Proyecto de Grado
2008

Asistente de traducción de documentos basado en memoria y difusión
[Proyecto Baboon]

Viviana Folgar Rey **C.I. 4.060.547-4**
Sergio Pio Alvarez **C.I. 3.328.674-2**

Tutor: Ing. Ana Erosa

Cliente: Sofis Solutions / Gustavo Cirigliano, Santiago Atella

Resumen

En la actualidad, la traducción de documentos y la localización de software en diferentes idiomas es un requisito prácticamente indispensable para que las organizaciones puedan distribuir sus productos en diversos mercados. Desde hace mucho tiempo se han estudiado formas en que la informática puede asistir a las personas en el proceso de traducción, desde sistemas que solo realizan algún tipo de simplificación de la tarea hasta sistemas de traducción automática.

La empresa Sofis Solutions realiza como parte de sus actividades la traducción de documentos técnicos, para lo cual utiliza herramientas que no satisfacen sus necesidades, por esta razón se propuso contar con una herramienta propia. Tras realizar un prototipo de evaluación concluyó que era factible la construcción de un sistema de asistencia a la traducción dando origen a este proyecto de grado.

El proyecto comprende una investigación en el área de asistencia a la traducción por computadoras, especialmente en la asistencia a la traducción por memorias. Incluye un estudio teórico respecto de las herramientas existentes en el mercado y sus principales características, los beneficios que éstas proporcionan, y las técnicas y estándares utilizados comúnmente en su desarrollo. La investigación realizada se presenta como un estudio del estado del arte en el cual se analizan diferentes sistemas de asistencia a la traducción utilizando memorias, para concluir cuáles son las características comunes de todas ellas necesarias a considerar cuando se planifica la construcción de un nuevo sistema. Además describe diferentes formas de resolver los problemas que surgen durante el desarrollo, al igual que herramientas que permiten implementar las soluciones planteadas.

En base a lo investigado se diseña la arquitectura de un sistema de asistencia a la traducción por memorias y se realiza la implementación de una solución utilizando tecnologías de actualidad, como JavaEE, Java Server Faces y Enterprise Java Beans, entre otras. Para esta implementación se definen las características funcionales que el sistema debe presentar, las técnicas que más se ajustan a la solución diseñada, los estándares de representación de información más difundidos, como XLIFF y TMX, y las herramientas que permiten llevar a la práctica la solución, entre las que se encuentran Freeling y OpenNLP.

Como parte de la implementación se ensayan diferentes técnicas de búsqueda de candidatos utilizando algoritmos de comparación difusa entre segmentos de texto, tales como *Cantidad de lemas en común normalizado utilizando sinónimos*, *Levenshtein extendido para fragmentos de texto* y *Cantidad de tri-gramas en común*. Se realiza una comparación de las mencionadas técnicas y se obtienen conclusiones que permiten determinar cuáles se adecuan más para la traducción de documentos reales.

Finalmente se proponen extensiones al sistema implementado y se detallan algunos mecanismos para realizarlas.

Palabras clave: Traducción, Localización, Traducción automática, Asistencia a la traducción, Memoria de traducción, Freeling, OpenNLP, TMX, XLIFF, Comparación difusa, Fuzzy matching, Función de similaridad.

Indice

1 Introducción

1.1 Contexto

La traducción de documentos, así como de interfaces de software, es en general una tarea importante y parte fundamental de todo proceso, trabajo u organización, especialmente en estos tiempos en que la globalización exige que se presenten los mismos contenidos en múltiples mercados comprendiendo muchos y muy variados idiomas.

Las tareas de traducción son en sí mismas costosas, tanto en tiempo como en recursos, especialmente si se desea obtener resultados de buena calidad. Particularmente, en muchos proyectos, el tiempo que consume la tarea de traducción puede ser excesivo para los plazos manejados, ocasionando incrementos en los costos y provocando retrasos en las fechas de publicación o salida al mercado; este problema se ve acrecentado cuando el documento original es afectado por frecuentes cambios, los cuales muy probablemente deban ser trasladados a sus traducciones. Adicionalmente, otro problema que suelen enfrentar las organizaciones está referido a la calidad de las traducciones, tanto por la competitividad de los traductores involucrados como por el número de los mismos que trabajan sobre un proyecto. Cuanto mayor sea el número de personas trabajando en un proyecto, tanto más amplia será la gama de estilos, capacidades y características que se verán reflejadas en el resultado, algo que, en muchos casos, las organizaciones no pueden aceptar, requiriendo una homogeneidad en los estilos y vocabularios. Por ejemplo, el término inglés file podría ser traducido como archivo o fichero, pudiendo aparecer ambas traducciones en el mismo texto si en él intervienen dos traductores con diferentes orígenes o costumbres; lo mismo sucede en cuanto a estilos, pudiéndose traducir la oración *the big black dog run* como *el gran perro negro corrió* o *el perro negro grande corrió* o *el perro grande y negro corrió*. Un tercer problema que se debe enfrentar en un trabajo de traducción, es que el trabajo en sí mismo es monótono y los traductores suelen caer en el aburrimiento y el desgano, lo que redunda en un decaimiento del rendimiento y la calidad.

Para solventar los problemas antes mencionados, la ciencia informática ha desarrollado diferentes soluciones, siendo una de ellas los sistemas de asistencia a la traducción, en particular los llamados sistemas de memorias de traducción. Estos sistemas basan su funcionamiento precisamente en las llamadas memorias de traducción, que son, en pocas palabras, repositorios de traducciones. Cuando se usa uno de estos sistemas, los traductores son enfrentados al texto en forma segmentada, por lo que deben traducir segmento por segmento el documento. La granularidad del segmento es variable según el sistema, pudiendo tratarse de párrafos, oraciones, y hasta palabras sueltas. Para cada segmento, el sistema puede buscar en uno o más repositorios otros segmentos traducidos previamente que sean iguales o similares y así proponerle al traductor una o más traducciones. De esta forma, se logra reducir los tiempos puesto que el trabajo manual del traductor es menor, se gana en homogeneidad de estilos ya que la traducción del nuevo segmento seguirá el estilo de otro segmento traducido previamente, se mantiene una consistencia en el vocabulario y se logra que un fragmento de texto que aparece varias veces en un mismo documento, o en varios documentos del mismo proyecto, sea traducido en todos los casos de la misma manera.

1.2 Motivación

Teniendo en cuenta los problemas antes mencionados, y los beneficios aportados por los sistemas de asistencia a la traducción, la empresa Sofis Solutions se propuso contar

con una herramienta de este tipo. Con esta herramienta se propone realizar el trabajo de traducción de documentos y localización de software más rápidamente y eficientemente, con la posibilidad de comercializarlo en un futuro cercano.

Como primer acercamiento, procedió a implementar un prototipo de sistema de memoria de traducción, bastante sencillo pero suficiente para evaluar positivamente la construcción de una herramienta propia, a la que denominó Sofis Glosario.

A grandes rasgos, el sistema Sofis Glosario implementa una memoria de traducción sumamente básica que solo es capaz de procesar documentos en un formato XML preestablecido y muy limitado y solo es capaz de realizar búsquedas por coincidencias exactas. Propone una división del trabajo según usuarios, pudiendo asignar los documentos a los usuarios, los cuales están identificados por un nombre y una contraseña. Sin embargo, no permite hacer la división del trabajo según proyectos, ni mantener ningún otro tipo de control, tal como costos y tiempos. El prototipo se describe con mayor detalle en el apéndice [ap-sofisglosario]. La conclusión que se obtiene de este sencillo proyecto es que una herramienta de memorias de traducción sería muy útil para mejorar el proceso de la traducción de documentos, pero el prototipo realizado es completamente insuficiente.

Con la realización de este proyecto se desea estudiar y superar las debilidades del sistema existente, entre las que se encuentran el soporte de un único formato de documento y las búsquedas por coincidencia exacta. El sistema a desarrollar debe trabajar con diferentes tipos de documentos, entre los que se encuentran los más difundidos: Microsoft Word, OpenOffice, HTML, XML y texto. Las búsquedas en la memoria de traducción se deberán realizar utilizando algoritmos de coincidencia difusa, y se permitirá la edición de los textos sugeridos por el sistema . Adicionalmente, el sistema debe permitir realizar la división del trabajo según clientes, proyectos y usuarios.

Manejamos otras funcionalidades adicionales que debimos dejar fuera del alcance del proyecto. Estas funcionalidades abarcaban capacidades administrativas y estadísticas, como la estimación de tiempos y costos, y la facturación a clientes y traductores. Igualmente, aunque se dejó abierta la posibilidad de que el sistema permita trabajar con múltiples idiomas, solo se tenía el requerimiento de trabajar con inglés y español.

Los sistemas de asistencia a la traducción por memorias constituyen un área de investigación y producción relativamente nueva, por lo que si bien se han producido grandes avances y existen variadas implementaciones, algunas comerciales y otras disponibles libremente en Internet con diferentes características y particularidades, es seguro que aún existe mucho por hacer. En particular, en lo que a Sofis Solutions se refiere, los sistemas existentes en el mercado son costosos y cerrados, por lo que tomó la decisión de contar con una herramienta propia.

1.3 Objetivos

El objetivo del proyecto es investigar las características generales de los sistemas de memorias de traducción existentes actualmente, así como las tendencias para su desarrollo, para posteriormente proceder a la implementación de un software completamente funcional y que pueda ser extendido en trabajos futuros.

En primer lugar realizaremos un estudio del estado del arte donde se relevarán los proyectos existentes relacionados con el área de traducción por memorias con el fin de detectar aspectos que puedan servir de base para la realización de este proyecto. Adicionalmente, las herramientas que tienen que ver con el procesamiento de textos nos pueden ser de gran utilidad.

En segundo lugar, también realizaremos un estudio sobre técnicas y estándares en el

área. Esperamos detectar formas comunes de representar la información, tanto para introducirla al sistema como para almacenarla, así como algoritmos útiles y mecanismos de implementación de las funcionalidades que sean necesarias.

En tercer lugar, relevaremos las características de los sistemas comerciales existentes en la actualidad, de forma de determinar cuales son las que debe presentar necesariamente un sistema de este tipo, e identificar aquellas que sean de mayor interés para un sistema de memorias de traducción general.

Estableceremos las alternativas implementables en el marco de este proyecto, para posteriormente construir el software. Finalmente identificaremos y plantearemos las potenciales mejoras y extensiones que se pueden realizar.

1.3.1 Resultados Esperados

Como resultado del proyecto nos proponemos obtener un estudio completo del estado del arte, el diseño de un sistema de asistencia a la traducción por memorias, un software producto de la implementación de dicho sistema y el listado de las mejoras y extensiones que se deban o puedan ser aplicadas sobre el mismo.

Creemos que la definición de la arquitectura de software debe ser clara, de forma que nos permita la construcción del sistema en el marco de este proyecto y la utilización de la misma para trabajos futuros.

En cuanto a la implementación pretendemos obtener un software completamente funcional, que permita realizar la traducción de documentos en varios formatos, entre ellos Microsoft Word, HTML, OpenOffice y texto plano, entre los idiomas inglés y español, aunque con la posibilidad de ser extendido para otros idiomas y formatos.

1.4 Organización del Documento

A continuación describimos la organización del documento.

El capítulo 2 presenta el estado del arte en el área de la asistencia a la traducción por memorias, incluyendo un relevamiento de sistemas de asistencia a la traducción existentes, y los estándares comúnmente utilizados en su diseño y construcción, así como herramientas útiles para ello. El capítulo 3 describe el análisis realizado, conjuntamente con el diseño de la solución. El capítulo 4 describe la implementación de la solución propuesta, y las pruebas realizadas. El capítulo 5 recoge las conclusiones obtenidas como resultado del proyecto de investigación y desarrollo. Finalmente, el capítulo 6 presenta el glosario, el capítulo 7 lista las referencias utilizadas en el proyecto, y el capítulo 8 lista los apéndices que acompañan este informe.

2 Estado del Arte

En este capítulo resumimos la investigación que llevamos a cabo durante las primeras etapas del proyecto, en forma del estado del arte. Explicamos la razón de ser de los sistemas de asistencia a la traducción, y en particular aquellos basados en memorias de traducción, contemplando su origen y evolución hacia nuestros días, hasta alcanzar el relevamiento de los sistemas actualmente disponibles, proyectos relacionados y herramientas útiles para la construcción de un nuevo sistema. También enumeramos las principales características que presentan los sistemas de asistencia a la traducción por memorias.

2.1 Traducción

La traducción de documentos y textos presentes en las piezas de software es una tarea esencial en la gran mayoría de los proyectos y para casi todas las organizaciones. Esto ha sido así desde hace muchos años, y se ha resaltado el los últimos tiempos, en los que un mundo globalizado exige que los contenidos estén disponibles en la mayor cantidad posible de idiomas para acceder a los más variados mercados y públicos. Adicionalmente, la calidad también ha ido tomando relevancia: los documentos deberían ser tan buenos en cualquier idioma como en el idioma original; no hace mucho tiempo alcanzaba con producir documentos en el idioma inglés, pero cada vez más los usuarios exigen la localización de los documentos y software en su propio idioma. Esto puede verse en los manuales de usuario que acompañan los aparatos electrodomésticos y los productos de software, en los propios productos (sistemas operativos, procesadores de textos, incluso juegos) y libros de texto.

2.2 Problemas inherentes a la traducción de documentos

En la sección 1.1 mencionamos las principales dificultades que se presentan al abordar la tarea de traducción de documentos. A continuación las describimos con mayor detalle.

Excesivo consumo de tiempo y recursos

El proceso de traducción suele consumir mucho tiempo y recursos, a veces más de lo que se desea invertir. Es frecuente que se deban realizar varias revisiones del texto, ya que en ocasiones no queda del todo claro el sentido o significado de algunas oraciones, expresiones o términos hasta haber leído el texto por completo. En otras ocasiones se debe volver atrás al detectar que alguna traducción no corresponde con una similar en otra parte.

Competencias de los traductores

La calidad de la traducción depende de las habilidades de los traductores ya que se requiere un muy buen conocimiento de las lenguas involucradas así como un buen conocimiento de la temática, ya que frecuentemente están involucrados conceptos propios del tema, siglas, referencias a otros documentos, términos que pueden tener un significado diferente al común o ser exclusivos del área.

Diferencias entre traductores

Diferentes traductores tienen estilos y características muy disímiles. Por ejemplo, algunos traductores prefieren utilizar voz pasiva mientras que otros utilizan voz activa, algunos prefieren utilizar el sustantivo antes que los adjetivos mientras que otros lo hacen al revés. Esta heterogeneidad de estilos lleva a que el resultado de la traducción no sea uniforme.

Decaimiento de calidad y rendimiento

El trabajo de traducción suele ser una tarea monótona, lo que lleva a que los traductores pierdan la motivación decayendo el rendimiento y la calidad del resultado.

Repetición de textos y pérdida de coherencia en sus traducciones

Aunque no es un problema en sí mismo, cuando se trata de manuales y software, muchos fragmentos de texto se repiten a lo largo de un mismo documento y entre diferentes textos. Cuando la traducción se hace manualmente, el traductor debe traducir una y otra vez el mismo texto ocasionando que, potencialmente, un mismo fragmento de texto sea traducido de forma diferente, afectando a la coherencia.

2.3 Sistemas de traducción automática

Desde sus comienzos, la ciencia informática trató de encontrar soluciones para los problemas antes mencionados. En un primer momento, los esfuerzos estuvieron enfocados en la construcción de traductores automáticos, capaces de tomar como entrada un texto en un idioma determinado y producir como salida la traducción de dicho texto en otro idioma. Esta técnica demostró pronto ser más compleja de lo que parecía en un principio.

Los primeros sistemas de traducción automática estuvieron orientados a la traducción literal, palabra por palabra, a veces con sencillas transformaciones gramaticales, resultando en traducciones de pésima calidad, apenas útiles para que una persona pudiera *adivinar* el contenido del texto. La conclusión principal fue que los lenguajes son muy diferentes entre sí y demasiado complejos, y que es necesario transformaciones mucho más elaboradas. Se intentó entonces llevar a la práctica dos antiguas teorías: una que supone que dados dos lenguajes en la mayoría de los casos es posible encontrar transformaciones lingüísticas que permitan transformar uno en otro; y otra que afirma que existe un lenguaje básico común, al que la mayoría de los lenguajes existentes pueden ser transformados. Los partidarios de la primer teoría trabajaron en *traductores directos* (de un lenguaje a otro), mientras que los partidarios de la segunda trabajaron en *traductores indirectos* (de un lenguaje a otro base, llamado *interlingua*, y de éste al lenguaje destino). En ambos casos fracasaron, y hasta el día de hoy no existen traductores tan efectivos como se desearía.

Viabilidad de los sistemas de traducción automática

Cabe la pregunta acerca de si es posible en algún momento hacer de la traducción automática una realidad. Ningún programa de computador es capaz de realizar traducciones cuya calidad se asemeje, ni siquiera remotamente, a la de una traducción realizada por un traductor profesional [1]. El principal problema para llevar a cabo esta tarea es la ambigüedad inherente a las lenguas. A lo largo de siglos, los seres humanos han desarrollado diferentes lenguajes, más o menos complejos, pero en todos los casos con una base en la capacidad de los individuos de entendimiento y comprensión. Mientras una persona generalmente no tiene problemas para reconocer oraciones e identificarlas con un sentido particular, hasta el software más avanzado aún tiene problemas con términos que resultan sencillos para un niño.

Sin embargo, nada implica que los sistemas de traducción automática sean inexistentes. Tal vez, cambiando un poco la definición de traductor automático se logre encontrar exactamente lo que se precisa [2]: si en lugar de enfocar los sistemas en la suplantación de los traductores manuales se enfoca en la reducción de los problemas que implican las barreras de idiomas, se puede definir un sistema de traducción automática como *un software capaz de reducir los efectos negativos de las barreras de idiomas* [2].

Bajo esta definición, el problema se enfoca en la palabra *reducir*: para algunos usuarios, reducir puede significar disminuir costos, para otro minimizar el tiempo de entendimiento, e incluso para otros simplemente comprender un texto expresado en otro idioma, aunque la traducción no sea perfecta.

En general, lo anterior conduce por varios caminos, mientras en algunos casos la tendencia es reducir el espacio de opciones al mínimo de forma de producir soluciones excelentes en campos muy reducidos, en otros casos la tendencia es ampliar dicho espacio, de forma de que las soluciones sean tan solo adecuadas para permitir el entendimiento o la reducción de costos, a cuenta de la calidad de la traducción.

2.4 Sistemas de asistencia a la traducción

Tras el relativo fracaso de los traductores automáticos, surgió la idea de los sistemas de asistencia a la traducción, en los que el trabajo principal sería realizado por seres humanos con la asistencia de las computadoras. Los más sencillos solo se limitaban a traducir cada una de las palabras del texto y correspondía al usuario reacomodarlas en el nuevo idioma; algunos más avanzados lograban inferir varias alternativas de las cuales el usuario podía elegir la más adecuada.

Entre estos sistemas surgieron los sistemas de memorias de traducción, los cuales registran las traducciones hechas por el ser humano, y cada vez que un nuevo texto debe ser traducido, intentan encontrar textos similares traducidos anteriormente para proponerlos como posibles traducciones. El nombre de estos sistemas se debe a que basan su funcionamiento en memorias de traducción, es decir en repositorios de textos en un idioma dado y sus traducciones en otros idiomas.

2.5 Sistemas de memorias de traducción

La tarea de un sistema de memorias de traducción es almacenar porciones de texto, comúnmente llamadas segmentos, conjuntamente con su traducción en uno o más idiomas. Un conjunto de segmentos relacionados es llamado memoria de traducción o simplemente memoria y puede ser guardada en archivos o bases de datos.

La idea fundamental tras las memorias de traducción es dividir los documentos en segmentos conceptuales para luego buscar en la memoria posibles coincidencias para su traducción. La traducción inicial de cada uno de los segmentos normalmente será realizada por un traductor en forma manual. Cuando se encuentran coincidencias para un segmento particular, se le asigna a cada una de ellas una *medida de similaridad*, de forma de ayudar al traductor a elegir la traducción más adecuada, si es que alguna se ajusta; pudiera ocurrir que ninguna de las traducciones sea aplicable y por lo tanto el traductor deberá realizar la traducción y el sistema registrará un nuevo segmento en la memoria. Un par conteniendo un mismo segmento en dos idiomas es llamado unidad de traducción (Translation Unit, TU).

Coincidencia exacta y difusa

Cuando se habla de coincidencia entre segmentos, se debe distinguir entre dos tipos: exacta (full coincidence) y difusa (fuzzy coincidence). La coincidencia exacta se da cuando dos segmentos son exactamente iguales entre sí. Las coincidencias exactas rara vez se encuentran en el trabajo diario, sino que generalmente se tienen segmentos que son parecidos; este tipo de coincidencia se suele denominar coincidencia difusa o coincidencia parcial. Ante la coincidencia difusa, es probable que se tengan diferentes segmentos que presenten un cierto nivel de similaridad con el segmento a traducir.

El trabajo de un sistema de memorias de traducción es identificar los segmentos que

presentan algún tipo de coincidencia con el segmento a traducir, asignarle la *medida de similaridad* y ofrecer las diferentes alternativas para que el traductor pueda seleccionar entre ellas la que mejor se ajusta o, en caso de que ninguna sea adecuada, pueda ingresar una traducción manual; en este último caso el sistema debe registrarla para uso futuro.

Utilidad de los sistemas de memorias de traducción

Las memorias de traducción son especialmente adecuadas para documentos repetitivos o monótonos, tales como manuales técnicos, y para la traducción incremental de documentos (documentos que cambian frecuentemente con el tiempo pero que en gran medida mantienen el texto), así como la localización de software.

