: PuntuationTask.train.en

Este fichero contiene transcripciones de charlas de TED, procesadas y con los signos de puntuación incluidas. Por ejemplo, las primeras líneas de este fichero son:

And it can be a very complicated thing, what human health is.
And bringing those two together might seem a very daunting task, but what I'm going to try to say is that even in that complexity, there's some simple themes that I think, if we understand, we can really move forward.
And those simple themes aren't really themes about the complex science of what's going on, but things that we all pretty well know.
And I'm going to start with this one: If momma ain't happy, ain't nobody happy.
We know that, right? We've experienced that.

Corpus de test: PuntuationTask.test.en

Este fichero contiene también transcripciones de charlas de TED, pero no incluye signos de mayúscula (todo son minúsculas), y no contiene los siguientes signos de puntuación:

Punto: .

Coma: ,
Punto y coma: ;
Dos puntos: :
Interrogación: ? (al ser textos en inglés solo usaremos el signo de cierre de interrogación)
Exclamación: !

Se han mantenido en este corpus signos como la comilla simple en palabras en inglés como I'm o there's, así como el signo – en multipalabras (como great-grandchild) o para marcar pausas con –

Por ejemplo, las primeras líneas de este fichero contienen lo siguiente:

it can be a very complicated thing the ocean
and we're making the ocean pretty unhappy in a lot of different ways
people working in these canneries could barely stay there all day because of the smell but you know what they came out saying
we made the ocean unhappy we made people very unhappy and we made them unhealthy
we see the base of the food chain the plankton the small things and we see how those animals are food to animals in the middle of the pyramid and on so up this diagram

Corpus de verificación: PuntuationTask.check.en

Este fichero contiene la versión "correcta" del fichero de test anterior. El objetivo es que podamos comparar el resultado de nuestros algoritmos de puntuación con la versión correcta para evaluar la calidad de los mismos.

Por ejemplo, las mismas frases anteriores aparecen como:

It can be a very complicated thing, the ocean.
And we're making the ocean pretty unhappy in a lot of different ways.
People working in these canneries could barely stay there all day because of the smell, but you know what they came out saying?
We made the ocean unhappy; we made people very unhappy, and we made them unhealthy.
We see the base of the food chain, the plankton, the small things, and we see how those animals are food to animals in the middle of the pyramid, and on so up this diagram.

Estructura del ejercicio

Apartado 1:

Se debe implementar un sistema que permita recibir una expresión como entrada (será una expresión formada solo por minúsculas y sin los signos de puntuación mencionados (los seis signos con los que vamos a trabajar en este ejercicio y que serán siempre, el punto, la coma, los dos puntos, el punto y coma, el signo de cierre de interrogación y de exclamación: `.,;:?!`), y la salida será la misma expresión pero con los cambios correspondientes a la introducción de mayúsculas y signos de puntuación indicados.

A un nivel alto de especificación podremos considerar que este método tiene esta signatura:

```
string addPunctuationBasic(string)
```

Es decir recibirá como entrada un string y devolverá como salida un string.

Como primera versión de esta función addPunctuationBasic se implementará un modelo que simplemente cambia la primera letra por mayúscula y añade al final del string de entrada un punto.

Por ejemplo, al ejecutar

addPunctuationBasic("*it can be a very complicated thing the ocean")*

se obtendrá

"It can be a very complicated thing the ocean."

Apartado 2:

Implementar la función verifyPunctuation con la siguiente signatura

[(pos,err)] verifyPunctuation(string check, string test)

Para realizar esta operación se llevará a cabo una tokenización de ambos strings. En este caso se considerarán tokens todas las secuencias de letras, números y cualquier otro signo que no sea uno de los seis indicados como signos de puntuación en este ejercicio (`.,;:?!`)

Es decir, esta función devolverá una lista de pares, donde cada par contendrá la posición (indicada como índice en el string check) y el tipo de error. Los errores posibles son

- 'I' → Insertion
- 'D' → Deletion
- 'S' → Substitution

Por ejemplo, consideremos que el string de referencia correcto (check) es

“**Hello. What’s your name?**”

La tokenización generará los siguientes 6 tokens de referencia

```
Token 0: Hello
Token 1: .
Token 2: What’s
Token 3: your
Token 4: name
Token 5: ?
```

Consideremos que nuestro algoritmo de puntuación (un caso hipotético para analizar el algoritmo de verificación) genera la siguiente salida

“Hello what’s your, name?”