Además de la asistencia en la traducción, los sistemas de memorias de traducción aseguran la consistencia en los documentos, tanto en estilo (*coherencia*) como en terminología (*concordancia*).

Concordancia bilingüe

Para algunos investigadores, la expresión memoria de traducción tiene un significado diferente. Para ellos, entre quienes se encuentran los integrantes del laboratorio RALI, de la Universidad de Montreal, una memoria de traducción es una base de datos que contiene pares de fragmentos de texto en dos idiomas específicos, en la cual se puede realizar búsquedas avanzadas, en base a ideas o conceptos en lugar de hacerlas en base a palabras. Esta definición es normalmente llamada *concordancia bilingüe* y está orientada tanto a traductores como a lingüistas, o cualquier persona que necesite expresar una idea en un idioma diferente al suyo [3] .

Según [3] no existe la necesidad de elegir una u otra definición, ya que ambas son útiles en determinados casos. Por ejemplo, para los casos en los que se trabaja con una nueva versión de un documento (actualizaciones) o en casos en los que hay una alta tasa de repetición es muy útil un sistema de memorias de traducción según la definición tradicional, pero para textos que no contienen un alto nivel de repetición un sistema de concordancia bilingüe seguramente es mejor.

2.5.1 Funcionamiento de sistemas de memorias de traducción

Básicamente, un sistema de memorias de traducción funciona en cinco etapas: carga de la memoria, preprocesamiento del texto a traducir, segmentación, traducción/escaneo y rearmado.

Carga o importación de la memoria

La carga o importación de la memoria, consiste en la introducción en la memoria de un conjunto de textos con su respectivas traducciones, puesto que, de comenzar a trabajar con una memoria completamente vacía, sería nulo el beneficio obtenido ya que para ningún texto se encontraría otro similar ya traducido. Este trabajo se suele hacer a partir de memorias de traducción producidas anteriormente.

Pero aún sin contar con una memoria de traducción inicial, queda una alternativa, mediante el proceso conocido como *alineación*. Este proceso toma como entrada un mismo documento en dos idiomas diferentes e intenta identificar segmentos correspondientes en ambos, ingresando los que encuentra en la memoria de traducción. Normalmente esto es hecho por programas especialmente diseñados para esta tarea, llamados *alineadores*. La tarea de alineación no es sencilla debido a que es común que lo que en un idioma se considere una sola oración, en otro idioma amerite más de una.

Preprocesamiento

El preprocesamiento del texto a traducir consiste en quitar todos aquellos elementos que no forman parte del texto en sí mismo sino que son decorativos, tales como indicadores de formato, imágenes y fragmentos de texto no traducibles (por ejemplo, código fuente). Estos elementos deben quitarse porque no solo introducen ruido en el texto, sino que puede inducir a confusión al traductor. De la misma manera el reconocimiento de elementos textuales especiales tales como nombres propios, números o fechas también se podría tener en cuenta durante el preprocesamiento. Estos elementos pueden ser la única diferencia entre dos segmentos y normalmente no requieren traducción. Por ejemplo, los segmentos *System's User Manual version 1.0* y *System's User Manual version 2.0* son materialmente los mismos, salvo por los números de versión, los que no requieren traducción.

El preprocesamiento puede realizarse bien manualmente o bien mediante alguna herramienta especialmente diseñada para esto, generalmente llamada *filtro.*

Segmentación

La segmentación consiste en dividir el texto en fragmentos más reducidos, llamados segmentos. Esta etapa normalmente suele hacerse simultáneamente con la etapa anterior, siendo en la práctica una especificación de la misma ya que mientras que en el preprocesamiento se remueven los elementos superfluos del texto, los fragmentos resultantes son en sí mismos segmentos de texto traducibles. Sin embargo, existe una discusión acerca de la *granularidad* que deben presentar los segmentos, desde la máxima granularidad posible, es decir, palabras individuales, hasta la mínima, o sea el texto completo. En este rango, se destacan dos alternativas ampliamente utilizadas, como son las oraciones y los párrafos, siendo la primera de éstas la más usual. Cuanto menor sea la granularidad, menos serán los segmentos a traducir, pero también será menor la posibilidad de encontrar otros textos que tengan una semejanza adecuada; por otra parte, cuanto mayor sea la granularidad, se corre el riesgo de trabajar con palabras aisladas, extraídas de todo contexto y por tanto sin una clara significación o sentido.

Traducción

La traducción es la etapa que implica mayor trabajo por parte del traductor. El sistema le presenta al usuario los segmentos en secuencia; para cada uno de ellos puede también presentarle un conjunto de posibles traducciones extraídas de la memoria, de las cuales el usuario puede elegir una y potencialmente modificarla o realizar una traducción completamente manual. Para cada candidato encontrado, el sistema calcula su medida de similaridad con el segmento a traducir para ayudar al usuario a decidir la traducción más apropiada.

Durante esta etapa el sistema aplica la técnica llamada *escaneo*, en la que busca, para cada segmento, traducciones apropiadas para el mismo en la memoria de traducción.

Rearmado

Finalmente, la última etapa consiste en volver a combinar el formato extraído del texto original con el texto traducido, con el objetivo de obtener un documento con un formato similar al original pero con el texto en el nuevo lenguaje.

2.5.2 Escaneo

Como dijimos anteriormente, el escaneo es el proceso por el cual un sistema de memorias de traducción busca en su memoria traducciones para segmentos similares a cada uno de los segmentos que se deben traducir. Esto generalmente se hace utilizando

coincidencia difusa.

Métrica de similaridad

La comparación difusa comprende un conjunto de técnicas para encontrar segmentos que son similares a uno dado. El concepto clave es el de función de similaridad, que intenta cuantificar el grado de coincidencia que presentan dos segmentos. Existen dos tipos de funciones de similaridad: las que calculan valores porcentuales (en el rango 0 a 100, donde 100 es coincidencia exacta) y las que calculan distancias, siendo el óptimo el valor 0, y cuanto mayor sea el valor obtenido menor será la similaridad entre los segmentos.

Algoritmos on-line y off-line

Tradicionalmente, los algoritmos de comparación difusa han sido divididos en dos categorías: *on-line* y *off-line* [web-25]. La diferencia fundamental entre ellos es que los primeros deben hacer la búsqueda sin la posibilidad de contar con índices, puesto que el espacio de búsqueda se va conociendo a medida que se avanza, mientras que los segundos basan su funcionamiento en índices ya que el espacio de búsqueda se conoce por completo antes de comenzar.

Los algoritmos on-line generalmente se basan en técnicas de programación dinámica y son continuamente mejorados. Sin embargo su principal inconveniente es que su desempeño para textos de gran tamaño es, generalmente, inaceptable. Los algoritmos off-line presentan el inconveniente de que rara vez es posible contar con el espacio de búsqueda completo antes de comenzar.

2.5.3 Funciones de similaridad comunes

A continuación detallamos algunas metodologías para calcular la similaridad entre fragmentos de texto.

Similaridad de Jaccard

La similaridad de Jaccard, también conocida como coeficiente de similaridad de Jaccard [web-24] es uno de los métodos más simples pero relativamente efectivos, aunque poco usado debido a la existencia de otros mejores. En este método se consideran a los textos como bolsas de palabras (conjuntos sin orden y en el cual pueden repetirse elementos). La similaridad de Jaccard es tan simple como calcular la relación entre las palabras comunes a los fragmentos, sobre el total de palabras resultante de la concatenación de ellos:

$$Jaccard(A,B)=\frac{|(A\cap B)|}{|(A\cup B)|}$$

Distancia de n-gramas

Una de las funciones de similaridad más usada es la llamada *distancia de n-gramas* [12] , la cual pertenece al grupo de algoritmos on-line. Consiste en determinar la relación entre la cantidad de n-gramas en común entre dos textos y la cantidad total de n-gramas. Un n-grama es una secuencia de n elementos, donde los ítems pueden ser letras, palabras o sintagmas, entre otros.

Distancia de Levenshtein

La distancia de Levenshtein [web-2, web-3], también conocida como distancia de

edición simple, es un tipo de medida basada en la *distancia de edición* (cambios necesarios para transformar un texto en otro) y pertenece al grupo de algoritmos de comparación off-line. Se define la distancia de edición entre dos textos como el costo de transformar uno en el otro, utilizando la menor cantidad de operaciones necesarias. La distancia de Levenshtein reconoce como operaciones a las siguientes:

- Inserción: agregar un carácter (distancia 1).
- Eliminación: quitar un carácter (distancia 1).
- Sustitución: quitar un carácter y en su lugar colocar otro diferente (distancia 1).
- Transposición: intercambiar de lugar dos caracteres (distancia 2). Esta operación es contemplada por la extensión de Damerau-Levenshtein.

Aunque existen muchas variantes, la mayoría solo son aplicables a textos muy cortos, generalmente palabras aisladas, y existen pocas implementaciones eficientes para textos más largos. Sin embargo, en [web-1] se describe un algoritmo aplicable a oraciones.

Métodos de Jaro y Jaro-Winkler

El método planteado por Jaro (1995) es uno de los más difundidos de entre los que no son basados en la distancia de edición. El método se basa en el número y posición de los caracteres entre las cadenas, como se explica a continuación [5].

Si A = a1...am y B = b1...bn son dos cadenas de largo m y n respectivamente, se dice que un carácter ai de A es común con B (o compartido con B) si existe algún bj tal que ai = bj y (i – H) ≤ j ≤ (i + H), donde H = (min(m, n) / 2). De esta forma se definen dos nuevas cadenas: A' = a'1...a'm' y B' = b'1...b'n' compuestos por los caracteres comunes de A y B respectivamente, en el mismo orden en que aparecen originalmente. En base a estas dos nuevas cadenas, se define una transposición en la posición k si a'k ≠ b'k. Se define entonces la magnitud TA',B' como la mitad de la cantidad de transposiciones que existen entre A' y B'. Finalmente, se define la métrica de similaridad de Jaro según la siguiente fórmula:

$$Jaro(A,B)=\frac{1}{3}\left(\frac{m'}{m}+\frac{n'}{n}+\frac{m'-T_{A',B'}}{m'}\right)$$

El método de Jaro-Winkler (1999) [web-26] es una extensión del método de Jaro en el que además se tiene en cuenta la longitud del prefijo más largo común a ambas cadenas. Si se define la magnitud P como el largo del prefijo más largo común a A y B, a continuación se define P' = max (P, 4). La expresión de Jaro-Wrinkler toma la siguiente forma:

$$Jaro-Wrinkler(A,B)=Jaro(A,B)+\frac{P'}{10}(1-Jaro(A,B))$$

Tanto los algoritmos de Jaro como de Jaro-Wrinkler son ideales para cadenas cortas, pero se vuelven altamente ineficientes a medida que la longitud de las entradas crece. Son ampliamente utilizados en búsqueda de nombres y direcciones en bases de datos.

TFIDF

TFIDF, también llamado *similaridad por coseno* (cosine similarity) es otro de los algoritmos que, al igual que Jaccard, toman como base las palabras que conforman las cadenas. Es un algoritmo bastante complicado de comprender e implementar aunque es ampliamente utilizado para la recuperación de información (típicamente, para búsqueda de entidades con nombre).

Funciones de distancia de nivel 2 - Monge-Elkan

En algunos casos, es posible lograr mejores medidas de similaridad entre cadenas combinando dos o más funciones de similaridad. Cuando se combinan dos funciones al resultado se le suele denominar función de distancia de nivel 2. Una de las más usadas es el *esquema de unificación recursiva* (recursive matching scheme), propuesto inicialmente por Monge y Elkan [7, 11]. Bajo este método, las cadenas de entrada A y B son descompuestos en cadenas menores, tal que A = A1...Am y B = B1... Bn, a partir de los cuales se define la similaridad entre A y B como el máximo entre la similaridad de una subcadena de A con una de B, según la siguiente fórmula:

$$Monge-Elkan(A,B)=\frac{1}{m}\sum_{i=1}^{m} max_{j=1}^{n}(simil(A_i, B_j))$$

En la fórmula, simil es cualquier función de similaridad, llamada segunda función. En [5] indican que han hecho experimentos usando las funciones de Jaro, Monge-Elkan, y Jaro-Winkler.

Sistema Oliver

Es el utilizando para la implementación de algunas funciones de búsqueda de patrones en cadenas de caracteres, en el lenguaje PHP, particularmente en la función *similar_text*. Notar que PHP provee varias funciones para realizar búsquedas de texto por similaridad, y para la cuantificación de la similaridad de dos cadenas. Por más detalles ver la documentación de PHP y la implementación en C de las funciones de strings [web-4].

Sistema Metaphone

El sistema M*etaphone* es un algoritmo de búsqueda de coincidencias de texto propuesto originalmente por Lawrence Phillips en 1990 (Computer Language v7 n12, diciembre de 1990, págs. 39-43.). Está basado en otro sistema anterior, SoundEx, utilizado en Estados Unidos para el registro de personas.

Básicamente, el algoritmo funciona transformando los textos a un alfabeto reducido, compuesto por 21 símbolos, basándose principalmente en la similaridad fonética. Este nuevo alfabeto tiene la particularidad de que cada símbolo tiene una correspondencia "muchos a muchos" con el alfabeto de uso corriente. Además, estadísticamente, la mayor parte de las palabras y nombres propios que se construyen con el alfabeto tradicional inglés pueden ser construidas con no mas de 4 símbolos del nuevo alfabeto. Para el idioma inglés, el nuevo alfabeto está compuesto por las letras A, E, I, O, U, B, X, S, K, J, T, F, H, L, M, N, P, R, 0 (cero representa el sonido TH), W, Y. Adicionalmente, no existe diferencia entre mayúsculas y minúsculas. Con un estudio apropiado, puede lograrse determinar un alfabeto para otros idiomas, en particular para el español.

Podas

El enfoque de fuerza bruta (comparar un segmento con todos los disponibles, uno a la vez) es el más directo que puede ser aplicado en el escaneo, pero naturalmente no es eficiente. A medida que la cantidad de segmentos que deben ser comparados aumenta, también aumenta el tiempo requerido. Por esta razón, se debe realizar algún proceso previo de poda del espacio de búsqueda, para considerar solo aquellos segmentos que en lo previo puedan dar resultados útiles. La idea aquí es determinar cuáles segmentos darán resultados útiles sin antes hacer una comparación con ellos. Para esto, se han investigado algunas alternativas.

En general, es normal asociar a cada segmento un conjunto de variables que permitan realizar la tarea de poda, y que potencialmente sea común a todos los segmentos de un

mismo tipo [5]. Por ejemplo, se podría asociar a cada segmento una variable que indique la temática del texto original del que fue extraído, de forma de que, en principio, dado un segmento, se tenga mayor probabilidad de encontrar los segmentos más similares entre aquellos extraídos de la misma temática (informática, economía, etc). Para comenzar, el hecho de buscar solo entre aquellos segmentos que comparten el mismo idioma es en sí un método de poda, ya que resulta natural pensar que habrá mayor similaridad con algún segmento del mismo idioma que con algún otro de un idioma diferente; en este caso, la variable considerada es el idioma de origen.

Otra forma de realizar la poda es mediante el uso de la técnica que llamamos *de diccionario*, que consiste en registrar todas las palabras no genéricas que aparecen en los segmentos ya procesados, indicando para cada una todos los segmentos en los que aparece. Incluso este método puede ser mejorado utilizando los lemas, las raíces o los sinónimos de las palabras.

2.5.4 Características de los sistemas de memorias de traducción

A continuación se describen algunas características generales que se han observado en las diferentes implementaciones de memorias de traducción o proyectos relacionados.

Características estructurales

Existen dos maneras de implementar la memoria de traducción. Una de ellas es como una base de datos en la que se almacenan los pares de segmentos (en lengua origen y lengua destino) complementado con algún tipo de información adicional denominada *metadata* [4] (fecha de creación, autor, texto origen, etc). La otra es mediante conjuntos de archivos con anotaciones, conteniendo el texto completo en ambos idiomas, e información que permite relacionar los segmentos; ésto es conocido como *corpus bilingüe*.

Se debe también considerar si se admiten múltiples traducciones para un mismo segmento o si cada nueva traducción sustituye a una anterior. De la misma manera se debe establecer si se admiten segmentos duplicados o si cada uno puede aparecer solo una vez con múltiple metadata.

Gestión de proyectos

Los sistemas de memorias de traducción suelen dividir su espacio de trabajo en *proyectos*. Un proyecto está compuesto por un conjunto de documentos que deben ser traducidos, conjuntamente con información sobre el cliente, traductores asignados, plazos, costos, etc. Entre las ventajas obtenidas de hacer esta división se encuentra el poder mantener una coherencia de estilos entre los diferentes archivos traducidos dentro del mismo proyecto, a la vez que puede haber diferencias marcadas entre proyectos.

Gestión de terminología

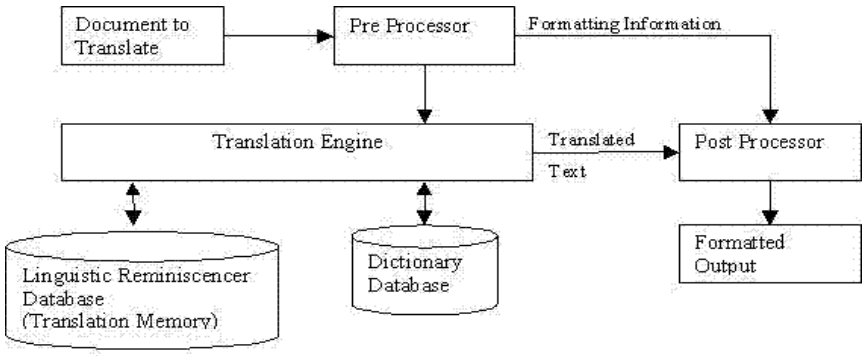
Las diferentes áreas, temas o incluso clientes, presentan su propia terminología específica. Es una tarea importante y consumidora de tiempo y recursos recolectar los términos, sus definiciones y su traducción, ya que no suelen encontrarse en diccionarios comunes y bilingües o corpus difundidos, o a veces existiendo las traducciones se utilizan acepciones fuera de lo común. El software especializado para esta tarea es denominado *Sistema de Administración de Terminología (Terminology Management System o TMS).*

Una forma de realizar la carga del TMS es mediante el procesamiento de una cierta cantidad de textos representativos escogidos, de forma de poder identificar aquellos

términos que no se encuentren en los diccionarios tradicionales; este proceso es llamado *extracción de términos*.

Características arquitectónicas básicas

En general, todas las herramientas de asistencia a la traducción por memorias mantienen una arquitectura básica común, similar a la que se muestra a continuación:



Esta imagen, correspondiente a la implementación de Trados Translation's Workbench [6], muestra los elementos principales que componen la parte directamente asociada a la traducción de documentos. Funcionalmente, el procesamiento es como sigue: partiendo de un documento a traducir, un preprocesador se encarga de extraer de él información de formato que posteriormente será utilizada para volver a reconstruir el documento con la misma presentación en un nuevo idioma. Luego de extraer dicha información y registrarla, el preprocesador redirecciona el flujo hacia el llamado motor de traducción, mediante el cual los traductores podrán trabajar con los documentos, y en el que también pueden existir capacidades de traducción automática y fundamentalmente de asistencia de traducción por memorias. Para esto, el motor de traducción interacciona con una memoria de traducción, y potencialmente con un diccionario multilingüe. Finalizado el trabajo de traducción, la salida del motor de traducción es combinada con la información de formato extraída en el comienzo, produciendo un nuevo archivo de salida.

2.6 Relevamiento de sistemas de memorias

Existen en la actualidad múltiples herramientas que implementan variantes del proceso anteriormente descrito, algunas de ellas comerciales y otras disponibles gratuitamente en Internet. A continuación brindamos una breve descripción de los principales sistemas de asistencia a la traducción por memorias (CAT).

SDL Trados

SDL Trados [web-5] es uno de los paquetes de asistencia a la traducción mas famosos y completos del mercado, siendo una de las principales referencias; muchas veces la evaluación de cualquier herramienta CAT incluye una comparación con Trados. Está conformado por diferentes módulos, tales como un sistema de memorias de traducción, un alineador de textos y un gestor de terminología, entre otros. Soporta más de 80 idiomas y muy variados formatos de archivos.

OmegaT

OmegaT [web-6] es una de las implementaciones más difundidas en el ambiente Open Source. Está implementada totalmente en Java, por lo que existen versiones para MS

Windows, Linux, Mac y Unix. Permite trabajar simultáneamente con múltiples memorias de traducción e importar y exportar las memorias de traducción en el estándar TMX. Aunque es la herramienta open source mas usada no está al nivel de las comerciales ya que, por ejemplo, no brinda soporte para la gestión de proyectos o facturación, y es capaz de trabajar con un conjunto reducido de formatos de textos.

Atril DejaVu

DejaVu [web-7] es un entorno completo de asistencia a la traducción, que permite trabajar con múltiples herramientas, como una memoria de traducción, diccionarios y glosarios, e incluye un subsistema de gestión de proyectos. Es un sistema muy poderoso y adaptable que combina las tecnologías de memoria de traducción con las técnicas de traducción automática basada en ejemplos. El principal inconveniente de esta herramienta es que solo está disponible para MS Windows.

Wordfast y Metatexis

Mientras que la mayoría de los sistemas de memorias de traducción funcionan en forma independiente, Wordfast [web-32] y Metatexis [web-8] están diseñados como plugins para Microsoft Word, y otros productos del paquete Microsoft Office. Aunque no son tan completas como otras herramientas especializadas, son productos fáciles de instalar y usar.

Heartsome

Heartsome [web-9] consiste en un conjunto de productos que pueden ser adquiridos de forma independiente o empaquetados para ser utilizados en conjunto. Entre ellos se destacan un editor de traducciones XLIFF, un editor TMX y un gestor de terminología. Provee además soporte para una gran variedad formatos de archivos.

TranSearch

El proyecto TransSearch [web-10] es un intento de construir un sistema de memorias de traducción basado en la World Wide Web [7]. Está basado en el sistema del mismo nombre del laboratorio RALI, de la Universidad de Montreal, el cual es utilizado desde 1996 para permitir al público canadiense acceder a transcripciones en inglés y francés de los debates parlamentarios del país. Otro de sus objetivos es promover la construcción de bases de datos bitextuales para proveer a los traductores de memorias de traducción de libre acceso.

Star Transit

Star Transit [web-11] es uno de los productos que lleva más tiempo en el mercado Posee la mayoría de las características de otros sistemas, tales como gestión de proyectos, soporte de múltiples formatos, incorporación de un módulo de gestión de terminología y capacidad de trabajar con una gran cantidad de idiomas diferentes. Incorpora una etapa de pre-traducción en la que aplica algunas técnicas de traducción automática, pasando posteriormente a la etapa de traducción manual con asistencia de la memoria de traducción.

Virtaal

Virtaal [web-30] es uno de los productos más recientes en el mercado aunque por ser totalmente libre y gratuito está ganando terreno rápidamente. Está diseñado para ser muy fácil de instalar y usar, e intuitivo, de forma de que pueda ser utilizado por traductores no expertos en el campo de la computación. Además, como funcionalidad más relevante se destaca el permitir integrar varias fuentes para obtener traducciones, tales como sistemas servidores de traducciones entre los que se cuenta Open-Tran.

Otros sistemas

Además de los antes mencionados, existen otros sistemas que, o bien no están tan evolucionados como ellos, o han sido abandonados.

Translation Manager [art-1], de IBM, fue uno de los primeros sistemas de asistencia a la traducción por memorias, dominando el mercado durante mucho tiempo, a pesar de su reconocida falta de usabilidad y dificultad de aprendizaje, pero fue abandonado en 2002.

KbabelDict es una sencilla implementación de un editor de memorias de traducción que forma parte del proyecto Kbabel; trabaja exclusivamente con el formato PO, sin soporte para TMX.

TransEditor es un editor XLIFF. Fue construido como demostración de las capacidades de las Open Language Tools (OLT), un conjunto de filtros para el tratamiento de lenguaje natural. No maneja proyectos, usuarios, ni permite llevar estadísticas; sin embargo, es utilizado por muchos traductores en su trabajo diario debido a su simplicidad de uso.

Pootle

Pootle [web-28] es un portal de asistencia a los traductores, aunque está más orientado a la localización de software que a la traducción de documentos. Permite trabajar tanto con XLIFF como con PO. Al ser un portal, toda su interfaz es web, por lo que una vez instalado en un servidor, puede ser accedido desde cualquier parte a través de Internet, lo que lo ha ubicado muy bien en la comunidad Open Source, siendo actualmente la herramienta utilizada para la localización de Mozilla Firefox y OpenOffice [web-29], entre otros productos.

Open-Tran

Open-Tran [web-31] se propone ser un repositorio on-line de traducciones, donde todos los interesados puedan publicar sus propias traducciones y consultar las de los demás, de forma de obtener una consistencia entre los diferentes productos, independientemente de su origen, y acceder a traducciones en diferentes idiomas no tradicionales.

2.7 Estándares

Uno de los grandes inconvenientes que se debe resolver al diseñar un sistema de asistencia a la traducción es la gran heterogeneidad de formatos en que se distribuyen los documentos con los que se trabaja usualmente: HTML, MS Office (doc, xls), OpenOffice y StarOffice (odt, sxw), XML, DocBook, etc. La solución por la que han optado la mayoría de los implementadores es transformar todos los formatos a uno común, estándar y bien definido.

Con ese objetivo, se han desarrollado varias propuestas de formatos, de entre las cuales destacan dos: PO y XLIFF.

2.7.1 El estándar PO

El formato PO [web-12] es el estándar del proyecto GNU; en particular, es el formato con el que trabaja gettext, la principal herramienta GNU de localización de software.

Un archivo PO se divide en bloques, también llamados entradas, cada uno de los cuales contiene información sobre un fragmento de texto en un idioma y su traducción en otro, además de información adicional. Cada archivo admite solo dos idiomas.

Aunque es un formato muy usado para la localización de software, resulta muy limitado al momento de trabajar con documentos más grandes y complejos, por lo que no es muy

frecuente su uso en la traducción de textos en general.

2.7.2 El estándar XLIFF

XLIFF [web-13, web-14, web-15] es el formato más difundido para el intercambio de información de localización. Su objetivo es permitir la conversión de documentos en cualquier formato a un formato común, con el cual puedan trabajar los sistemas de traducción, pero que ofrezca mayor flexibilidad y sea más completo que PO.

Este formato fue desarrollado en colaboración por un gran número de empresas (Sun Microsystems, IBM, Novell, Oracle, entre otras) y puesto bajo la supervisión de un comité dependiente de OASIS. Apunta a tres objetivos fundamentales:

- Separar el texto del formato.
- Permitir a diferentes herramientas trabajar con texto y agregar información sobre el mismo.
- Almacenar información que sea relevante al proceso de localización.

XLIFF define dos tipos de formatos aplicables a los documentos: el *formato en línea* (inline format) y el *formato estructural* (structural format). El formato en línea es parte del texto (por ejemplo, negritas, subrayados, itálicas, etc.), mientras que el formato estructural hace referencia a la estructura del texto (por ejemplo, el tag <BODY> de HTML).

XLIFF almacena el formato estructural, o esqueleto del documento, en un archivo separado, en el cual existen referencias a las unidades traducibles indicando la posición que ocupa cada una dentro del texto. Sin embargo no define ningún formato para dicho archivo, por lo que el software encargado de convertir un documento a XLIFF debe ser capaz de volver a convertir posteriormente de XLIFF al formato original (u otro), ubicando los segmentos traducidos en su lugar dentro del esqueleto.

El formato en línea presenta un inconveniente: no puede ser removido por completo pues es deseable que el documento traducido presente las mismas características que el original, pero no debe ser interpretado como parte del texto, ni debe interferir en la traducción. XLIFF permite utilizar dos mecanismos para tratar el formato en línea: remover las etiquetas del texto original y almacenarlas en el esqueleto, o re-etiquetar el texto marcando las etiquetas de formato con otras propias.

XLIFF está pensando para permitir el proceso de traducción en fases o etapas, pudiendo almacenar información de cada etapa en el encabezado y así registrar el historial de traducción. Además permite anotar las unidades de traducción con información adicional, como por ejemplo el traductor, la fecha, su estado (no traducido, traducido manualmente, aprobado, etc).

XLIFF y PO

Generalmente se suele decir que los archivos PO son la versión arcaica de XLIFF de la era pre-XML. Actualmente hay una tendencia a remplazar el formato PO por XLIFF o en su defecto lograr una abstracción del uso de PO o XLIFF de tal manera que puedan utilizarse indistintamente. Esto último es a lo que apunta el proyecto WordForge (proyecto que trabaja en la localización de software) sosteniendo que la abstracción permitirá que las herramientas sigan siendo útiles a los que trabajan en la localización si continúan usando PO, pero también permitirá la migración al formato XLIFF.

Existen programas conversores de PO a XLIFF, entre los que se destacan XLIFF PO Tools, XLIFF Tools for Java y Translate Toolkit.

2.7.3 El estándar TMX

Las memorias de traducción deben ser almacenadas de alguna manera, tanto para poder ser reutilizadas, como para poder ser transportadas de un sistema a otro. Dada la complejidad y el costo de la alimentación inicial de los sistemas de memorias de traducción, se han diseñado formas de transferir memorias entre sistemas, entre los que se destaca el formato TMX, propuesto inicialmente por Melby en 1998 y que se ha convertido en el estándar de la industria. Actualmente es desarrollado y mantenido por OSCAR.

TMX está basado en XML, definiendo un conjunto reducido de etiquetas para marcar los elementos básicos de una memoria de traducción y respetando los estándares ISO para fechas, números y códigos internacionales. Además está codificado en Unicode de forma de poder ser usado en la más amplia variedad de sistemas e idiomas.

Un archivo TMX consta de dos partes: un *encabezado* (header) y un *cuerpo* (body). El encabezado se utiliza para representar información sobre el archivo y su creación, tal como la herramienta que la generó y el tipo de segmentación usada, así como información particular de las herramientas que lo usan. En el cuerpo del documento se encuentran todas las unidades de traducción junto con sus respectivas traducciones. TMX no hace distinción entre idioma original y otros idiomas, por lo que el archivo puede ser usado para la asistencia a la traducción entre cualquier par de idiomas que esté representado en él.

2.7.4 El estándar UMTF

UMTF compite con TMX como formato de almacenamiento de memorias de traducción y también está basado en XML, con la capacidad de ser almacenado en forma comprimida. Fue desarrollado por Fatih Demir específicamente para Gtranslator, una herramienta ampliamente utilizada para la internacionalización en el proyecto Gnome. Por el momento no existe un proceso de desarrollo formal o estándar ni una asociación o comisión para la toma de decisiones sino que se desarrolla según surgen los requerimientos.

2.8 Herramientas

Además de los sistemas dedicados especialmente a la traducción, existe una gran cantidad de software que es ampliamente utilizado para el análisis de lenguaje natural y que, por tanto, es de mucha utilidad en el proceso de diseño y construcción de este tipo de sistemas. A continuación describimos algunas de las que nosotros consideramos más relevantes.

Freeling

Freeling [web-16] es una herramienta que provee servicios de análisis del lenguaje, distribuida bajo la licencia GPL. Aunque está diseñada como una biblioteca para ser integrada a aplicaciones, provee también un sencillo programa ejecutable que permite utilizar todas sus capacidades desde una línea de comandos. A diferencia de la mayoría de las herramientas de procesamiento de lenguaje natural, enfocadas principalmente en el idioma inglés, Freeling hace una apuesta fuerte por el idioma español y el catalán, además de inglés, gallego e italiano. Ejecuta únicamente en entornos Unix y Linux, aunque se ha logrado hacerla funcionar en entornos MS Windows con la ayuda del programa CygWin, un pseudoemulador de Linux [web-44]. Hasta la versión anterior solo proveía un API para C++ pero a partir de la versión 2.0 incluye también un API para Java.

Entre las principales funcionalidades que ofrece Freeling, se encuentran la tokenización de textos, la segmentación, el análisis morfológico, el tratamiento de sufijos, la

lematización, la separación de contracciones, la detección de entidades con nombre, fechas, números, magnitudes y el PoS tagging. Para el PoS tagging utiliza el conjunto de etiquetas Eagles para el idioma español y Penn TreeBank [web-46] para inglés; aunque detectamos casos de signos de puntuación en el idioma inglés en los que usa las etiquetas Eagles.

OpenNLP

OpenNLP [web-17] es el resultado de la unificación de varios proyectos relacionados al procesamiento de lenguaje natural. Su principal objetivo es facilitar el desarrollo y la investigación en el área del procesamiento de lenguaje natural. Por el momento está enfocado en los idiomas inglés, español y alemán.

Entre los proyectos comprendidos se encuentran módulos que permiten realizar la segmentación de textos, la tokenización, el PoS tagging, la detección de entidades con nombre y la lematización de términos. Para el PoS tagging utiliza el conjunto de etiquetas Eagles para el idioma español y Penn TreeBank [web-46] para inglés.

Se distribuye como un API para Java y un conjunto de modelos (datos compilados, que incluyen diccionarios, glosarios, reglas, etc) pensados para realizar un trabajo particular sobre el texto. Para algunas tareas requiere de la disponibilidad de un diccionario similar a WordNet. Tal vez no se encuentre tan avanzado como Freeling pero provee integración directa a programas Java desde su concepción.

UIMA

UIMA [web-18] es una arquitectura para el desarrollo, descubrimiento y composición de componentes especializados en el análisis de información no estructurada, capaces de realizar análisis semántico y búsquedas. Originalmente fue desarrollada por IBM para luego ser lanzada como proyecto Open Source en el marco de la Apache Software Foundation y adoptada por un grupo especial de OASIS (UIMA TC). La arquitectura de componentes propone una muy marcada descomposición de los sistemas, produciendo unidades funcionales especializadas en alguna actividad particular y capaces de interaccionar entre sí.

La plataforma UIMA incluye un SDK para facilitar el desarrollo de componentes en lenguaje Java y C++, y un plugin para el entorno de desarrollo Eclipse. El objetivo es asistir en el desarrollo de aplicaciones capaces de procesar grandes volúmenes de información no estructurada para descubrir y capturar información relevante para un usuario final. La teoría básica es sencilla: una aplicación se compone de un conjunto de componentes que exponen interfaces para otros componentes, que luego se encadenan de alguna manera de forma que la salida de uno se transforma en la entrada de otro.

File2XLIFF4J

File2Xlifff4J [web-19] es un conjunto de filtros que permiten convertir desde varios formatos a Xliff y viceversa. Permite trabajar con archivos en formato ODF, HTML, properties, PO, PDF, FrameMaker (MIF) y varios formatos propios de Microsoft, tales como DOC, PPT y RTF. Se distribuye en forma de un API para Java, de forma que puede ser integrado a cualquier aplicación en desarrollo que necesite transformar documentos al estándar XLIFF y viceversa.

Open Language Tools (OLT)

Open Language Tools [web-20] es un conjunto de herramientas de asistencia a la traducción liberadas por Sun Microsystems como un proyecto Open Source, aparentemente abandonado a principios de 2007. El objetivo del proyecto es ayudar a los desarrolladores de software de asistencia a la traducción utilizando para ello estándares

tales como XLIFF y TMX y trabajando con múltiples formatos de archivos.

Uno de los principales aportes de OLT es su conjunto de filtros para XLIFF. Además, provee algunas otras herramientas, incluido un editor de archivos XLIFF, llamado TransEditor. Si bien no es el objetivo del proyecto Open Language Tools construir un sistema completo, las herramientas para usuarios finales son bastante completas para los requerimientos mínimos de un traductor.

Transolution

Es una suite similar a Open Language Tools pero implementada en Python, con pocas capacidades y aparentemente abandonado desde 2006. Puede trabajar con un conjunto reducido de formatos, no tiene soporte completo para XLIFF, pero, a diferencia de OLT, no requiere la codificación de nuevos filtros en un lenguaje de programación, sino que se definen mediante archivos de configuración.

TMXEditor

TMXEditor [web-21] es un sencillo editor de archivos TMX desarrollado por el departamento de impuestos del estado de Vermont, en EEUU y liberado bajo una licencia Open Source. Una de sus principales características es que permite añadir tantos idiomas como sea necesario. Además, es muy fácil de utilizar, lo que lo convierte en una herramienta para traducciones rápidas. No es más que un editor de archivos TMX, por lo que no provee capacidades relativas a las memorias de traducción.

Una característica a destacar respecto de esta herramienta es que el código fuente Java está disponible en su totalidad, lo que permite agregar las capacidades que provee a cualquier otro software desarrollado en el mismo lenguaje.

Snowball

Snowball [web-22] es una librería de funciones tanto para C como Java, que permiten obtener raíces de palabras en una gran variedad de idiomas. Además, define un lenguaje propio, que lleva el mismo nombre, mediante el cual se pueden implementar nuevos algoritmos para obtener raíces. Es precisamente usando este lenguaje que provee las implementaciones para los algoritmos en cada uno de los idiomas soportados.

XLIFF Tools

XLIFF Tools es un proyecto comenzado en 2005 con el objetivo de fomentar el uso de XLIFF en los procesos de localización de proyectos Open Source. Se trata únicamente de conversores de PO a XLIFF y viceversa, implementados tanto en Java (XLIFF Tools Java), C (XLIFF Tool) como Python (Translate Toolkit).

WordNet

WordNet [web-23] es una base de datos de información léxica, centrada exclusivamente en el idioma inglés. La idea tras el proyecto es agrupar diferentes categorías gramaticales, tales como sustantivos, verbos, adjetivos y adverbios, en conjuntos de sinónimos cognitivos, llamados synsets (synonyms sets, conjuntos de sinónimos). De esto, resulta una red completa de términos relacionados que pueden ser navegados lógicamente en base a su significado. En particular, WordNet solo comprende las categorías gramaticales abiertas, descartando las cerradas (determinantes, preposiciones, pronombres, conjunciones y participios).

Está disponible para ser descargado libremente desde su web, en versiones tanto para MS Windows como Unix. Normalmente, WordNet se distribuye en formato ASCII plano, junto a un conjunto de aplicaciones, pero también provee una interfaz para el lenguaje C

y para muchos otros lenguajes, incluido Java (a través de proyectos independientes), incluyendo una interfaz SQL.

SecondString

SecondString [web-27] es una librería para Java de funciones de comparación difusa utilizando diferentes algoritmos. Fue desarrollada como proyecto Open Source y liberada en 2006. Implementa diferentes algoritmos de comparación, como Jaccard, Jaro, Jaro-Winkler, Levenshtein, Monge-Elkan, y muchos otros.

2.9 Conclusiones

Los sistemas de asistencia a la traducción son adecuados para textos técnicos o con alto nivel de repetición. Es relativamente viable construir traductores para áreas temáticas acotadas, pero dicha viabilidad se reduce a medida que se intenta abordar más temáticas. De todas maneras, casi ninguna técnica de traducción automática o asistida por computadora es excepcionalmente útil para realizar la traducción de textos creativos o expresivos.

Algunas de las características en común encontradas en los sistemas de memorias de traducción investigados son el uso de estándares para la representación de información, el soporte para los formatos de archivos más comunes y el soporte para la utilización de varias memorias simultáneamente.

El estándar más difundido para el almacenamiento de memorias de traducción es TMX siendo la mayor ventaja de su uso la posibilidad de transferir memorias entre sistemas diferentes.

Con respecto a los estándares de los archivos a procesar, encontramos que XLIFF es el más difundido. Consideramos que su utilización es una buena opción no solo por la flexibilidad de trabajar con un formato común, sino porque permite la separación del texto y el formato, y además existen ya múltiples herramientas que permiten convertir de diferentes formatos a XLIFF y viceversa.

3 Solución propuesta

3.1 Introducción

En este capítulo describimos la solución propuesta comenzando por un análisis de los requerimientos del sistema, la definición de las funcionalidades que debe presentar y el diseño que realizamos para implementarlas.

3.2 Descripción del sistema

La principal tarea que debe desempeñar un sistema de asistencia a la traducción es facilitar a las personas el trabajo de traducción de documentos. Entendemos por documento cualquier texto que tenga valor para las personas. En general, el proceso de traducción de un documento utilizando un sistema de asistencia a la traducción comienza por la carga del documento en el mismo, atraviesa una serie de etapas, algunas de las cuales requieren participación humana y concluye con el documento traducido, en lo posible en un formato similar.

Los sistemas relevados presentan más o menos las mismas características, algunas de las cuales consideramos fundamentales para un sistema de asistencia a la traducción por memorias, otras consideramos muy importantes y deseables y un tercer grupo pueden ser consideradas adicionales. De igual manera, los proyectos relevados también presentan características similares, y utilizan técnicas y herramientas comunes.

3.2.1 Funcionalidades básicas

A continuación detallamos las funcionalidades básicas que consideramos que nuestro sistema debe presentar.

Asistencia a la traducción

El sistema debe dividir los documentos en fragmentos de texto y proponer candidatos para su traducción. Para proponer estos candidatos debe utilizar memorias de traducción en las que realiza búsquedas difusas.

Trabajo con memorias

El sistema debe permitir importar memorias de traducción de forma que cuando los usuarios utilicen el sistema ya existan traducciones que puedan ser útiles. Además, una vez en el sistema estas memorias importadas deben ampliarse con cada nueva traducción. También debe permitir exportar las memorias a archivos para poder ser importadas a otros sistemas.

Etapas del proceso de traducción

Dividimos el proceso en seis etapas: *carga, evaluación, preprocesamiento, traducción, revisión y exportación*. Esto es lo que hemos denominado *flujo del documento*, o *workflow*. Incluso no necesariamente debe darse en la secuencia indicada sino que existe la posibilidad de *volver atrás* y pasar varias veces por la misma etapa. Este workflow se muestra en la siguiente imagen:

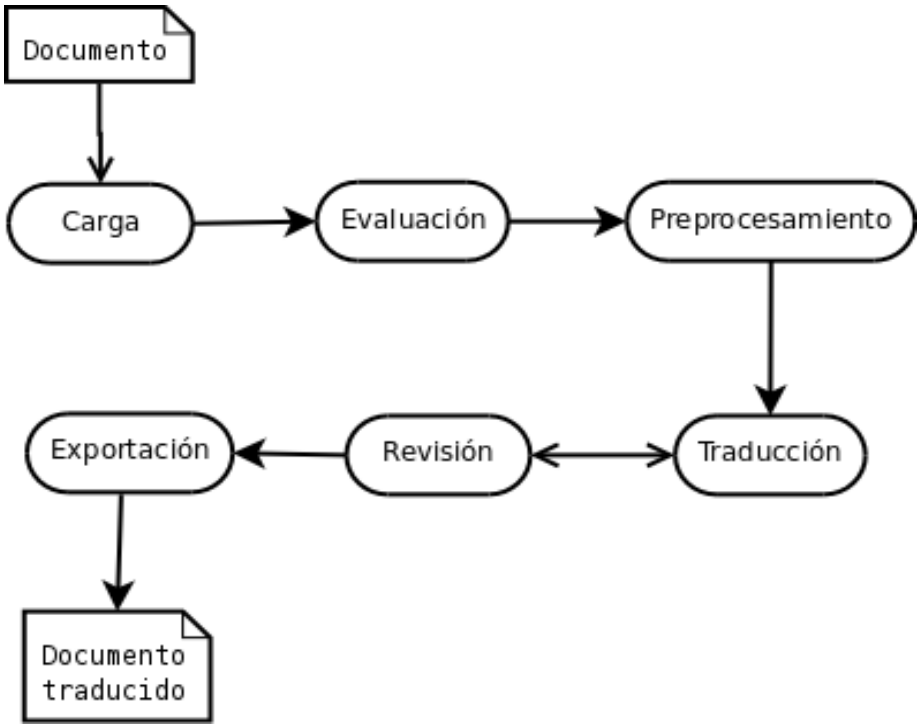


Figura 3.1 Workflow de documentos

Carga de documento

La primer etapa consiste en la importación de un documento al sistema. Consideramos que deben soportarse al menos los siguientes formatos de archivos: Microsoft Word, OpenOffice Writer, HTML y texto plano. La razón para esto es que los mencionados son los formatos más usuales para el tratamiento de documentos en la actualidad. Adicionalmente, siguiendo uno de los requerimientos del proyecto, agregamos Framemaker (MIF) y XML.