Es decir los tokens generados en este caso son:

```
Token 0: Hello
Token 1: what’s
Token 2: your
Token 3: ,
Token 4: name
Token 5: ?
```

Debemos devolver la lista de cambios necesarios para convertir la cadena de tokens generados (hipótesis) en la cadena de tokens correctos. Para ello, podemos inspirarnos en el algoritmo de la distancia de Levenshtein. Es importante tener en cuenta que dadas dos cadenas A y B:

Dist(A,B) == Dist(B,A)

Por lo que podemos abordar el problema tanto desde el punto de los cambios que hay que hacer para llegar desde la hipótesis hasta el modelo correcto, o bien desde el modelo correco (referencia o check en la terminología de este ejercicio) hasta la hipótesis generada por nuestro algoritmo de puntuación.

Podemos ver que respecto a nuestro string de referencia (check), el string de test ha ignorado (podemos decir que borrado respecto al de referencia un ., por tanto tendríamos el error

```
(‘D’,1)
```

En segundo lugar, el token 2 del string de referencia (check) se ha quedado mal en el string de test ya que en lugar de “What’s” aparece “what’s”. Se trata por tanto de un error de substitución de una palabra por otra (en este caso por un error de introducción de mayúsculas):

```
(‘S’,2)
```

Y en tercer lugar, entre los tokens ‘your’ y ‘name’ del string correcto (check) se ha introducido un nuevo token en el string de test, una coma en concreto, por tanto hay un error que calificaríamos como

```
(‘I’,4)
```

Como se puede observar, todos los errores se posicionan (usan los índices) respecto al string correcto de referencia (check).

Así pues el resultado de

```
verifyPunctuation("Hello. What's your name?",
                  "Hello what's your, name?")
```

sería

```
[ ('D',1), ('S',2), ('I', 4) ]
```

Apartado 3:

Implementar una herramienta que permita recorrer todo el corpus de test y verificación. Es decir, iría recorriendo una a una las líneas de cada fichero (que están alineadas), aplicaría sobre la frase de test el algoritmo básico de puntuación (apartado 1: **`addPunctuationBasic()`**) y a continuación comprobaría si el resultado es o no correcto usando la función **`verifyPunctuation()`** del apartado 2.

Obtener a continuación los valores relativos a precisión, exhaustividad (recall) y F1 para el algoritmo **`addPunctuationBasic()`** implementado en el apartado 1.

Consideraremos estos valores como el baseline, el modelo más básico de puntuación que podemos realizar para estudiar posibles mejoras.

Apartado 4:

Utilizando el corpus de entrenamiento contenido en PunctuationTask.train.en construir un modelo de lenguaje inspirado en la idea de 4-gramas. No es exactamente un 4-grama pero está basado en dicho modelo.

El objetivo de este pseudo 4-grama será predecir si en la posición P de un string debemos introducir un signo de puntuación (siempre estamos restringiendo el alcance a los seis signos de puntuación considerados en este ejercicio), o si debemos cambiar la palabra en dicha posición P por mayúscula.

En última instancia, el 4-grama contendrá tuplas de la siguiente forma:

```
(token1, token2, token3, operación)
```

donde token1, token2 y token3 serán tokens cualesquiera incluidos en el corpus de entrenamiento. Por tanto estos tokens podrán ser palabras o signos de puntuación, y 'operación' será una de las siguientes operaciones:

signo de puntuación (que podrá ser uno de los seis considerados: .,;;?!
mayúscula (que indica que la siguiente palabra se debe poner en mayúscula)
minúscula (que indica que la siguiente palabra deberá estar en minúscula)

La operación se decide observando el fenómeno más común (frecuencia relativa) de las distintas operaciones para cada tríada de tokens (token1 token2 token3).

Por ejemplo, podríamos detectar que para la tríada de tokens
(‘by’, ‘the’, ‘way’)
la operación más frecuente es insertar el signo de puntuación coma (,)

Una vez creado este modelo de lenguaje se debe implementar una segunda versión de la función que añade signos de puntuación denominada

```
string addPunctuation4gram(string)
```

que recibirá como en el apartado 1 un string de entrada, y devolverá el nuevo string con los cambios introducidos aplicando el modelo de lenguaje previamente entrenado con el 4-grama previamente indicado.