Evaluación

La segunda etapa consiste en analizar el documento tal cual fue cargado al sistema a los efectos de determinar si está en condiciones de ser traducido o no. Las razones por las que un documento puede no ser traducible escapa al sistema sino que es decisión de la persona que realiza el trabajo, por lo que el sistema solo se limita a ofrecerle al usuario la capacidad de observar el documento y a continuación indicar si el mismo puede ser pasado a traducción o no.

Preprocesamiento

La tercer etapa en la que hemos dividido el proceso es el preprocesamiento del documento. En ésta se remueven las partes que no son traducibles, incluyendo las imágenes y la información adicional que afecta a la estructura del documento, como las marcas de formato. Por ejemplo en el caso de las páginas web se remueven las etiquetas HTML. Además, es en esta etapa en que el texto que sí es traducible se divide en fragmentos de menor tamaño. Optamos por realizar la segmentación por oraciones, ya que entendimos que las unidades más grandes, como los párrafos, además de ser más costosas de traducir para las personas (mayor número de palabras, conceptos e ideas), reducen las posibilidades de encontrar textos similares ya traducidos en la memoria del sistema; por otra parte, las unidades más chicas tienden a hacer poco clara la semántica del texto (palabras sueltas fuera de contexto), siendo también difícil de traducir.

Traducción

La cuarta etapa es la más importante del proceso. En ésta se le presenta al usuario cada uno de los segmentos en que se dividió el texto original, junto con una lista de traducciones candidatas tomadas de las traducciones de otros textos más o menos similares, cada una de ellas acompañada de su medida de similaridad con el segmento a traducir. Para la similaridad utilizamos una métrica porcentual, en la cual un valor de 100 indica que el texto original de la traducción es idéntico al segmento a traducir. En base a esta métrica, el sistema presenta las traducciones candidatas en orden de similaridad, pero solo aquellas que superen un cierto mínimo, por ejemplo, 65%.

Revisión

Cuando todos los segmentos del documento han sido traducidos, el sistema ofrece al usuario la posibilidad de revisar el trabajo hecho, pudiendo generar dos resultados: o bien se aprueba la traducción, o bien se solicita que se vuelva a trabajar sobre el documento. En esta etapa, el usuario puede analizar el trabajo completo y tomar la decisión.

Exportación

Finalmente, la última etapa del proceso es la exportación o descarga del documento. Esta etapa comprende también volver a colocar los elementos no traducibles que fueron removidos en la etapa de preprocesamiento para obtener un documento con un formato lo más similar posible al original, pero cuyo texto se encuentra en otro idioma.

Asignación de documentos a usuarios

Para fortalecer aún más el flujo, introdujimos en el mismo el concepto de asignación. Un documento está, o bien asignado a un usuario y solo ese usuario puede trabajar con él, o bien no asignado y por lo tanto ninguna tarea puede hacerse con él más que asignarlo a un usuario o eliminarlo del sistema. Así, antes de entrar en cada etapa existe un paso previo que es la asignación del documento a un usuario para que la lleve a cabo.

Usuarios, roles y permisos

Aunque el proceso es uno solo, existen actividades que deberían ser realizadas por usuarios diferentes. Por ejemplo, el usuario que realiza la traducción y el que la evalúa deberían ser distintas personas. Es así que decidimos introducir los conceptos de usuarios, roles y permisos.

En general, un usuario es una persona que puede usar el sistema. Sin embargo, restringimos esta definición, transformándola en *un usuario es una persona que puede hacer uso de algunas funcionalidades del sistema*. De esta forma resaltamos la idea de que no todas las funcionalidades están disponibles para todos los usuarios. Las funcionalidades que están disponibles para un usuario dado están dadas por los roles que tenga asignados. Un rol engloba un conjunto de permisos sobre el sistema, indicando cuáles son las funcionalidades a las que puede acceder cada usuario.

Establecimos cinco roles:

- Asignador: habilita al usuario que lo posee a asignar documentos que no estén asignados a algún usuario adecuado a la tarea.
- Evaluador: habilita al usuario que lo posee a analizar los documentos recientemente cargados en el sistema para determinar si ameritan ser traducidos o no.
- Traductor: habilita a los usuarios que lo poseen a traducir documentos.
- Revisor: habilita a los usuarios que lo poseen a revisar los documentos traducidos por los usuarios traductores, para determinar si su trabajo es adecuado

o si debe volver a trabajar sobre ellos.

- Administrador: habilita efectivamente a hacer uso de todas las funcionalidades. Solo debería ser otorgado a unos pocos usuarios, aunque al menos uno debe tener este rol, ya que existen tareas que ningún otro rol permite desempeñar, tales como la configuración del sistema.

Clientes y proyectos

Los documentos cargados en el sistema tienen un origen, es decir, son provistos por alguien. Este alguien, sea una persona o una organización, es lo que denominamos cliente del documento. A su vez, los documentos se agrupan en proyectos, de forma tal que un cliente puede tener varios proyectos, y un proyecto puede contener varios documentos.

Memoria a utilizar en la traducción

Debido a que las memorias de traducción son un bien organizacional normalmente no son compartidas entre clientes. Por esta razón nuestro sistema debe permitir seleccionar una memoria de traducción con la que se ha de trabajar en un documento teniendo en cuenta el proyecto y el cliente al que pertenece el mismo. Para esto decidimos permitir asociar las memorias de traducción a clientes y proyectos, aunque no a documentos; al momento de importar una memoria, se debe indicar a cual cliente corresponde, y de esta manera toda memoria siempre queda asociada a un cliente; de forma similar, cuando se comienza un nuevo proyecto, el cual debe estar asociado a un cliente, se debe indicar cuál de todas las memorias del cliente se desea usar para el proyecto. Al cargar un documento al sistema, indicando el cliente y el proyecto, el sistema determina cual memoria usar.

Existe la posibilidad de que un cliente no posea una memoria propia para entregar junto con sus documentos. Para estos casos es que consideramos una memoria especial, compartida por todos los clientes (la única posible) a la que llamamos Memoria Global. Esta memoria también puede ser usada en proyectos de clientes que de hecho tengan una o más memorias propias.

En nuestro sistema decidimos no permitir trabajar con más de una memoria en un mismo proyecto, y por tanto en un documento.

Habilidades de usuarios

Es deseable que los documentos sean asignados a los usuarios que estén más capacitados para trabajar con ellos, según la temática, los lenguajes involucrados, etc. Para resolver esto, introdujimos el concepto de habilidad. Cada documento puede tener asociadas un conjunto de habilidades que requiere, mientras que los usuarios, especialmente los que tienen el rol de traductor, tienen asociado un conjunto de habilidades que ellos mismos poseen. En base a esto, un sistema podría ser capaz de seleccionar el usuario más capacitado para un documento dado, simplemente comparando las habilidades requeridas por el documento contra las habilidades de cada uno de los usuarios. Aunque nuestro sistema no proporciona esta funcionalidad, pone a disposición del asignador la información necesaria para que éste tome la decisión.

Preferencias de estilo de traducción

Las preferencias de estilo son un conjunto de pautas que se deberían seguir en la redacción o traducción de documentos. Por ejemplo, un estilo puede marcar que no se debe utilizar voz pasiva en los documentos y/o que las locuciones extranjeras deben ser mantenidas en el idioma original. La definición de estilos solo se trata de recomendaciones. Aunque existen estudios en el área del procesamiento de lenguaje

natural relacionados a esto, consideramos que perseguir el mantenimiento de estilos automáticamente está fuera del alcance de este proyecto y que por tanto corresponde al traductor el respeto a los estilos del documento.

3.2.2 Otras funcionalidades

En la sección anterior mencionamos las funcionalidades básicas que tras nuestra investigación destacamos como fundamentales, sin las cuales nuestro sistema no cumpliría las mínimas expectativas. Pero existen otras funcionalidades o características que son deseables, algunas de las cuales implementamos.

Gestión de usuarios, roles, clientes y proyectos

La administración del sistema incluye tareas tales como la creación de usuarios, indicando el o los roles asociados, la creación de roles, el registro de clientes y proyectos, y la carga de documentos.

Facturación a usuarios y clientes

El sistema podría llevar un registro de la cantidad de trabajo efectuado por cada usuario, considerando tiempos, extensión del texto traducido, cantidad de veces que se necesitó volver a trabajar una vez que finalizó un documento, etc. De esta manera, podría calcularse el monto que habría que abonar a cada uno. Es muy común que las organizaciones no posean una plantilla propia de traductores a los que se les paga un salario mensual, sino que los contratan según las necesidades y se les remunera de acuerdo al trabajo realizado. Para esto resultaría muy útil que el sistema fuese capaz de realizar los cálculos y presentar la información.

En lo que se refiere a los clientes, el sistema podría calcular el trabajo realizado para cada cliente, considerando el esfuerzo en tiempo y usuarios dedicados a ellos, y en base a eso emitir informes que permitan realizar la facturación.

Seguimiento de usuarios y clientes

El sistema podría proveer capacidades de gestión en lo que tiene que ver con el rendimiento de los usuarios y los requerimientos de los clientes, de forma de ayudar a las personas que toman decisiones, a determinar si existen usuarios del sistema que no satisfacen los requerimientos o por el contrario son subaprovechados, y de igual manera determinar si hay clientes que suelen presentar requerimientos especiales que ameriten que sean considerados con mayor prioridad o todo lo contrario. Esto también incluye el establecimiento de plazos y contralor de los mismos.

Notificaciones y alertas

Durante el proceso de traducción de un documento en el sistema es común que se presenten situaciones que requieran la atención de los usuarios, y quizás también la ejecución de una acción como consecuencia. Es así que introdujimos el concepto de notificaciones, las cuales podrían ser usadas para dar conocimiento al usuario de una situación excepcional, como por ejemplo la permanencia de un documento durante un tiempo excesivo en un mismo estado, así como notificar de cambios en características o propiedades de los mismos que podrían necesitar una acción inmediata.

Históricos de actividad sobre documentos

Una característica muy deseable para casi cualquier sistema informático es la posibilidad de mantener históricos de actividad y permitir obtener informes de ellos. En el caso de nuestro sistema se mantienen históricos de los documentos encargados por los

diferentes clientes, los trabajos hechos por los usuarios y los tiempos consumidos. Con una presentación adecuada estos históricos podrían permitir tomar decisiones a los administradores (por ejemplo, determinar la eficiencia de los traductores, o las dificultades de los documentos de ciertos clientes).

Complementos

Otra característica deseable es que el sistema brinde al usuario otro tipo de asistencia, tal como permitir buscar términos en diccionarios y enciclopedias on-line, como WordReference y Wikipedia, así como en buscadores de Internet, como Google.

3.3 Requerimientos relevados

A continuación se presenta el análisis realizado con respecto a los requerimientos incluidos dentro del alcance de la implementación a entregar.

Gestión de clientes

Se debe proporcionar la capacidad de registrar clientes que solicitan la traducción de un documento por primera vez, modificar clientes ya registrados, dar de baja clientes de forma de que no puedan aceptarse más trabajos de los mismos, y consultar los clientes registrados.

Gestión de proyectos

El sistema debe permitir añadir nuevos proyectos, modificar y dar de baja proyectos existentes, y consultar los proyectos registrados. Al momento de ingresar un proyecto en el sistema, se debe asociar al cliente correspondiente. Este cliente debe existir previamente en el sistema. No se requiere que se pueda cambiar la asociación de un proyecto de un cliente a otro. Se debe seleccionar también la memoria que se desea usar para la traducción de los documentos pertenecientes al proyecto. Las memorias disponibles para elegir son todas las del cliente más la memoria global. Las preferencias de estilo que se desea sean seguidas durante el trabajo con los documentos pertenecientes al proyecto se definen de manera textual.

Gestión de documentos

Se requiere poder trabajar con al menos los idiomas inglés y español y soporte para los formatos MIF (Maker Interchange Format), ODT (Open Document Format), .properties, DOC (Microsoft Word), HTML y XML. El sistema debe permitir ingresar nuevos documentos, modificar y eliminar documentos existentes, y realizar consultas sobre los mismos. Al momento de cargar un documento se debe asociar a un proyecto existente del cliente seleccionado como propietario del mismo. Se configura además una prioridad, la cual es un número entero entre 1 y 10 que indica la urgencia o importancia de la traducción del documento, siendo el número 1 la mínima prioridad y el 10 la máxima prioridad. Se debe poder asociar habilidades a los documentos al momento de cargarlos; estas habilidades indican cualidades y/o capacidades que son deseables que posea el usuario traductor que va a trabajar con el mismo.

Gestión de roles

El sistema debe permitir añadir nuevos roles, modificar roles existentes, y consultar los roles registrados. Al momento de dar de alta un nuevo rol, se le debe asociar uno o varios permisos que definen las actividades que podrán realizar los usuarios poseedores de ese rol.

Gestión de usuarios

El sistema debe permitir registrar usuarios (brindando un nombre de usuario y una contraseña como mínimo), editarlos y consultarlos. Los usuarios deben tener asociado uno o más roles. Además se debe poder configurar las habilidades que éstos poseen, seleccionando de un conjunto de habilidades predefinidas, y asociar además un puntaje para cada habilidad elegida que indicará el grado de idoneidad. Se debe poder deshabilitar temporalmente a los usuarios sin necesidad de eliminarlo completamente.

Gestión de memorias

El sistema debe permitir cargar memorias desde archivos indicando el cliente propietario. Ninguna memoria podrá estar asociada a dos clientes diferentes, salvo la memoria común a todos. Además, debe permitir descargar las memorias a archivos.

Proceso de traducción

Todo documento ingresado al sistema debe ser asignado a un usuario con el rol apropiado para evaluarlo, y determinar si el mismo es aceptado o no. Durante la etapa de evaluación, el usuario asignado al documento debe examinarlo y determinar si el mismo efectivamente puede ser traducido. Como resultado de esta etapa el documento puede quedar evaluado y pendiente de asignación a un traductor, o rechazado. Los documentos pendientes de asignación para traducción quedan a la espera de que un usuario con el rol apropiado los asigne a un traductor con las habilidades requeridas.

En la etapa de traducción el sistema seleccionará la memoria de traducción a utilizar en función del proyecto o del cliente. Al traductor se le presentará cada uno de los segmentos de texto que componen el documento, con una serie de posibles traducciones según el resultado de un proceso de selección de alternativas en base a lo registrado en la memoria de traducción seleccionada; estas alternativas estarán ponderadas según algún algoritmo de similaridad y ordenadas según dicho valor. El traductor seleccionará una de las candidatas, o ingresará textualmente la traducción. Cuando todos los segmentos están traducidos, el traductor dará por finalizada la traducción del mismo, y éste pasará a la siguiente etapa. Alternativamente, el usuario traductor podría liberar el documento antes de completar la traducción, volviéndolo al paso anterior, para que sea asignado a otro usuario.

Los documentos que han finalizado el proceso de traducción quedan a la espera de ser asignados a un usuario con roles apropiados para que los revise. El usuario asignado para revisión debe evaluar si la traducción se ajusta a los estándares de calidad previamente definidos, o si, en cambio, debe volver a ser asignado a un traductor. En caso de aceptación del documento, las nuevas entradas ingresadas a la memoria de trabajo durante el proceso de traducción podrán ser aceptadas y así pasar definitivamente a formar parte de la memoria de trabajo seleccionada.

A continuación se muestra gráficamente el proceso descrito:

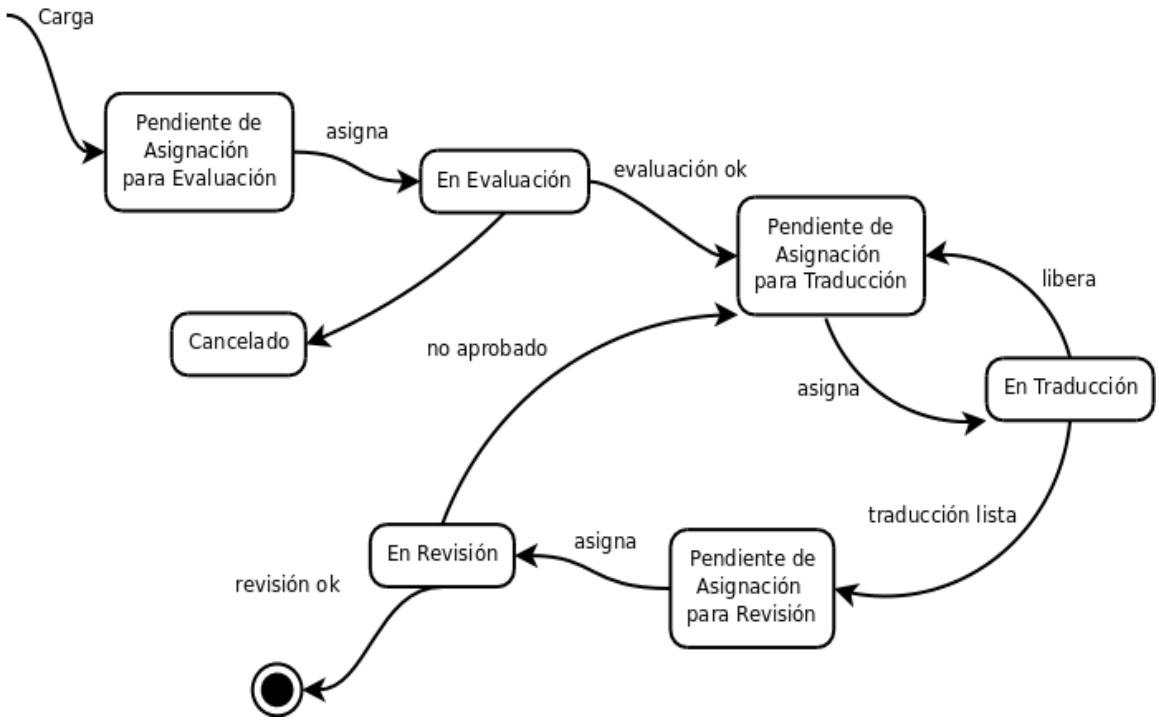


Figura 3.2 Diagrama de estados de los documentos

Listados de documentos

Los usuarios podrán consultar los documentos que tienen asignados. Este listado desplegará la información básica de los mismos, incluyendo nombre, proyecto, código, además de otra información relevante como ser el grado de avance de la traducción del documento, y la actividad que el usuario que lo tiene asignado debe realizar sobre el mismo. Las tareas definidas son: asignar, evaluar, traducir y revisar.

El sistema permitirá acceder al listado de documentos activos donde se desplegarán aquellos que no hayan sido cerrados y permitirá su descarga.

3.3.1 Requerimientos no funcionales

Los primeros requerimientos no funcionales presentados están relacionados a la tecnología, ya que desde un primer momento se nos planteó que debíamos usar la tecnología JavaEE, empleando el lenguaje de programación Java y el servidor de aplicaciones Glassfish. Adicionalmente, se nos planteó que usáramos el entorno de desarrollo integrado Netbeans de forma que el cliente pueda continuar desarrollando el producto. La interfaz del sistema para los usuarios debe ser web, accediendo a la aplicación a través de cualquier navegador web, como Internet Explorer o Mozilla Firefox; la misma debe ser implementada usando la tecnología Java Server Faces (JSF), en particular con las librerías Woodstock para la construcción de interfaces web.

La persistencia de la información debe ser hecha usando MySQL (versión 5).

Optamos por asumir que se debe dar preferencia a las herramientas OpenSource, por lo que tomamos como requerimiento no funcional tal condición.

3.4 Diseño

A continuación en esta sección explicamos los aspectos mas importantes del diseño realizado, describimos los casos de uso relevantes a la arquitectura y el tipo de

arquitectura usada mencionando las capas y subsistemas definidos; por más detalles recurrir al apéndice [ap-diseño-arquitectura] donde se encuentra todo el proceso de diseño.

Identificamos cuatro casos de uso relevantes a la arquitectura:

- *Login:* es el ingreso al sistema mediante la especificación del usuario y contraseña. El sistema determina los roles asociados al usuario y carga sus permisos; en base a éstos despliega el menú con las funcionalidades disponibles. La relevancia de este caso de uso se debe a que define las entidades Usuario, Rol y Permiso.
- *Carga de documento:* es un caso de uso relevante a la arquitectura porque define algunas de las entidades más importantes del sistema: Documento, Proyecto, Cliente, Estilo, Habilidad. Involucra el ingreso de un nuevo documento al sistema mediante la especificación de la ruta al mismo, el cliente propietario, el proyecto al cual está asociado y algunos otros datos de gestión. Se puede indicar también las habilidades que requiere el documento para ser traducido. El sistema realiza una copia del archivo a una ubicación en el servidor y éste pasa a formar parte del conjunto de documentos a traducir.
- *Asignación de documento para traducción:* es relevante a la arquitectura debido a que involucra a las entidades Documento, Usuario, Habilidad y Notificación. Comprende la asignación de un documento previamente evaluado a un usuario con el rol adecuado para la traducción. Al realizar una asignación, el sistema notificará al usuario involucrado en la misma del evento. Esta notificación se realizará por medio de mensajes, como por ejemplo correo electrónico.
- *Traducción: e*ste caso de uso es relevante porque introduce las principales entidades que forman parte del sistema: Documento, Segmento, Unidad de Traducción, Memoria y Complementos. Consiste en la traducción de un documento por parte del usuario al que le fue asignado.