Apartado 5:

Aplicar el modelo de verificación implementado en el apartado 2, pero contrastando el corpus de referencia con el resultado generado por el algoritmo de puntuación basado en 4-gramas del apartado 4.

A continuación obtener los valores de precisión, exhaustividad (recall) y F1 sobre estos nuevos resultados y compararlos con los obtenidos en el apartado 3.

Este nuevo algoritmo de puntuación basado en 4-gramas, ¿mejora los resultados? Analiza si mejora o empeora, ¿por qué puede ocurrir?

Apartado 6:

Utilizando también ejemplos de TED talks, en un artículo de 2016, Ottokar Tilk y Tanel Alum ha aplicado un modelo de redes recurrentes bidireccionales para esta misma tarea (restauración de signos de puntuación en textos no segmentados).

El artículo donde lo describen se encuentra publicado en este enlace:

- https://www.isca-speech.org/archive/Interspeech_2016/pdfs/1517.PDF
- Bidirectional Recurrent Neural Network with Attention Mechanism forPunctuation Restoration (InterSpeech 2016).

El código correspondiente a su implementación se encuentra también disponible en github:

- https://github.com/ottokart/punctuator2

Los resultados de la evaluación para tres signos de puntuación concretos han sido estos (disponibles en la dirección github anterior):

PUNCTUATION	PRECISION	RECALL	F-SCORE
,COMMA	64.4	45.2	53.1
?QUESTIONMARK	67.5	58.7	62.8
.PERIOD	72.3	71.5	71.9
Overall	*68.9*	*58.1*	*63.1*

El objetivo de este apartado es estudiar la implementación que han hecho estos autores con una red neuronal y adecuarla al problema que se ha planteado en este ejercicio.

En concreto, este apartado consistirá en la adecuación del modelo de Tilk y Alum al escenario de este ejercicio. Es decir, aplicar el entrenamiento de la red propuesta por estos autores al corpus de entrenamiento (PuntuationTask.train.en) y a continuación evaluar el resultado obtenido usando el modelo de verificación implementado previamente.

Los resultados que obtienes para tu evaluación son similares a los publicados por estos autores.

Nota: Observar el modelo implementado por estos autores en el script error_calculator.py, como contraste a vuestro algoritmo de verificación (apartado 2) y evaluación (apartados 3 y 5).

Apartado 7:

A partir del trabajo realizado y teniendo en cuenta otros enfoques disponibles en el ámbito de las tecnologías del lenguaje, investiga qué otros enfoques o estrategias alternativas para esta misma tarea. El objetivo de este apartado es que busques bibliografía relevante reciente sobre esta tarea, selecciones uno o dos artículos y describas brevemente el enfoque y resultados obtenidos.

Presentación del ejercicio

La entrega del ejercicio consistirá en dos partes.

En primer lugar el proyecto mismo incluyendo todo el código implementado. Es importante que la entrega sea lo más “autocontenida” posible. Si se requiere la utilización de alguna librería o módulo no incluido con el proyecto, se deben indicar claramente las instrucciones para su instalación.

La implementación de los módulos 1 al 5 debe ser trabajo personal. Si se utiliza algún módulo externo debe explicarse qué se ha utilizado y con qué fin. Por ejemplo, no se considerará como válido en el apartado 4 utilizar una librería para la creación de n-gramas, ya que el objetivo es que cada alumno construya su propia implementación.

La segunda parte de la entrega será un informe relativo al desarrollo realizado, según el modelo ya indicado al principio del enunciado de este examen. En este caso se tratará de un documento que describirá las ideas clave que habéis utilizado en cada apartado. Podéis incluir parte del código fuente

que consideréis especialmente relevante, pero la clave es explicar el enfoque, algoritmos usados y resultados obtenidos, así como dificultades que se han encontrado e ideas que se os ocurren como parte del desarrollo y evaluación de la tarea. La extensión máxima de esta parte es de una cara de un folio por apartado, usando una letra tipo 12 puntos, e interlineado simple.

Valoración de los apartados

Los apartados de este ejercicio se puntuarán según el siguiente criterio

- Apartado 1: 0,5 puntos
- Apartado 2: 2,0 puntos
- Apartado 3: 1,5 puntos
- Apartado 4: 2,0 puntos
- Apartado 5: 1,5 puntos
- Apartado 6: 1.5 puntos
- Apartado 7: 1 punto