3.4.1 Arquitectura en capas

Decidimos usar una arquitectura en cuatro capas, donde la primera es la capa de presentación, luego se encuentra la capa de seguridad, por debajo de ésta la capa de lógica y por último la capa de acceso a datos.

Capa de presentación

La capa de presentación es la que permite que los actores interaccionen con el sistema, proveyendo la interfaz apropiada según las características del propio actor. Esta capa está conformada por un conjunto de páginas web dinámicas, que son las encargadas de implementar las vistas para el usuario, y una sección de lógica de presentación que se encarga del orden en que se van mostrando las distintas páginas y el manejo de datos básicos para la presentación.

Capa de seguridad

La capa de seguridad se encuentra entre la capa de lógica y la de presentación de manera de autenticar y controlar para todas las llamadas hechas desde la presentación que se tengan los privilegios necesarios según los roles del usuario.

Capa de lógica

La capa de lógica es donde se implementa la lógica del negocio de la aplicación. Para esto provee interfaces que permiten invocar la lógica necesaria para realizar las tareas asociadas a los casos de uso. También se encarga del workflow que determina el camino de los documentos dentro del sistema.

Capa de datos

En la capa de datos se encuentran las funcionalidades asociadas a la persistencia de datos, ya sea en una base de datos como en sistema de archivos (para el caso de los documentos ingresados al sistema y la exportación de los mismos).

3.4.2 Descomposición en subsistemas

Definimos a su vez una descomposición en subsistemas, donde cada subsistema publica una interfaz mediante la cual brinda los servicios a ser utilizados por los otros. Esta descomposición se muestra en el siguiente diagrama:

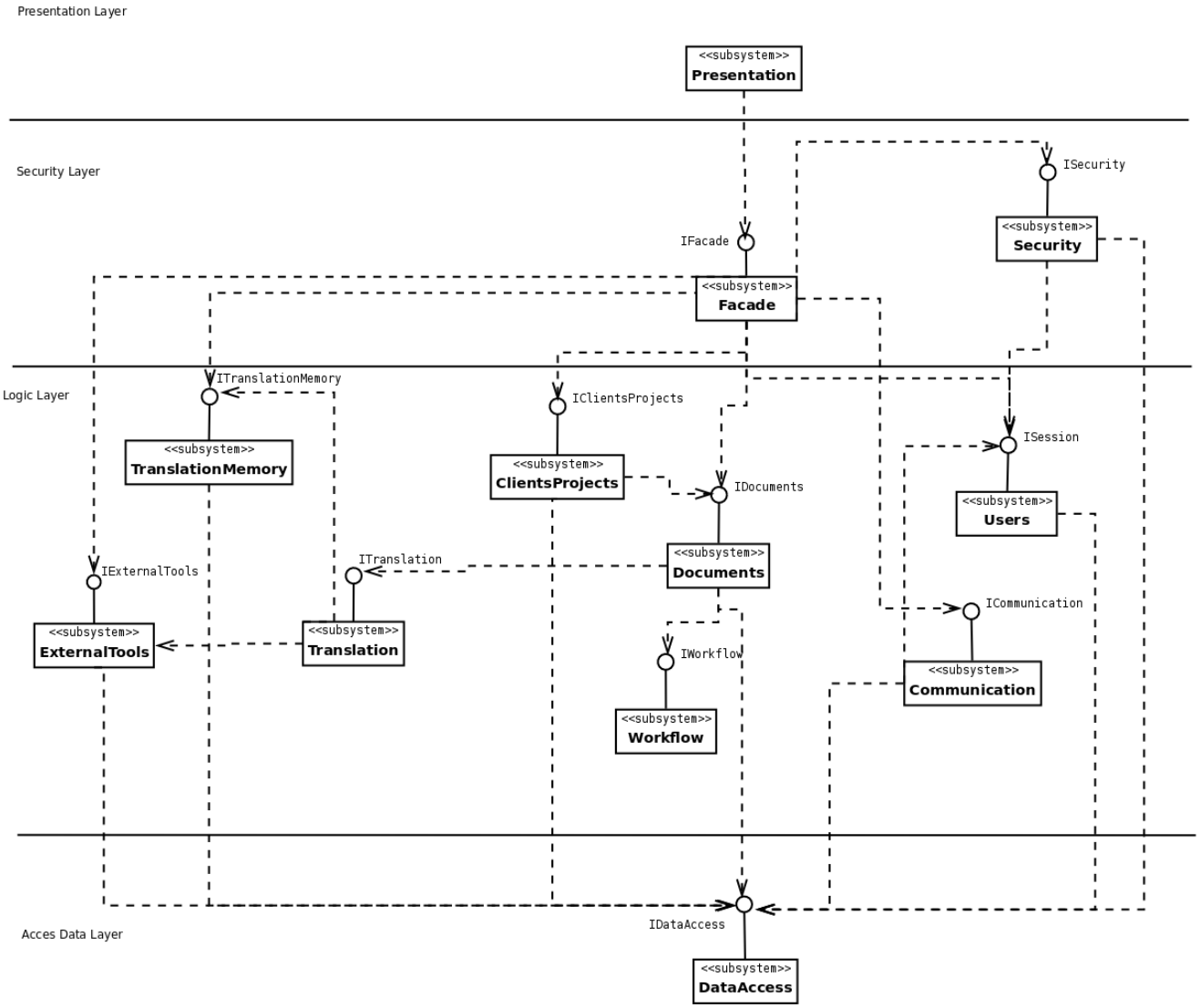


Figura 3.3 Descomposición en subsistemas

- *Presentación:* es el subsistema encargado de realizar la interacción con el usuario final, la cual se realiza por medio de páginas web dinámicas. Para esto utilizamos la tecnología Java Server Faces (JSF).
- *Seguridad:* se encarga de controlar el acceso a las funcionalidades provistas por el sistema, realizando verificaciones de autorización sobre las solicitudes según el usuario que lo intenta. Es responsable de indicar si un usuario puede acceder al sistema, si puede ejecutar una acción determinada y de indicarle al subsistema de presentación cuáles son las opciones del menú que debe mostrarle al usuario.
- *Fachada:* es el encargado de realizar las veces de punto de entrada a la lógica

del sistema, recibiendo todas las solicitudes de invocación de funcionalidades del sistema y despachándolas a los subsistemas que corresponda, previa consulta al subsistema de seguridad para determinar si el usuario está autorizado.

- *Documentos:* se encarga de realizar tareas con documentos, tales como registrarlos en el sistema, segmentarlos y gestionarles el workflow.
- *Traducción:* es el subsistema que tiene las tareas más intensivas, ya que entre ellas se encuentra la búsqueda y calificación de unidades de traducción candidatas, el registro de las elecciones realizadas por los usuarios traductores, y el tratamiento de unidades de traducción en general. Tiene una importante interacción con el subsistema de Memorias de Traducción.
- *Memorias de Traducción:* es el subsistema responsable del tratamiento de las memorias de traducción, permitiendo importarlas, consultarlas, actualizarlas, exportarlas y eliminarlas del sistema.
- *Workflow:* tiene a su cargo las tareas de administración de estados por los que pasa un documento luego de ser importado en el sistema.
- *Usuarios:* provee las funcionalidades para el manejo de usuarios, permitiendo dar de alta usuarios, asignarle roles y habilitar o deshabilitar el acceso al sistema. Además, también tiene a su cargo el tratamiento de información adicional sobre los usuarios, tal como sus habilidades.
- *Clientes y Proyectos:* es el subsistema que provee las funcionalidades para el tratamiento de la información sobre clientes y proyectos.
- *Comunicación:* se encarga del tratamiento de los avisos y alertas que se definan, enviando comunicaciones según el mecanismo apropiado (mail, notificación en vivo, etc).
- *Complementos:* es el subsistema encargado de gestionar todo tipo referencias y material complementario que puede ser utilizado por los traductores, tales como diccionarios, enciclopedias y acceso a sistemas externos (Google, Word Reference, etc).
- *Acceso a Datos:* encapsula las funcionalidades relacionadas con la persistencia de la información administrada por el sistema, tanto en base de datos como en sistema de archivos.

3.4.3 Diagrama de distribución

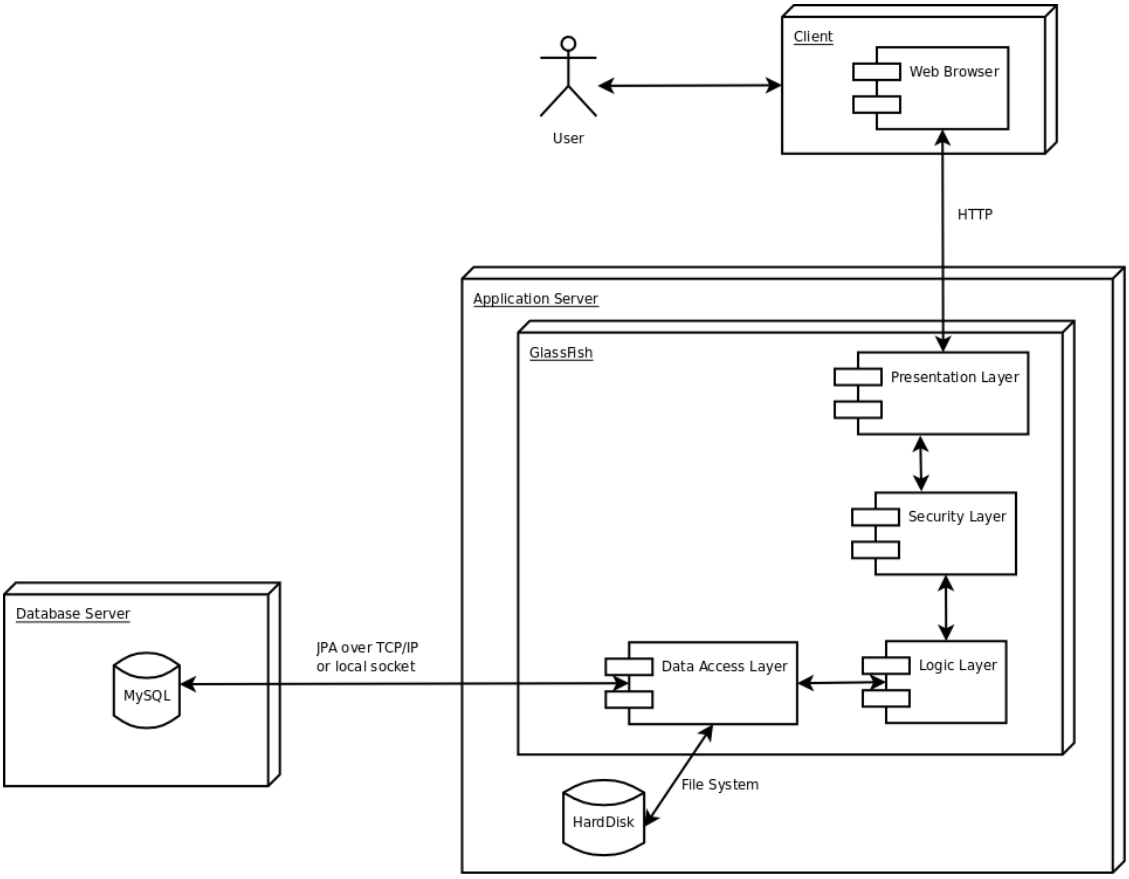


Figura 3.4 Diagrama de deployment

4 Implementación

4.1 Decisiones

En esta sección presentamos las principales decisiones que tomamos en lo referido a la implementación.

4.1.1 Tecnología

Java, Glassfish y MySQL

No consideramos como decisiones propias el hecho de haber utilizado Java como lenguaje de programación, Glassfish [web-39] como servidor de aplicaciones y MySQL [web-40] como servidor de bases de datos, ya que éstos fueron requerimientos.

Java Server Faces

Optamos por Java Server Faces (JSF) [web-38] para realizar la implementación de la presentación, con base en páginas JSP. Consideramos como alternativa la posibilidad de usar Struts [web-41] pero nos inclinamos por JSF porque es más moderno y está siendo cada vez mas usado.

AJAX

Para la mayoría de los casos, la transición entre páginas JSP nos resultaba adecuada, pero para otros necesitamos de mayor flexibilidad y velocidad, por lo que optamos por la tecnología AJAX [web-42], que permite obtener datos desde el servidor sin necesidad de recargar la página. Un caso notorio para esta necesidad es el correspondiente al proceso de traducción, en el cual para cada segmento se debe obtener y mostrar una lista de candidatos, y, una vez elegido uno o ingresada la traducción se la debe registrar y pasar al siguiente segmento. Si esto se hiciera recargando la página cada vez sería muy lento. Utilizando AJAX logramos transferir desde el cliente al servidor y viceversa solo la información necesaria. Del lado del cliente utilizamos Javascript para realizar la implementación, mientras que del lado del servidor implementamos un conjunto de servlets.

JSON

Para facilitarnos la implementación de lo mencionado, utilizamos JSON [web-35] para representar la información de forma de poder enviarla de un extremo al otro (cliente y servidor) y poder interpretarla correctamente en el extremo receptor.

Mootools y Mocha

Siguiendo con la implementación del lado del cliente, complementamos el uso de Javascript con las librerías Mootools [web-33] y Mocha [web-34], que proveen funcionalidades que simplifican la tarea de desarrollo.

EJB 3.0

Decidimos utilizar la tecnología EJB 3.0 [web-36, web-37]. De esta manera, implementamos las funcionalidades como Enterprise Java Beans (EJBs) y representamos los conceptos del dominio (usuarios, documentos, segmentos, etc) como entidades. Para complementar esto, la persistencia de la información decidimos hacerla con la tecnología JPA (Java Persistence API).

4.1.2 Subproyectos

Para la implementación dividimos el proyecto en cuatro sub-proyectos de Netbeans: uno que solo contiene la presentación (páginas JSP, imágenes, hojas de estilo, etc), al que llamamos *baboon-web*, otro que solo contiene la lógica y el acceso a datos (ejbs y entidades), llamado *baboon-ejb*, otro que contiene las interfaces remotas para los ejbs, al que llamamos *baboon-remote* y finalmente otro que contiene los elementos comunes a todos ellos (datatypes y utilidades), llamado *baboon-lib*.

La lógica de la aplicación reside únicamente en el proyecto baboon-ejb, y cualquier cambio que deba hacerse a ésta, como agregar una nueva clase para búsqueda de candidatos, o algoritmo de comparación, debe hacerse en este proyecto. La comunicación entre la presentación y la lógica se realiza mediante las interfaces remotas; para simplificar nuestro trabajo implementamos clases, a las que llamamos *delegates,* que se encargan de obtener referencias a las interfaces (lookup) e invocar las operaciones. El intercambio de información entre la presentación y la lógica y viceversa siempre involucra *datatypes* (TOs).

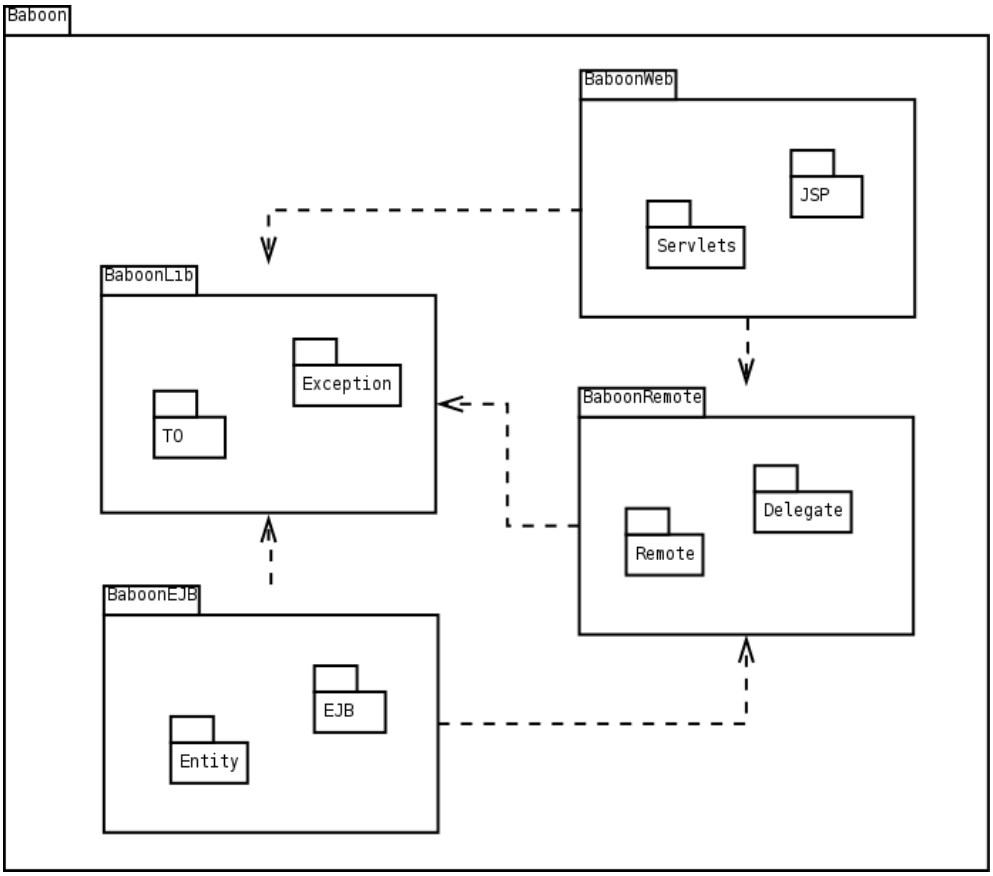


Figura 4.1 Diagrama de subproyectos

Implementamos el patrón de diseño MVC (Model – View – Controller, o Modelo – Vista – Controlador) [web-43] que persigue el objetivo de separar la lógica de la presentación, estableciendo un modelo o una representación abstracta de la realidad, el cual es el único punto de contacto entre ambos. De esta forma, pueden introducirse cambios en la lógica del sistema o en la presentación sin necesidad de alterar la otra parte.

4.1.3 Herramientas y estándares

Xliff como formato de representación de los textos

Decidimos utilizar el estándar XLIFF ya que actualmente es muy usado y además encontramos por lo menos dos herramientas open source que permiten realizar la conversión de un documento a XLIFF y viceversa: OLT y File2XLIFF4J. Luego de algunas pruebas con ambos, nos decidimos por File2XLIFF4J por dos razones: tiene soporte para el formato propietario usado por Framemaker, uno de los formatos requeridos, y tuvimos acceso al código fuente para poder adaptarlo a nuestras necesidades.

Segmentación con File2XLIFF4J

Habiendo elegido XLIFF como formato de representación de los textos a traducir, y File2XLIFF4J como herramienta conversora, resolvimos también el problema de la segmentación del texto. Esto englobaba en realidad dos problemas: determinar cuál es la granularidad adecuada para la segmentación del texto, y cómo realizar la segmentación. En cuanto a la granularidad optamos por la segmentación en oraciones, ya que consideramos que por párrafos los fragmentos de texto podrían resultar demasiado grandes, disminuyendo la posibilidad de encontrar otros fragmentos ya traducidos similares, y obligando al traductor a un mayor esfuerzo.

TMX como formato de representación de las memorias

Utilizamos TMX como formato de representación de las memorias de traducción ya que es el estándar más ampliamente utilizado y es soportado por la mayoría de las herramientas de asistencia a la traducción por memorias.

Construcción de memorias de traducción

Dado que al comenzar a trabajar no contábamos con una memoria de traducción nos vimos obligados a construir algunas. Para ésto buscamos en Internet artículos técnicos (principalmente con referencia al lenguaje de programación Java para mantener una temática estable) que estuviesen en español e inglés. Luego de seleccionar varios documentos con las características mencionadas, utilizamos una herramienta para transformarlos a XLIFF (implementada por nosotros en base al proyecto File2XLIFF4J), logrando así una segmentación igual que la que usaría nuestro sistema. En el caso ideal, al tratarse de pares de documentos en dos idiomas distintos, los segmentos se corresponderían exactamente uno a uno. Sin embargo, debido a las diferencias de puntuación entre los idiomas, la implementación del segmentador usado para los diferentes lenguajes, y las diferencias en los criterios al escribir artículos y traducirlos, frecuentemente los segmentos no se correspondían. Por esto, y dado lo complejo de la tarea de construir un alineador automático, decidimos hacer el trabajo manualmente. Utilizamos entonces editores de TMX (TMXEditor y HeartSome) para generar las memorias, eligiendo el lenguaje inglés como base y para cada segmento en los artículos de dicho idioma, fuimos buscando, o armando cuando fuera necesario, los segmentos correspondientes en el idioma español. Así obtuvimos un conjunto de memorias de traducción, cada una con un promedio de 300 entradas (segmentos con sus traducciones), sumando un total de alrededor de 800 entradas.

4.2 Búsqueda de candidatos

Al momento de traducir un segmento, el sistema debe buscar en la memoria otros textos ya traducidos que sean similares. En este caso se engloban dos complicaciones: por un lado, el problema fundamental radica en cómo comparar dos segmentos para obtener una métrica de similaridad, y en segundo lugar cuándo considerar que dos

segmentos son similares en base a la métrica anterior. Para lo primero, ensayamos varias estrategias, que detallamos a continuación. Respecto de lo segundo, definimos un valor, al que llamamos *similaridad mínima admitida* (SMA), tal que solo los segmentos que tienen un valor de similaridad superior a la SMA son considerados similares.

Para darle una base común a todas las estrategias definimos una interfaz que llamamos *Candidater,* que debe ser implementada para poner en práctica los diferentes algoritmos de búsqueda de candidatos. Además, implementamos el patrón Factory Method mediante el cual logramos que el sistema utilice el algoritmo apropiado según la configuración existente. De esta forma reducimos el acoplamiento, flexibilizamos la incorporación de nuevas estrategias y permitimos cambiar de estrategia sin necesidad de detener o modificar la implementación del sistema.

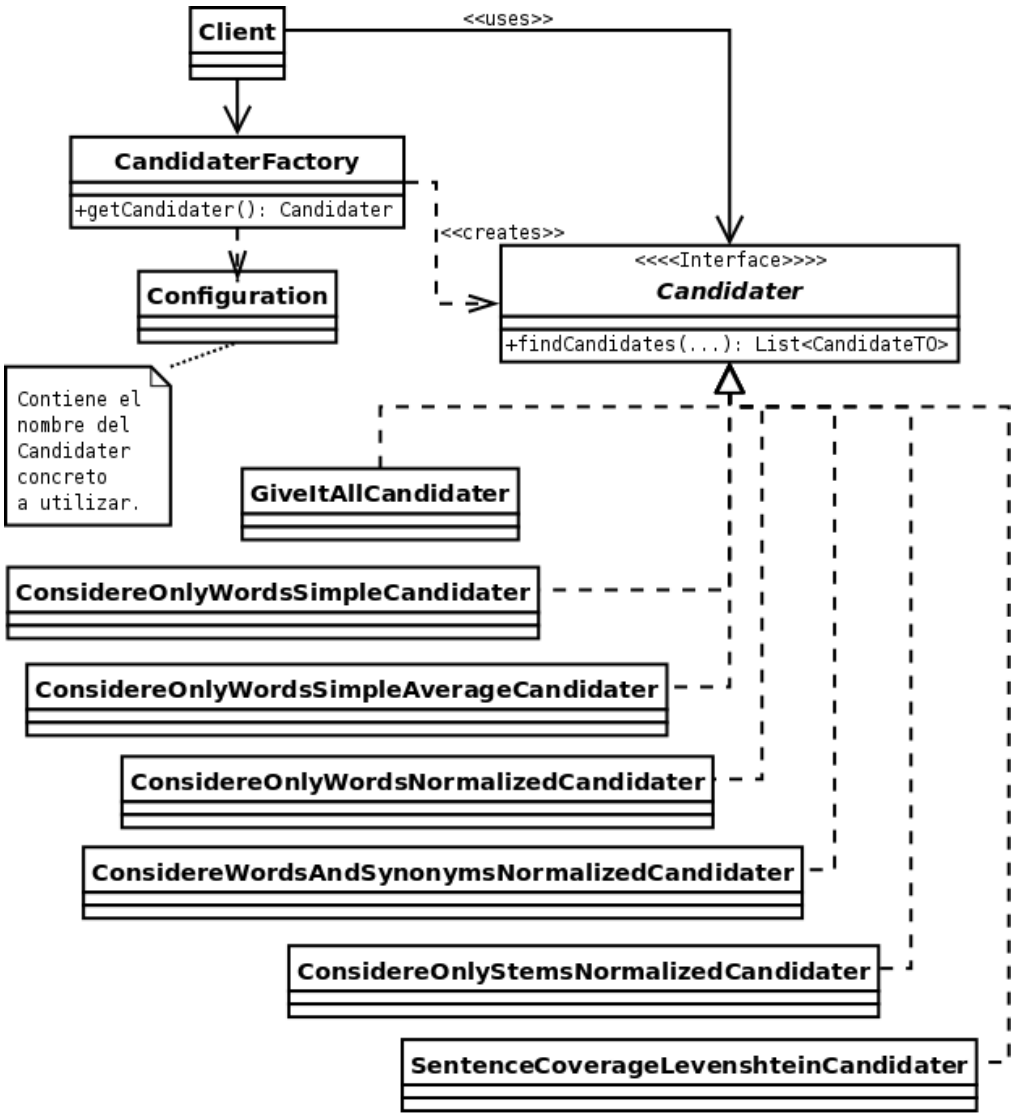


Figura 4.2 Diagrama de clases de candidatos

La clase Factory obtiene de la configuración el nombre de la clase que implementa el algoritmo que se quiere utilizar, e instancia y retorna el objeto deseado. Esto permite que el algoritmo utilizado para obtener los candidatos pueda ser cambiado durante la ejecución del sistema. De igual manera la similaridad mínima admitida puede ser modificada.

Técnica de diccionario

Sería imposible comparar cada segmento a traducir contra todos los segmentos almacenados en la memoria de traducción. Por esto necesitamos un mecanismo para filtrar los segmentos contra los que vale la pena comparar de aquellos que en lo previo es seguro que no podrán ser considerados candidatos a traducciones. Para hacer esto, utilizamos la técnica de diccionario, como se explica a continuación.

La *técnica de diccionario* basa su funcionamiento en la indización de los segmentos por las palabras que contienen. Así, de cada segmento se extraen sus palabras y se registran "punteros" de éstas a todos los segmentos en los que figuran. Cuando se debe procesar un nuevo segmento, se buscan en el diccionario todos los segmentos ya ingresados y que tienen alguna palabra en común; esto es posible bajo el preconcepto que dos segmentos no pueden ser similares si no tienen alguna palabra en común. Igualmente, cada vez que un nuevo segmento es traducido, se añade un puntero para cada palabra que lo compone hacia el propio segmento, de forma de que para el próximo segmento, éste sea considerado.

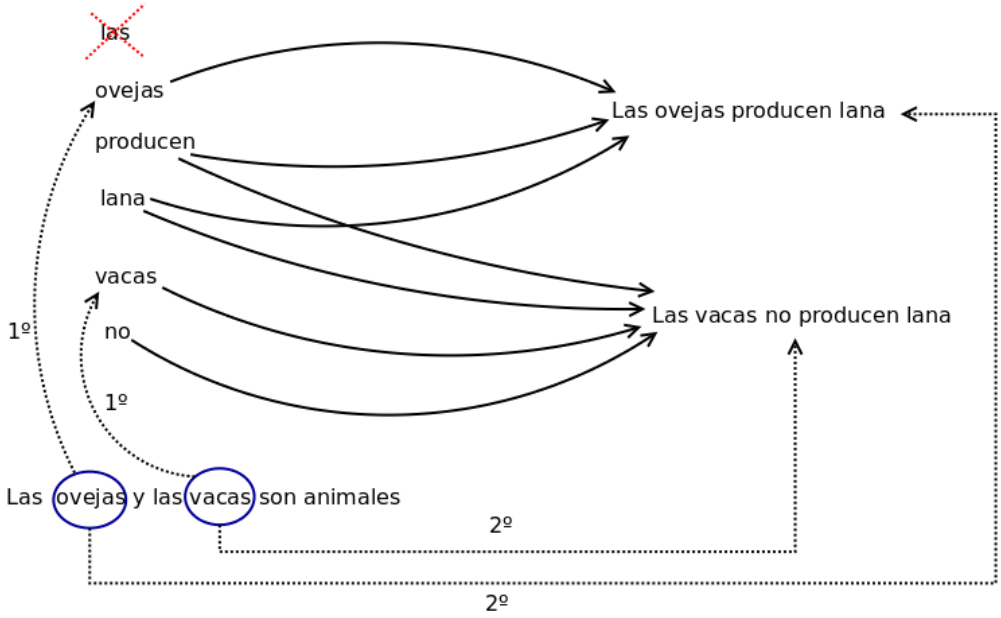


Figura 4.3 Diagrama técnica de diccionario

Gracias a la técnica de diccionario, logramos recortar la memoria de traducción para obtener solo los segmentos que por lo menos tienen alguna palabra en común. Estos segmentos deben ser comparados con el segmento actual para determinar su nivel de similaridad. Incluso se puede añadir la restricción de que cada segmento debe contener un número fijo o un porcentaje de las palabras en común con el segmento a traducir para ser considerado.

4.2.1 Generadores de candidatos

A continuación describimos los diferentes algoritmos que implementamos.

Búsqueda por coincidencia exacta

El algoritmo mas trivial de búsqueda de candidatos es el que simplemente retorna todos los segmentos de texto que se encuentran en la base de datos sin realizar ningún tipo de filtro, calculando la similaridad tan solo por exactitud. Por lo tanto retornará los

candidatos con un valor de similaridad de 100 si es idéntico al segmento actual o 0 en caso contrario.

Comparación por cantidad de palabras en común

La segunda aproximación que usamos fue calcular el porcentaje simple de palabras que ambos segmentos tienen en común; el problema con esta aproximación es que depende en exceso del largo de cada uno de los segmentos, y especialmente de la diferencia entre dichos largos, además de que, teniendo en cuenta dicha diferencia, no queda claro sobre cuál de las longitudes se debe promediar. Por ejemplo, para los fragmentos de texto *Primera muestra* y *Esta es la primera muestra del trabajo*, se nota que tienen dos palabras en común, pero la relación respecto del primer segmento sería 2/2=1, o sea, 100%, mientras que respecto del segundo segmento sería 2/7=0.29, o sea, 29%. Podría sugerirse considerar siempre el más largo para promediar o el segmento a traducir, pero estas no son más que decisiones arbitrarias.

El algoritmo implementado considera solo aquellos segmentos que tienen palabras en común con éste. Una vez obtenidos los segmentos que pueden llegar a ser candidatos realizamos un simple cálculo de promedio de palabras en común entre el segmento actual y cada uno de los potenciales candidatos obtenidos:

$$simil = \frac{pc}{pt}$$

donde *simil* es la medida de similaridad resultante, *pc* es la cantidad de palabras en común y *pt* la cantidad de palabras en el segmento a traducir. Finalmente, para cada potencial candidato si el valor obtenido es mayor o igual que la SMA entonces el segmento pasa definitivamente a formar parte del conjunto de candidatos del segmento actual. Implementamos también una variante de este algoritmo en donde pt es la cantidad de palabras del candidato considerado.

Comparación por cantidad de palabras en común promediado

Visto el problema que presenta el método anterior, decidimos utilizar un método que no se vea afectado por la decisión arbitraria de cuál segmento utilizar para promediar: se usan los dos, simplemente se considera la semisuma de ambos promedios. Así, considerando el mismo ejemplo, se obtiene una única métrica de (2/2 + 2/7) / 2 = 0.64, o sea, 64%.

El algoritmo que implementamos obtiene la medida de similaridad a partir de la semisuma de ambos promedios de la siguiente manera:

$$simil = \frac{\frac{pc}{pst} + \frac{pc}{psc}}{2}$$

donde *pc* es la cantidad de palabras en común, *pst* es la cantidad de palabras del segmento a traducir, y *psc* es la cantidad de palabras del segmento con el que se está comparando.

El problema de este algoritmo se da cuando la longitud de los segmentos a comparar difiere mucho. Por ejemplo, si consideramos los segmentos *XML Sample 1* y *XML Sample 1 shows the contents of the purchase order before it is signed,* obtenemos una similaridad del 60% gracias a que el primero de ellos está totalmente contenido en el

otro, sin embargo nuestro sentido común indica que la similaridad entre ellos es menor.

Comparación por cantidad de palabras en común normalizado

En este algoritmo decidimos aplicar un calculo más complejo, al reemplazar el promedio simple por una normalización [8]: en lugar de dividir la suma de los promedios entre dos, consideramos únicamente la cantidad de palabras en común dividida entre la raíz cuadrada del producto de las longitudes de ambos segmentos (la técnica de normalización estándar):

$$simil = \frac{pc}{\sqrt{pst * pc}}$$

El hecho de normalizar, evita que la diferencia entre largos de los segmentos de texto que se comparan influya demasiado, y permite que el peso de la comparación caiga específicamente en las palabras en común. Además consideramos los lemas y no las palabras para buscar y comparar de forma que ampliamos el espacio de búsqueda, considerando segmentos presentes en la memoria de traducción que, si bien no tienen las palabras exactas que contiene el segmento a traducir, contienen términos con el mismo lema.

Con el mismo ejemplo utilizado en el caso anterior obtuvimos un valor de similaridad menor: 2 / raíz(2 * 7) = 0.53, o sea 53%.

Comparación por cantidad de palabras en común normalizado utilizando sinónimos

Como extensión del algoritmo anterior probamos considerar también los sinónimos de cada lema del segmento actual. La razón para esto es que es frecuente que para una misma palabra en un segmento, exista más de una traducción, por lo que es posible que un segmento no sea encontrado buscando una palabra en particular pero sí un sinónimo de ésta.

Entonces, al momento de obtener los segmentos candidatos que tuvieran algún lema en común con el segmento actual, solicitamos también aquellos que tuvieran algún sinónimo de los lemas del segmento actual. Por ejemplo, si el segmento actual es *It specifies exactly how the model data should be presented*, con el algoritmo anterior se buscarían los segmentos que tuvieran alguno de los siguientes lemas: *it, specify, exactly, how, model, datum, should, be, present*; pero con el nuevo método obtendríamos sinónimos para 5 de ellos y entonces el conjunto de lemas se vería aumentado con: *stipulate*, *precisely*, *framework*, *exist*, *show*. Por razones de rendimiento debimos limitarnos a considerar solo un sinónimo para cada palabra, aunque para muchas de ellas existen más de uno, incluso siendo importante la categoría gramatical de la misma.

El cálculo que utilizamos es el mismo que en algoritmo anterior, salvo que la cantidad de palabras en común puede haber aumentado, al igual que la cantidad de segmentos a considerar.

Comparación por cantidad de raíces en común normalizado

Este algoritmo es similar a los anteriores, en el sentido de que utiliza la misma fórmula para calcular la similaridad, pero solo toma en cuenta las raíces (stems) de las palabras, no las propias palabras, sus lemas o sinónimos.

Comparación por tri-gramas en común normalizado

Este algoritmo considera la cantidad de tri-gramas en común de los segmentos para calcular su similaridad. Aunque la búsqueda en el diccionario se sigue haciendo por

palabras, una vez que los posibles candidatos fueron seleccionados se determinan los tri-gramas de cada uno y se calcula su similaridad con el segmento a traducir como:

$$simil = \frac{tc}{\sqrt{ts * tsm}}$$

donde *tc* es la cantidad de tri-gramas en común, *ts* la cantidad de tri-gramas del segmento a traducir y tsm la cantidad de tri-gramas del segmento que se extrajo de la memoria.

Levenshtein extendido

Los algoritmos anteriores, excepto tri-gramas, tienen una característica en común: todos ellos parten de contar la cantidad de palabras (tokens, en realidad) que tienen en común los segmentos a comparar. Como alternativa, quisimos probar un algoritmo basado en el algoritmo de similaridad de Levenshtein aunque extendido para textos pequeños (el algoritmo de Levenshtein está pensado para palabras individuales). A este algoritmo lo denominamos *algoritmo de cubrimiento de oración* [web-1].

El algoritmo consiste en realizar el producto cartesiano entre cada par de segmentos a comparar y a continuación calcular la distancia de Levenshtein para cada par de palabras. Al hacer el producto cartesiano, se obtiene un conjunto de pares de palabras en las que la primer componente aparece en el segmento a traducir y la segunda en el segmento extraído de la memoria de traducción. Si bien Levenshtein calcula distancias de edición, este algoritmo requiere porcentajes de similaridad, para lo cual realizamos el cálculo siguiente: similaridad_entre_palabras = 100 – (distancia_de_levenshtein * 100) / (largo_de_la_palabra_del_segmento_a_traducir).

Con lo anterior se puede construir una tabla en la que las palabras del segmento a traducir se ubican verticalmente y las del segmento obtenido de la memoria se ubican horizontalmente; las entradas de la tabla corresponden con las similaridades calculadas anteriormente. Finalmente se debe analizar la tabla y buscar todas las combinaciones posibles de entradas tales que en cada combinación no hay dos obtenidas de la misma fila o de la misma columna. Para cada una de tales combinaciones encontradas se suman los valores y se divide entre la cantidad de elementos que forman parte de la combinación. El mayor valor obtenido es precisamente la métrica de similaridad de los segmentos.

Aunque este parece ser un buen algoritmo, padece el mismo problema que presenta el algoritmo de Levenshtein: la implementación no es sencilla y es difícil lograr una implementación eficiente. Empíricamente comprobamos que es inaplicable para segmentos mas o menos largos (más de 10 palabras) ya que el tiempo que toma el algoritmo es excesivo (minutos). Para superar esta dificultad decidimos que en caso de que alguno de los dos segmentos a comparar tenga más de 10 palabras solo consideramos las que no son comunes de ambos. Además, para elegir los candidatos a comparar con el segmento original tomamos aquellos que tienen un mínimo de 5 palabras en común, reduciendo así la cantidad de veces que se debe aplicar el algoritmo de Levenshtein extendido. Con estas modificaciones comprobamos que se obtienen resultados similares a otros algoritmos en un tiempo aceptable.

Agregado de otros algoritmos

No es difícil agregar nuevos algoritmos ya que solo es necesario implementar la interfaz que mencionamos anteriormente, la cual define una operación, llamada findCandidates que toma los siguientes parámetros:

- EntityManager em: una instancia de un manejador de entidades como se define

en la tecnología EJB3. Esto es necesario para poder acceder a la base de datos utilizando el mismo sistema de persistencia que el resto de la aplicación.

- String text: texto para el cual se buscan candidatos.
- String srcLang: código ISO de dos letras que indica el idioma original del segmento anterior.
- String dstLang: código ISO de dos letras que indica el idioma al cual se está traduciendo.
- Integer memId: identificador de la memoria de traducción que se está usando (no se debería acceder a otra memoria que no sea la indicada).

El resultado de este método es una lista de candidatos, especificados como el texto y la similaridad.

4.3 Análisis léxico y gramatical

OpenNLP y Freeling para análsis

A partir de la investigación realizada destacamos dos herramientas: OpenNLP y Freeling. Luego de algunas pruebas comparativas, optamos por trabajar principalmente con Freeling. Sin embargo, decidimos también implementar las mismas funcionalidades usando OpenNLP y dejar a elección del usuario la herramienta a usar. Por el momento OpenNLP es la única alternativa si la aplicación se ejecuta en un entorno Microsoft Windows.

Con estas herramientas realizamos el trabajo referido al análisis morfológico y gramatical de las palabras, lo que en última instancia resultó fundamental para la implementación de las variantes de la búsqueda por diccionario, por un número de razones: primero, pudimos detectar las palabras genéricas, así como fechas y números; segundo, porque algunas palabras tienen más de una acepción y esto debe considerarse para la búsqueda de diccionario (no es lo mismo buscar el término *copa* como sustantivo que como verbo); y tercero, porque nos permitió obtener lemas y sinónimos de las palabras.

Análisis con WordExtractor

Como dijimos anteriormente, construir nuevos generadores de candidatos no implica gran esfuerzo. Sin embargo, la tarea requiere funciones de análisis de texto, lo cual es logrado con Freeling u OpenNLP. Para abstraer al implementador de esto decidimos encapsular las funcionalidades comunes. Es así que creamos una interfaz que define las siguientes operaciones:

- tokenize: toma como entrada un texto y devuelve una lista de los tokens que lo conforman.
- extractWords: toma como entrada un texto y devuelve una lista de palabras que están registradas en la base de datos.
- isWorthly: toma una categoría gramatical y un idioma y devuelve true si es adecuado considerar las palabras que tienen dicha categoría o no (por ejemplo, para las palabras genéricas devuelve false).
- getSynonyms: toma una palabra, una categoría gramatical y un lenguaje y devuelve una lista de sinónimos para dicha palabra considerando solo las acepciones que correspondan a la categoría indicada.

Para esta interfaz, realizamos dos implementaciones, una utilizando Freeling y otra utilizando OpenNLP. En ambos casos, abarcamos la mayor cantidad posible de idiomas. Para esto, utilizamos las funcionalidades propias de las herramientas cuando nos fue posible, y realizamos implementaciones genéricas para los casos en que no lo fue. Por

ejemplo, la operación tokenize se encarga de extraer los tokens (típicamente palabras, aunque depende del idioma), pero esta funcionalidad es provista por las herramientas solo para algunos idiomas; para otros idiomas recurrimos simplemente a dividir el texto por espacios.

En ambas implementaciones, previo al procesamiento del texto, el mismo se convierte completamente a minúsculas para evitar las complicaciones que involucra tener mayúsculas y minúsculas al momento de comparar las palabras o tokens. Además notamos que en algunos casos tanto Freeling como OpenNLP se confunden al realizar el PoS tagging, considerando dos o más palabras consecutivas que empiezan con mayúsculas (por ejemplo en un título) como nombres propios.

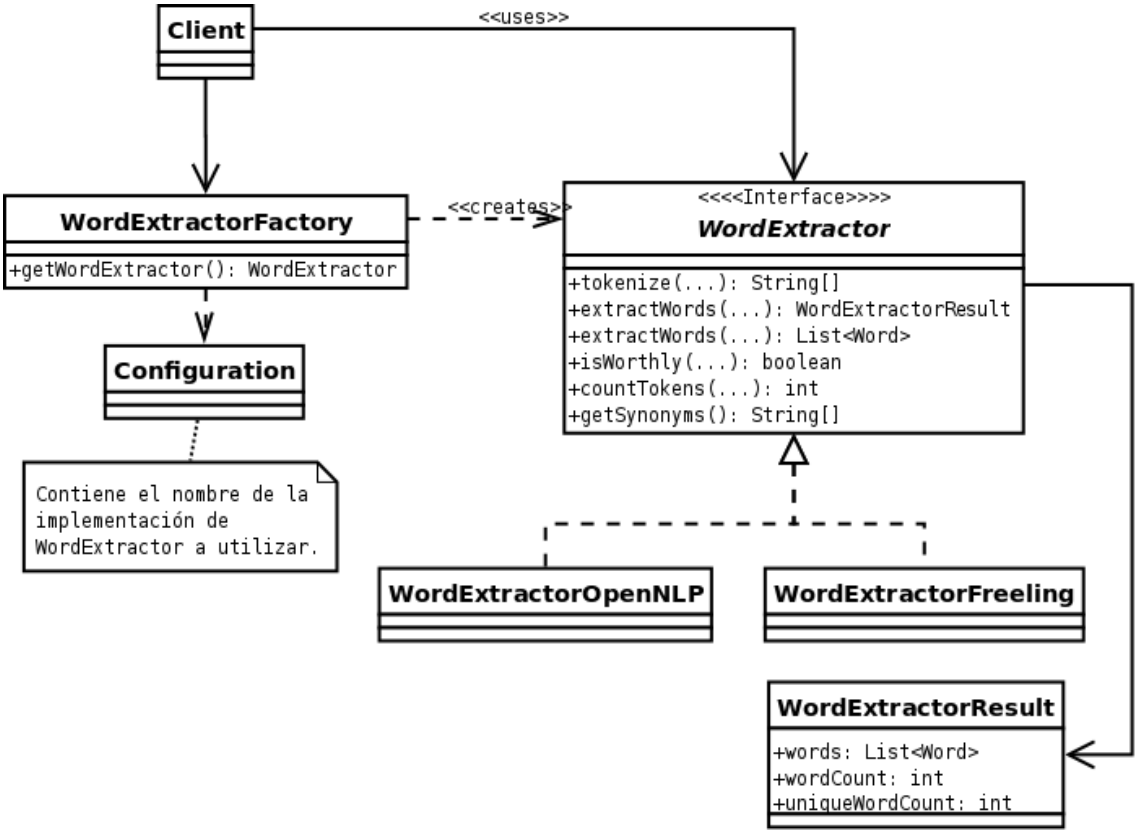


Figura 4.3 Diagrama WordExtractor e implementaciones

Snowball para calcular raíces de palabras

Además de Freeling y OpenNLP, para el procesamiento de lenguaje natural debimos recurrir a otra herramienta ya que éstas, a pesar de ser muy completas, no poseen la capacidad de determinar la raíz de las palabras. Aunque existen varios algoritmos para esto, tampoco era nuestro interés estudiarlos e implementarlos, especialmente cuando nos encontramos con una herramienta llamada Snowball, que hace eso para varios idiomas.

4.4 Pruebas

Realizamos un testing basado en las funcionalidades en el cual probamos diferentes escenarios para cada una. Las pruebas se encuentran documentadas en el apéndice [ap-testing]. Para hacer las pruebas cargamos inicialmente el sistema con un par de memorias de traducción, elegimos un artículo en inglés para traducir al español y otro en español para probar la traducción hacia el inglés. Dada las características de este tipo de

sistemas los artículos que elegimos eran técnicos sobre el lenguaje de programación Java, en particular seleccionamos un artículo que estaba en ambos idiomas. A medida que fuimos implementando los algoritmos los probamos con los mismos artículos. Durante el desarrollo también realizamos pruebas unitarias de los diferentes subsistemas aunque no lo fuimos documentando.

4.5 Evaluación de resultados

Como primer resultado destacamos que la configuración del sistema está sujeta al ambiente en el que se ejecute el servidor. Esto es así debido a que la utilización de Freeling solo está disponible por el momento para Linux, por lo que en Windows se deberá configurar el sistema para que utilice OpenNLP. Esto adquiere mayor relevancia si tenemos en cuenta que los resultados obtenidos configurando el sistema para que utilice OpenNLP son de menor calidad que los obtenidos configurándolo para utilizar Freeling, ya que comprobamos que las capacidades de análisis lingüístico de este último son ampliamente superiores.

4.5.1 Comparación de los algoritmos implementados

Para realizar las pruebas comparativas cargamos una memoria en el sistema que contiene 780 entradas en inglés y español. Seleccionamos cuatro segmentos de la memoria, a los que les aplicamos diferentes modificaciones y configuramos el sistema para que utilice Freeling como analizador de textos y no considere las palabras genéricas. Para cada uno de estos segmentos, en ambos idiomas, ejecutamos nuestros algoritmos de búsqueda de candidatos. Para no hacer muy extensa la comparación de resultados, a continuación mostramos en forma resumida solo los correspondientes a la traducción de inglés a español.

Segmento 1: XML Sample 1 shows the contents of the purchase order before it is signed.		
Generadores de candidatos	**Candidatos**	**Sim***
Coincidencia exacta	El Ejemplo de XML 1 presenta el contenido de la orden de compra antes de ser firmada.	100
	Cantidad de candidatos de similaridad menor a 30: 779	
Cantidad de palabras en común	El Ejemplo de XML 1 presenta el contenido de la orden de compra antes de ser firmada.	100
	Ejemplo de XML 1	100
	Ejemplo de XML 2	100
	Contenidos	100
	El Ejemplo de XML 3 presenta el elemento SignedInfo que contiene la información que realmente se firma.	55
	Ejemplo de Código 1	50
	Ejemplo de Código 2	50
	El ejemplo que este artículo usará es una firma XML envuelta generada sobre los contenidos de un documento XML, una muestra de orden de compra.	42
	Uno o más elementosReference identifican los datos que se firman.	40
	El Ejemplo de Código 5 muestra los pasos clave para validar una firma XML.	37
	(entre otros)	
	Cantidad de candidatos de similaridad menor a 30: 204	
Cantidad de palabras en común promediado	El Ejemplo de XML 1 presenta el contenido de la orden de compra antes de ser firmada.	76
	Ejemplo de XML 1	56
	Ejemplo de XML 2	56
	Contenidos	53
	El Ejemplo de XML 3 presenta el elemento SignedInfo que contiene la información que realmente se firma.	41
	El ejemplo que este artículo usará es una firma XML envuelta generada sobre los contenidos de un documento XML, una muestra de orden de compra.	40
	Cantidad de candidatos de similaridad menor a 30: 236	
Cantidad de lemas en común normalizado	El Ejemplo de XML 1 presenta el contenido de la orden de compra antes de ser firmada.	100
	El Ejemplo de XML 3 presenta el elemento SignedInfo que contiene la información que realmente se firma.	56
	El ejemplo que este artículo usará es una firma XML envuelta generada sobre los contenidos de un documento XML, una muestra de orden de compra.	55
	Por lo tanto, puede ser importante mostrar los contenidos de lo que exactamente ha sido firmado por el usuario que realiza la validación.	53
	Ejemplo de XML 1	50
	Ejemplo de XML 2	50
	Observe que el elemento Signature ha sido insertado dentro del contenido que se firma, convirtiéndola de ese modo en una firma envuelta.	39
	Las firmas XML se describen con frecuencia como pertenecientes a uno o más de tres tipos:	38
	(entre otros)	
	Cantidad de candidatos de similaridad menor a 30: 291	
Cantidad de raíces en común normalizado	El Ejemplo de XML 1 presenta el contenido de la orden de compra antes de ser firmada.	100
	El Ejemplo de XML 3 presenta el elemento SignedInfo que contiene la	56

	información que realmente se firma.	
	El ejemplo que este artículo usará es una firma XML envuelta generada sobre los contenidos de un documento XML, una muestra de orden de compra.	55
	Ejemplo de XML 1	50
	Ejemplo de XML 2	50
	El Ejemplo de Código 5 muestra los pasos clave para validar una firma XML.	38
	Una firma XML puede firmar datos arbitrarios, siendo estos XML o binarios.	35
	Ejemplo de Código 1 presenta algunos pasos claves para generar una firma en XML :	35
	(entre otros)	
	Cantidad de candidatos de similaridad menor a 30: 242	
Cantidad de lemas en común normalizado utilizando sinónimos	El Ejemplo de XML 1 presenta el contenido de la orden de compra antes de ser firmada.	100
	El Ejemplo de XML 3 presenta el elemento SignedInfo que contiene la información que realmente se firma.	56
	El ejemplo que este artículo usará es una firma XML envuelta generada sobre los contenidos de un documento XML, una muestra de orden de compra.	55
	Por lo tanto, puede ser importante mostrar los contenidos de lo que exactamente ha sido firmado por el usuario que realiza la validación.	53
	Ejemplo de XML 1	50
	Ejemplo de XML 2	50
	Este mensaje muestra el elemento canonicalizado SignedInfo antes de ser firmado.	40
	Observe que el elemento Signature ha sido insertado dentro del contenido que se firma, convirtiéndola de ese modo en una firma envuelta.	39
	Ahora se debe instanciar el documento a ser firmado, crear el objeto XMLSignature y generar la firma, como se puede ver en el Ejemplo de Código 3.	39
	Las firmas XML se describen con frecuencia como pertenecientes a uno o más de tres tipos:	38
	Una firma XML puede firmar datos arbitrarios, siendo estos XML o binarios.	35
	(entre otros)	
	Cantidad de candidatos de similaridad menor a 30: 310	
Comparación por tri-gramas en común normalizado	El Ejemplo de XML 1 presenta el contenido de la orden de compra antes de ser firmada.	100
	El ejemplo que este artículo usará es una firma XML envuelta generada sobre los contenidos de un documento XML, una muestra de orden de compra.	46
	El Ejemplo de XML 3 presenta el elemento SignedInfo que contiene la información que realmente se firma.	43
	Esta sección le mostrará cómo usar la API para generar una firma XML sobre los contenidos del elemento PurchaseOrder que el artículo introdujo anteriormente.	40
	Ejemplo de Código 1 presenta algunos pasos claves para generar una firma en XML :	39
	Ejemplo de XML 1	37
	Cantidad de candidatos de similaridad menor a 30: 219	
Levenshtein extendido	El Ejemplo de XML 1 presenta el contenido de la orden de compra antes de ser firmada.	88
	Cantidad de candidatos de similaridad menor a 30: 3	

Segmento2: XML Sample 1		
Generadores de candidatos	**Candidatos**	**Sim***
Coincidencia exacta	Ejemplo de XML 1	100
	Cantidad de candidatos de similaridad menor a 30: 779	
Cantidad de palabras en común	Ejemplo de XML 1	100
	Ejemplo de XML 2	100
	Ejemplo de Código 1	50
	Ejemplo de Código 2	50
	Validar una Firma en XML	33
	Generación de una Firma XML	33
	Cantidad de candidatos de similaridad menor a 30: 85	
Cantidad de palabras en común promediado	Ejemplo de XML 1	83
	Ejemplo de XML 2	83
	El Ejemplo de XML 1 presenta el contenido de la orden de compra antes de ser firmada.	45
	El Ejemplo de Código 5 muestra los pasos clave para validar una firma XML.	45
	Ejemplo de Código 1 presenta algunos pasos claves para generar una firma en XML :	44
	La firma XML envuelta resultante, con sangría y formateada para su legibilidad, aparece en el Ejemplo de XML 2.	43
	Ejemplo de Código 1	41
	Visión general de las Firmas XML	33
	Cantidad de candidatos de similaridad menor a 30: 76	
Cantidad de lemas en común normalizado	Ejemplo de XML 1	100
	Ejemplo de XML 2	100
	El Ejemplo de XML 1 presenta el contenido de la orden de compra antes de ser firmada.	50
	El Ejemplo de Código 5 muestra los pasos clave para validar una firma XML.	50
	Ejemplo de Código 1	50
	Ejemplo de Código 1 presenta algunos pasos claves para generar una firma en XML :	50
	(entre otros)	
	Cantidad de candidatos de similaridad menor a 30: 70	
Cantidad de raíces en común normalizado	Ejemplo de XML 1	100
	Ejemplo de XML 2	100
	El Ejemplo de XML 1 presenta el contenido de la orden de compra antes de ser firmada.	50
	El Ejemplo de Código 5 muestra los pasos clave para validar una firma XML.	50
	Ejemplo de Código 1	50
	Ejemplo de Código 1 presenta algunos pasos claves para generar una firma en XML :	47
	(entre otros)	
	Cantidad de candidatos de similaridad menor a 30: 70	
Cantidad de lemas en común normalizado utilizando sinónimos	Ejemplo de XML 1	100
	Ejemplo de XML 2	100
	El Ejemplo de XML 1 presenta el contenido de la orden de compra antes de ser firmada.	50
	El Ejemplo de Código 5 muestra los pasos clave para validar una firma XML.	50

	Ejemplo de Código 1	50
	Ejemplo de Código 1	50
	Ejemplo de Código 1 presenta algunos pasos claves para generar una firma en XML :	47
	La firma XML envuelta resultante, con sangría y formateada para su legibilidad, aparece en el Ejemplo de XML 2.	45
	El elemento SignatureValue contiene el valor de firma codificada en base 64 sobre el elemento SignedInfo, como se puede ver en el Ejemplo de XML 4.	43
	(entre otros)	
	Cantidad de candidatos de similaridad menor a 30: 70	
Comparación por tri-gramas en común normalizado	Ejemplo de XML 1	100
	Ejemplo de XML 2	90
	Ejemplo de Código 1	67
	Ejemplo de Código 2	57
	El Ejemplo de XML 1 presenta el contenido de la orden de compra antes de ser firmada.	37
	Ejemplo de Código 1 presenta algunos pasos claves para generar una firma en XML :	34
	Cantidad de candidatos de similaridad menor a 30: 85	
Levenshtein extendido	Ejemplo de XML 1	100
	Ejemplo de XML 2	66
	(entre otros)	
	Cantidad de candidatos de similaridad menor a 30: 9	

Segmento 3: By Robert Eckstein, April 2005		
Generadores de candidatos	**Candidatos**	**Sim***
Coincidencia exacta	No encontró candidatos.	
	Cantidad de candidatos de similaridad menor a 30: 780	
Cantidad de palabras en común	Por Robert Eckstein, marzo de 2007	33
	Cantidad de candidatos de similaridad menor a 30: 72	
Cantidad de palabras en común promediado	Por Robert Eckstein, marzo de 2007	66
	Cantidad de candidatos de similaridad menor a 30: 0	
Cantidad de lemas en común normalizado	Por Robert Eckstein, marzo de 2007	100
	Cantidad de candidatos de similaridad menor a 30: 0	
Cantidad de raíces en común normalizado	Por Robert Eckstein, marzo de 2007	100
	Cantidad de candidatos de similaridad menor a 30: 0	
Cantidad de lemas en común normalizado utilizando sinónimos	Por Robert Eckstein, marzo de 2007	100
	Cantidad de candidatos de similaridad menor a 30: 0	
Comparación por tri-gramas en común normalizado	Por Robert Eckstein, marzo de 2007	71
	Cantidad de candidatos de similaridad menor a 30: 0	
Levenshtein extendido	Por Robert Eckstein, marzo de 2007	79
	Cantidad de candidatos de similaridad menor a 30: 0	

Segmento 4: Security concerns are explained in detail in the security considerations section.		
Generadores de candidatos	**Candidatos**	**Sim***
Coincidencia exacta	No encontró candidatos.	
	Cantidad de candidatos de similaridad menor a 30: 780	
Cantidad de palabras en común	Estos y otros asuntos se analizarán en más detalle en la sección de consideraciones de seguridad. (XML Signature Recommendation).	70
	Los escenarios son:	50
	La próxima sección analizará los KeySelectors en más detalle.	33
	Cantidad de candidatos de similaridad menor a 30: 67	
Cantidad de palabras en común promediado	Estos y otros asuntos se analizarán en más detalle en la sección de consideraciones de seguridad. (XML Signature Recommendation).	55
	Cantidad de candidatos de similaridad menor a 30: 69	
Cantidad de lemas en común normalizado	Estos y otros asuntos se analizarán en más detalle en la sección de consideraciones de seguridad. (XML Signature Recommendation).	69
	JDK 6 incluye una API para firma digital criptográfica que se describe más detalladamente en una lección sobre el rastreo de seguridad en el Tutorial Java.	37
	Cantidad de candidatos de similaridad menor a 30: 235	
Cantidad de raíces en común normalizado	Estos y otros asuntos se analizarán en más detalle en la sección de consideraciones de seguridad. (XML Signature Recommendation).	69
	Cantidad de candidatos de similaridad menor a 30: 73	
Cantidad de lemas en común normalizado utilizando sinónimos	Estos y otros asuntos se analizarán en más detalle en la sección de consideraciones de seguridad. (XML Signature Recommendation).	69
	JDK 6 incluye una API para firma digital criptográfica que se describe más detalladamente en una lección sobre el rastreo de seguridad en el Tutorial Java.	37
	Cantidad de candidatos de similaridad menor a 30: 244	
Comparación por tri-gramas en común normalizado	Estos y otros asuntos se analizarán en más detalle en la sección de consideraciones de seguridad. (XML Signature Recommendation).	66
	JDK 6 incluye una API para firma digital criptográfica que se describe más detalladamente en una lección sobre el rastreo de seguridad en el Tutorial Java.	34
	(entre otros)	
	Cantidad de candidatos de similaridad menor a 30: 65	
Levenshtein extendido	Estos y otros asuntos se analizarán en más detalle en la sección de consideraciones de seguridad. (XML Signature Recommendation).	55
	Cantidad de candidatos de similaridad menor a 30: 0	

Sim* = Similaridad del candidato respecto del segmento a traducir.

Conclusiones

- Consideramos que el algoritmo que brinda mejores resultados de acuerdo a nuestro criterio es el de lemas en común normalizado utilizando sinónimos. El algoritmo de tri-gramas normalizado proporciona también buenos resultados. Cualquiera de estos dos pueden ser utilizados para el trabajo real porque se puede observar que los candidatos que proponen son razonables y los valores de similaridad calculados para ellos elevados.
- Podemos observar que estableciendo un nivel de similaridad mínimo aceptable cercano al 65%, con la mayoría de los algoritmos obtenemos candidatos que necesitan cambios mínimos para ser utilizados como traducción.
- Como esperábamos, el algoritmo de coincidencia exacta no es aplicable al trabajo real debido a que la única manera de que encuentre un candidato para un segmento dado es que el mismo ya esté presente en la memoria. El algoritmo de cantidad de palabras en común tampoco es aplicable debido a que es muy dependiente del largo del segmento que se esta traduciendo (por la forma en que se calcula la similaridad); tampoco aportaría una mejora si en vez de dividir entre el largo del segmento a traducir se dividiera entre el largo del candidato.
- Para muchos de los algoritmos obtenemos un número elevado de candidatos totales pero la mayoría de ellos con niveles de similaridad muy bajos. El tratamiento de estos candidatos que seguramente no serán utilizados podría omitirse introduciendo la misma extensión que utilizamos en Levenshtein, o sea exigiendo que tengan un cierto mínimo de palabras en común antes de compararlos (no solo una).
- En algunos casos a pesar de que el candidato obtenido no es idéntico al segmento a traducir la similaridad calculada es de 100%; esto se debe a que al eliminar las palabras genéricas todas las restantes son comunes a los segmentos.
- Como mencionamos anteriormente, realizando las mismas pruebas con OpenNLP obtuvimos resultados de menor calidad. Los resultados de estas pruebas se pueden ver en el apéndice [ap-pruebas]

5 Conclusiones

En este capitulo describimos el trabajo que realizamos a lo largo del proyecto, mencionando los objetivos que alcanzamos y los que no, haciendo además un análisis del trabajo futuro.

5.1 Desarrollo del proyecto

Comenzamos el proyecto haciendo un relevamiento de los requerimientos, la mayoría de los cuales surgieron por parte del cliente, y otros por interés nuestro en el área. Para formarnos una idea clara de en qué consisten los sistemas de asistencia a la traducción por memorias y su utilidad procedimos a realizar una recopilación de sistemas existentes en el mercado, así como proyectos académicos relacionados. Para complementar realizamos también una investigación de las técnicas utilizadas en el área de procesamiento de lenguaje natural que pudieran sernos de utilidad. A esto le dedicamos los primeros cuatro meses ya que consideramos que era fundamental construir una base sólida sobre la cual trabajar posteriormente.

Partiendo de lo anterior, dedicamos un mes a realizar un análisis de las principales características de los sistemas de asistencia a la traducción por memorias, y así determinamos cuáles debían estar presentes en nuestro sistema. El resultado de esta etapa lo expresamos en forma de casos de uso y establecimos el alcance del proyecto.

Posteriormente realizamos el diseño de la solución, al mismo tiempo que investigamos herramientas que podíamos usar para resolver problemas comunes con los que nos fuimos enfrentando. En ocasiones encontramos más de una herramienta que nos permitía resolver el mismo problema; en estos casos realizamos pruebas comparativas y seleccionamos las que nos dieron mejores resultados según nuestro criterio. De igual manera relevamos estándares usados frecuentemente en otros sistemas y proyectos, de los cuales seleccionamos los que consideramos más apropiados. Aunque habíamos planificado dedicarle un mes a esta etapa, debimos extenderla dos semanas más por las dificultades que nos plantearon algunos problemas.

Como resultado de estas dos últimas etapas elaboramos un estudio del estado del arte en el área de la asistencia a la traducción por memorias.

Los siguientes dos meses y medio los dedicamos a la construcción del sistema, implementando los casos de uso y utilizando las herramientas que seleccionamos. Simultáneamente fuimos realizando pruebas ad-hoc.

Finalmente, los últimos tres meses los ocupamos en la creación de este documento. Esto incluyó la formalización de las pruebas de rendimiento y adecuación al uso, lo que nos llevó a realizar algunas modificaciones sobre lo implementado, especialmente en lo que tiene que ver con los generadores de candidatos.

5.2 Objetivos alcanzados

El principal objetivo que alcanzamos fue la construcción de un sistema de asistencia a la traducción por memorias. Este sistema provee las principales funcionalidades que detectamos durante el análisis incluyendo la capacidad de trabajar con múltiples formatos de archivos, gestionar documentos, proyectos y clientes así como la búsqueda de candidatos por comparación difusa; como mencionamos anteriormente el prototipo con el que cuenta Sofis carece de todas estas funcionalidades. Además el sistema no se limita a trabajar únicamente con los idiomas inglés y español sino que admite otros idiomas (aunque sin las capacidades de análisis lingüístico). Esto cubre una gran parte de las

necesidades iniciales, es decir, contar con una herramienta de este tipo para ser usada en el trabajo diario, principalmente del cliente pero no limitado a él.

Ese primer objetivo no se limita a la implementación del sistema mencionado sino que incluye la documentación del mismo y los proyectos de Netbeans, de forma de que otras personas puedan continuar el desarrollo para mejorarlo o adaptarlo a sus necesidades. Incluso el diseño de la arquitectura fue realizado pensando en esto logrando que el sistema pueda ser extendido sin mucho esfuerzo.

Otro objetivo alcanzado fue la realización de un estudio del estado del arte que resume las principales características de los sistemas existentes en la actualidad, así como técnicas y estándares frecuentemente usados. Este documento es de utilidad para quien se proponga realizar trabajos futuros con base a nuestro proyecto, o que se interese por el área.

5.3 Objetivos no alcanzados

Por razones de tiempo, prioridades y relevancia de ciertas funcionalidades, debimos recortar el alcance planteado inicialmente, por lo que dejamos sin implementar algunas funcionalidades que habíamos planificado. Estas funcionalidades incluyen la internacionalización del sistema (los textos de la interfaz del sistema se visualizan solo en español), la implementación de alertas y notificaciones, la gestión de tiempos y recursos y la presentación de históricos de actividad.

5.4 Extensiones al trabajo

Desde la concepción del sistema dejamos abierta la posibilidad de incorporar nuevos algoritmos de generación de candidatos sin necesidad de introducir modificaciones al resto del sistema, así como también variaciones de los ya implementados. Consideramos que los existentes pueden ser optimizados; por ejemplo se podría introducir en algunos algoritmos la restricción de un mínimo de palabras en común antes de calcular la similaridad con el segmento a traducir, de la misma manera que hicimos con la implementación del algoritmo de Levenshtein extendido.

Aunque permitimos configurar el sistema para que utilice tanto OpenNLP como Freeling, en Microsoft Windows solo funciona con OpenNLP. Según la investigación que hicimos se puede hacer funcionar Freeling en Windows utilizando el entorno Cygwin pero luego de algunos intentos infructuosos decidimos no continuar; en [web-44] y [web-45] se brindan lineamientos para resolver este problema aunque a nosotros no nos dieron el resultado esperado.

Quedó también pendiente la resolución del problema que se presenta cuando se hace deploy de la aplicación luego de haber utilizado al menos una vez alguna funcionalidad de Freeling y que obliga a reiniciar el servidor. Con respecto a esto buscamos una solución pero no la encontramos, por lo que no podemos presentar ninguna sugerencia.

Notamos que varios de los sistemas relevados permiten utilizar múltiples memorias en el trabajo con un documento; nuestro diseño no comprende esta posibilidad pero la valoramos como muy útil y consideramos que sería una extensión deseable.

El sistema desarrollado permite trabajar en forma apropiada solo con los idiomas español e inglés en ambos sentidos; aunque también funciona con otros idiomas, no realiza ningún tipo de análisis lingüístico por lo que la calidad de los resultados decae. La razón de esto es que las herramientas que utilizamos para hacer el análisis lingüístico están limitadas en este sentido: Freeling trabaja con español, inglés e italiano mientras que OpenNLP con español, inglés y alemán.

Debido a que existen algunas diferencias entre las dos herramientas que utilizamos para realizar el análisis lingüístico, en nuestro sistema no es posible usar una memoria que fue creada empleando una herramienta en el proceso de traducción de un documento que utiliza la otra. Consideramos que el sistema debería ser transparente al usuario en este sentido.

Queda pendiente la realización de un sistema de ayuda mediante el cual los usuarios puedan obtener asistencia para la utilización del software. Este sistema podría ser contextual, es decir desplegar diferentes temas de ayuda según la funcionalidad que está siendo utilizada.

6 Glosario

Lista alfabética de términos:

- **Alineación**: proceso de cargar un mismo documento en dos lenguas diferentes para registrar pares que puedan ser usados por un sistema de memorias de traducción.
- **CAT**: *Computer Aided Translation* (Traducción Asistida por Computadora), técnica de traducción de documentos con la asistencia de programas informáticos.
- **Coincidencia**: relación de similitud entre dos segmentos. Puede ser exacta (los segmentos son idénticos bajo alguna relación) o difusa (fuzzy, los segmentos tienen un determinado grado de similitud).
- **Comparación difusa**: técnica que permite comparar un texto con otro y decir, como resultado, si son similares entre sí o no y con qué grado de similaridad, considerando diferentes factores.
- **Corpus paralelo**: conjunto de pares de segmentos en dos lenguas anotados con diferentes tipos de información.
- **Distancia de edición**: cantidad de cambios que se necesitan para transformar un texto en otro agregando, quitando o remplazando caracteres.
- **Escaneo**: proceso de búsqueda en una memoria según un determinado criterio.
- **Filtro XLIFF**: programa que lee un archivo en algún formato determinado (doc, odt, html, properties, etc) lo segmenta y genera los archivos XLIFF. Normalmente puede también realizar el proceso inverso.
- **Fuzzy matching**: ver comparación difusa.
- **GPL**: *General Public License* (Licencia Pública General), licencia utilizada en el ambiente open source para distribuir productos de software.
- **Información no estructurada**: información que no posee una estructura claramente definida que permita tratarla fácilmente por un sistema informático; por ejemplo texto escrito, audio y video.
- **Internacionalización**: proceso de diseñar un software de manera que se pueda adaptar a diferentes idiomas y regiones sin necesidad de introducir modificaciones al mismo.
- **Lema**: forma canónica de una palabra, de la cual derivan otras de la misma familia; por ejemplo, *correr* es el lema de *correr*, *corrida*, *corren* y *corrió*, y *gato* es el lema de *gato*, *gata* y *gatuno*. No es sinónonimo de raíz.
- **Lematización**: proceso de reducción de la forma flexiva de una palabra a la forma canónica asociada, llamada lema. Por ejemplo, de las palabras *perro*, *perra* y *perrito* se obtiene el lema *perro*.
- **LISA**: (Localization Industry Standards Asociation) organización enfocada en la definición de estándares para la localización de contenidos.
- **Localización**: proceso de adaptar un software para una región específica de forma tal que ciertos elementos, tales como fechas, horas y monedas, se visualicen de acuerdo a las costumbres locales.
- **Machine translation**: ver traducción automática.
- **Medida de similaridad**: valor numérico que ayuda a determinar que tan

parecidos son dos segmentos según algún criterio predefinido

- **Memoria de traducción**: repositorio de información que contiene segmentos de texto en al menos dos lenguajes diferentes cada uno.
- **MMT**: *Machine Managed Translation* (Sistemas de traducción asistida por computadora), sistemas informáticos diseñados para asistir a las personas en la traducción de documentos.
- **N-grama**: secuencia de n elementos consecutivos; los elementos pueden ser letras, palabras o sintagmas, entre otros; en la práctica, la N se sustituye por un valor numérico natural. Por ejemplo, si se consideran como elementos a las letras, y N=3, se obtienen 3-gramas (tri-gramas) haciendo secuencias de tres letras consecutivas (abc, aei, xyz.).
- **ODF**: *Open Document Format,* estándar actual de la industria para almacenamiento e intercambio de documentos.
- **Open Source**: movimiento que propone que los desarrolladores de software y de documentos pongan a disposición de los usuarios sus productos junto con todo el conocimiento necesario para que éstos puedan usarlos y modificarlos sin necesidad de solicitar expreso consentimiento previo y en la mayoría de los casos sin tener que comprar licencias explícitas.
- **OSCAR**: *Open Standards for Container/Content Allowing Reuse* (Estándares Abiertos para Contenedores/Contenidos Reusables), uno de los grupos de interés especial (SIG, Special Interest Group) de LISA.
- **Palabras genéricas**: palabras de un idioma que son muy comunes y por tanto no son útiles para categorizar o individualizar textos. Incluyen a los artículos, los pronombres y las preposiciones.
- **PO**: formato ampliamente utilizado, aunque antiguo, para representar textos en un idioma con sus respectivas traducciones en otro idioma.
- **PoS tagging**: proceso por el cual se le asigna a cada palabra una categoría gramatical. Por ejemplo, a *perro* le corresponde la categoría *sustantivo* mientras que a *negro* le corresponde la categoría *adjetivo*. Es un proceso complicado ya que depende del contexto y en algunos casos no es posible determinar la categoria apropiada; por ejemplo, la palabra *tapa* puede ser un *verbo* (segunda persona presente de *tapar*) o un sustantivo (objeto que sirve para tapar).
- **Proyecto**: conjunto de archivos que debe ser traducido, asociados a cierta información variable: cliente, traductores, estimaciones de costos, tiempos, etc.
- **Raíz**: parte de una palabra que es compartida por todas sus inflexiones; por ejemplo, *gat* es la raíz de *gato*, *gata*, *gatos* y *gatuno*.
- **Requerimientos funcionales**: requerimientos que describen las funcionalidades que debe proveer un sistema, y que por tanto definen la utilidad o usabilidad del mismo.
- **Requerimientos no funcionales**: requerimientos que hacen referencia a características que no describen las funcionalidades que el sistema debe prestar, sino otro tipo de cualidades que la afectan, como por ejemplo las tecnologías a usar.
- **Scanning**: ver escaneo.
- **SDK**: *Software Development Kit* (Juego de Desarrollo de Software), conjunto de herramientas específicas para desarrollar programas.

- **Segmentación**: proceso de dividir un texto en segmentos.
- **Segmento**: porción de texto que se debe traducir.
- **Servidor de aplicaciones**: software que pone a disposición de los usuarios diferentes aplicaciones, las cuales son accedidas por éstos desde diferentes puntos a través de una red.
- **Sistema de memoria de traducción**: software de asistencia a la traducción que trabaja segmentando el texto original en unidades de traducción, y a continuación, para cada una de dichas unidades, busca en una base de datos las posibles traducciones para que el traductor seleccione la más apropiada, o introduzca una nueva.
- **TMS**: *Terminology Management* System (Sistema de Gestión de Terminología), software o parte de él dedicado a gestionar vocabulario específico, pudiendo proveer para cada término facilidades tales como definiciones, traducciones a otros idiomas y referencias a otros términos.
- **TMX**: *Translation Memmory eXchange format* (Formato de Intercambio de Memorias de Traducción), formato de marcado basado en XML cuya finalidad es permitir intercambiar memorias de traducción entre diferentes sistemas.
- **Token**: cadena de caracteres sin espacios, que puede ser considerado una palabra con o sin significado real.
- **Tokenización**: identificación de los tokens que forman parte de un texto. Específicamente, se identifican las palabras del mismo, pudiendo considerar las contracciones como uno o dos palabras según el caso, pero siempre un token.
- **Traductor automático**: software que realiza la traducción de documentos aplicando reglas gramaticales y usando un diccionario bilingüe.
- **Translation unit**: ver unidad de traducción.
- **TU**: *Translation unit* (Unidad de Traducción), ver unidad de traducción.
- **UMTF**: *Unified Message/Translation Format* (Formato de Mensajes Unificado para la Traducción), estándar de representación de información, incluido memorias de traducción.
- **Unicode**: estándar que establece cómo se deben representar cada uno de los caracteres de la mayoría de los alfabetos usados alrededor del mundo, de forma que cualquiera de ellos pueda ser utilizado en un documento dado, independientemente del idioma utilizado.
- **Unidad de traducción**: porción de texto que puede ser traducido y almacenado en una memoria de traducción.
- **XLIFF**: *XML Localization, Interchange File Format* (Formato de Intercambio de Información para la Localización basado en XML), estándar que permite representar los diferentes tipos de documentos comunmente utilizados (DOC, HTML, TXT, etc) en una forma común, de manera tal que la estructura y el contenido de los mismos se almacena por separado.
- **XML**: *eXtensible Markup Language* (Lenguaje de Marcas Extensible), lenguaje amplio que permite registrar información de forma estándar, no ambigua y completamente estructurada.

7 Referencias

[1] Machine translation: state of the art, tendencies and user perspectives. Steven Krauwer, Universidad de Utrecht, Holanda. 1995

[2] – Nina Wacholder: Simplex Nps Clustered by Head, A method for Identifying Significant Topics in a Document. In Proceedings of the Workshop on the Computational Treatment of Nominals, pages 70–79, Montreal, Canada, October 1998. COLING-ACL.

[3] TransSearch: A free translation memory on the World Wide Web. Elliott Macklovitch, Michel Simard, Philippe Langlais, Laboratoire RALI, Université de Montréal, Montreal, Canada. 2000.

[4] How to write a TM system – Tim Foster, Sun Microsystems. 2004.

[5] A comparison of string distance metrics for name-matching tasks. William Cohen, Pradeep Ravikumar, Stephen Fienberg, Carnegie Mellum University, EEUU. 2003.

[6] Design and development of Translator's Workbench. Akshi Kumar, Accurapid Journal. 2005.

[7] – TransSearch: A free translation memory on the Worl Widw Web. Elliott Macklovitch, Michel Simard, Philippe Langlais, Laboratoire RALI, Université de Montréal, Montreal, Canada. 2000.

[8] Detecting Text Similarity over Short Passages: Exploring Linguistic Feature Combinations via Machine Learning. Vasileios Hatzivassiloglou , Judith L. Klavans, and Eleazar Eskin. Columbia University, New York, USA.

[9] La traducción automática y la memoria de traducción. Marjorie León, Servicio de Traducciones de la Organización Panamericana de la Salud. 2002.

[10] Una guía al TMX – Josu Gómez, Grupo Deli, Universidad de Desusto. 2001.

[11] Usability of String Distance Metrics for Name Matching Tasks in Polish. Jakub Piskorski y Marcin Sydow.

[12] N-gram similarity and distance. Grzegorz Kondrak. Department of Computing Science, University of Alberta, Edmonton, Canada.

[art-1] From IBM: Translation Manager/2. Language Industry Monitor 11, Setiembre-Octubre 1992.

[web-1] Explicación del algoritmo de Levenshtein, que además describe una interesante variante pensada para textos algo más extensos que palabras aisladas: http://itman.narod.ru/english/ir/faq/fzfaq_calc.html (04/04/2009).

[web-2] Sitio dedicado a la distancia de Levenshtein: http://www.levenshtein.net (04/04/2009)

[web-3] Más sobre la distancia de Levenshtein: http://www.merriampark.com/ld.htm (04/04/2009)

[web-4] Implementación de algunas funciones de tratamientos de textos del lenguaje PHP: http://cvs.php.net/viewvc.cgi/php-src/ext/standard/string.c?revision=1.445.2.14.2.74&content-type=text%2Fplain&pathrev=1.445.2.14.2.74 (04/04/2009)

[web-5] Sitio oficial de Translazione, desarrollador de SDL Trados: www.translationzone.com (04/04/2009)

[web-6] Sitio oficial de OmegaT: http://www.omegat.org (04/04/2009)

[web-7] Sitio oficial de Atril, desarrollador de DejaVu: www.atril.com (04/04/2009)

[web-8] Sitio oficial de Metatexis: http://www.metatexis.com (04/04/2009)

[web-9] Sitio oficial de Heartsome: http://www.heartsome.net (04/04/2009)

[web-10] Página de TransSearch, en el sitio oficial de TSRALI: http://www.tsrali.com/?UTLanguage=en (04/04/2009)

[web-11] Página de Transit, en el sitio oficial de Start: http://www.star-group.net/star-www/description/transit/star-group/eng/star.html (04/04/2009)

[web-12] Página dedicada a GetText, en la que se explica el formato PO, dentro del sitio oficial de GNU: http://www.gnu.org/software/gettext/manual/gettext.html#PO-Files (04/04/2009)

[web-13] Descripción del proceso de transformar un documento a XLIFF, realizado por Rodolfo Raya: http://www.maxprograms.com/articles/xliff.html (04/04/2009)

[web-14] Descripción de XLIFF en el sitio de Sun Developer Network:

http://developers.sun.com/dev/gadc/technicalpublications/articles/xliff.html (04/04/2009)

[web-15] Explicación concreta de la utilización de XLIFF: http://www.mercury-online.com/es/tecnologias/xliff.html (04/04/2009)

[web-16] Sitio oficial de Freeling: http://garraf.epsevg.upc.es/freeling (04/04/2009)

[web-17] Sitio oficial de OpenNLP: http://opennlp.sourceforge.net (04/04/2009)

[web-18] Sitio oficial de UIMA: http://incubator.apache.org/uima (04/04/2009)

[web-19] Sitio oficial de File2XLIFF4J: http://file2xliff4j.sourceforge.net (04/04/2009)

[web-20] Sitio oficial de OLT: https://open-language-tools.dev.java.net (04/04/2009)

[web-21] Sitio oficial de TMX Editor: http://tmx-editor.sourceforge.net (04/04/2009)

[web-22] Página dedicada a la obtención de las raíces de las palabras: http://xapian.org/docs/stemming.html (04/04/2009)

[web-23] Sitio oficial de Wordnet: http://wordnet.princeton.edu (04/04/2009)

[web-24] Página de Wikipedia en la que se explica el algoritmo de Jaccard: http://en.wikipedia.org/wiki/Jaccard_index (04/04/2009)

[web-25] Página en Wikipedia en la que se explica la búsqueda de texto utilizando comparación difusa: http://en.wikipedia.org/wiki/Fuzzy_string_searching (04/04/2009)

[web-26] Página de Wikipedia en la que se explica la distancia de Jaro-Wrinkler: http://en.wikipedia.org/wiki/Jaro-Winkler_distance (04/04/2009)

[web-27] Sitio oficial de SecondString: http://secondstring.sourceforge.net (04/04/2009)

[web-28] Sitio oficial de Pootle: http://translate.sourceforge.net/wiki/pootle/index (04/04/2009)

[web-29] Portal para la localización de OpenOffice utilizando Pootle: http://www.sunvirtuallab.com:32300 (04/04/2009)

[web-30] Sitio oficial de Virtaal: http://translate.sourceforge.net/wiki/virtaal/index (04/04/2009)

[web-31] Sitio oficial de OpenTran: http://open-tran.eu (04/04/2009)

[web-32] Sitio oficial de WordFast: http://www.wordfast.net (04/04/2009)

[web-33] Sitio oficial de Mootools: http://mootools.net (04/04/2009)

[web-34] Sitio oficial de Mocha: http://mochaui.com (04/04/2009)

[web-35] Sitio oficial de JSon http://www.json.org (04/04/2009)

[web-36] Página dedicada a la tecnología EJB dentro del sitio oficial de Sun Microsystems: http://java.sun.com/products/ejb (04/04/2009)

[web-37] Página dedicada a la tecnología EJB dentro del sitio oficial de Jboss: http://docs.jboss.org/ejb3/app-server/tutorial (04/04/2009)

[web-38] Página dedicada a la tecnología JSF dentro del sitio oficial de Sun Microsystems: http://java.sun.com/javaee/javaserverfaces (04/04/2009)

[web-39] Sitio oficial de Glassfish: https://glassfish.dev.java.net (04/04/2009)

[web-40] Sitio oficial de MySQL: http://mysql.com (04/04/2009)

[web-41] Sitio oficial de Apache Struts: http://struts.apache.org (04/04/2009)

[web-42] Página dedicada a la tecnología AJAX: http://www.librosweb.es/ajax (04/04/2009)

[web-43] Página de Wikipedia dedicada al patrón de diseño MVC: http://en.wikipedia.org/wiki/Model-view-controller (04/04/2009)

[web-44] Página dentro del sitio de Freeling que explica como lograr utilizarlo en Microsoft Windows: http://garraf.epsevg.upc.es/freeling/index.php?option=com_simpleboard&Itemid=55&func=view&id=333&catid=5 (04/04/2009)

[web-45] Otra página que explica como usar Freeling en Microsoft Windows: http://www.smo.uhi.ac.uk/~oduibhin/oideasra/interfaces/winfreeling.htm (04/04/2009)

[web-46] Sitio oficial del Penn Treebank: http://www.cis.upenn.edu/~treebank (04/04/2009)

[web-47] Página que explica como construir la librería para Java de Freeling a partir del código fuente: https://

lafarga.cpl.upc.edu/plugins/scmcvs/cvsweb.php/freeling/APIs/java/README.diff?r1=1.7;r2=1.5;cvsroot=freeling (04/04/2009)

[web-48] Sitio oficial de SimMetrics: http://www.dcs.shef.ac.uk/~sam/simmetrics.html (04/04/2009)

8 Listado de apéndices

A continuación se listan los apéndices que acompañan este documento.

- **[ap-sofisglosario]** Documento que describe el sistema Sofis Glosario, un prototipo de un sistema de memorias de traducción realizado por Sofis Solutions.
- **[ap-estado-del-arte]** Documento que contiene un estudio del estado del arte con mayor detalle que el presentado en este documento.
- **[ap-diseño-arquitectura]** Documento que describe el proceso de diseño del sistema implementado.
- **[ap-pruebas]** Documento que resume los resultados de las pruebas comparativas realizadas con los algoritmos de generación de candidatos implementados.
- **[ap-testing]** Documento que resume los resultados de las pruebas realizadas de las funcionalidades del sistema.
- **[ap-manual-instalación]** Manual de instalación del sistema construido.
- **[ap-manual-usuario]** Manual de usuario del sistema construido.