- Tesis de Licenciatura en Ciencias de la Computación-

Normalización del texto de entrada para un sistema de síntesis del habla.

Alumna: Verónica Pechersky (vpechers@dc.uba.ar)
Universidad de Buenos Aires

Director: Dr. Agustín Gravano (gravano@dc.uba.ar)
Universidad de Buenos Aires

Diciembre de 2012

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Normalización del texto de entrada para un sistema de síntesis del habla

Resumen

La conversión del texto al habla (TTS, del inglés text-to-speech), consiste en transformar un texto escrito en su equivalente oral. Un sistema usado con este propósito recibe el nombre de sistema TTS. La primera fase de un sistema TTS es el preprocesamiento o normalización del texto de entrada, proceso que convierte el texto en palabras pronunciables como serían leídas por un hablante humano (Ej.: *el 16/12/2012 = el dieciséis de diciembre de dos mil doce*). Esta tesis busca implementar la normalización del texto de entrada para textos en español para un sistema TTS.

En este contexto es posible encontrar numerosos problemas a resolver, como por ejemplo la expansión de números (*124 = ciento veinticuatro*) y abreviaturas (*etc. = etcétera*). Para las múltiples tareas de la normalización del texto se utilizaron traductores de estados finitos con peso (en inglés *weighted finite-state transducers* o WFST). El objetivo fue construir un WFST para que reciba cualquier texto de entrada y devuelva otro texto escrito con un conjunto de traducciones posibles como serían leídas en su forma oral para seleccionar la que mejor aplique. Esta solución se selecciona en base al peso, beneficiando con el uso de marcas (tags) que permitieron indicar según información de contexto cual sería la traducción correcta (Ej.: *<GRADOS>1°</GRADOS> de temperatura = un grado de temperatura*).

El sistema fue evaluado utilizando fuentes de textos en español, como diarios on-line, artículos de la Web y Wikipedia, comparando los resultados obtenidos con textos de referencias donde las traducciones fueron realizadas por humanos.

Una aplicación futura de este sistema es que forme parte de un sistema TTS en desarrollo en la UBA, el cual permitirá la lectura de páginas Web para personas con problemas visuales, brindando la posibilidad a dichas personas de acceder a los contenidos de las páginas en español.

Agradecimientos

A mi familia, mis hijos y esposo, por todo el tiempo que permitieron que le dedique a la tesis.

A Agustín Gravano, que por su excelencia académica y calidad humana siempre me motivó, me guió y me contagió el interés en los temas de investigación de la tesis e hizo que fuera un honor tenerlo como director.

A Pablo Carbajo, mi amigo y alumno de esta facultad, que siempre estuvo atento y disponible cuando necesitaba su apoyo y que con su inteligencia y sabiduría hizo grandes aportes para esta tesis.

A Diego Valenti, mi compañero y egresado de esta facultad, que con su ayuda y por todos los TPs que hicimos juntos, casi sin darse cuenta me dio un empujón inicial para comenzar.

A mis amigos y familiares Roxana, Silvina, Pablo, Facundo, Rodolfo, Patricia, Marcelo y Felicitas, que dedicaron su tiempo para escribir las traducciones usadas como referencias y me brindaron una ayuda enorme.

A los profesores de la facultad, que transmiten su pasión por esta carrera.

A todos los que les pregunté alguna vez "¿Cómo leerías esto?", que aportaron mucho con sus respuestas y hasta a veces hasta los dejé pensando.

A todos los que me preguntaban "y ¿Cómo vas con la tesis?", que sin querer me daban una palmadita en la espalda.

A mi mamá, que siempre me alentó a seguir adelante y que seguramente me estaría diciendo "¡Todo llega!".

1 Capítulo I – Introducción

La conversión del texto al habla (TTS, del inglés text-to-speech), consiste en transformar un texto escrito en su equivalente oral. Un sistema usado con este propósito recibe el nombre de sistema TTS. La primera fase de un sistema TTS es el preprocesamiento o normalización del texto de entrada, proceso que convierte el texto en palabras pronunciables como serían leídas por un hablante humano (Ej.: *el 16/12/2012 = el dieciséis de diciembre de dos mil doce*). Este proyecto busca implementar la normalización para textos en español, dado que hasta el momento no se conocen herramientas de libre acceso que realicen esta tarea en nuestro idioma.

En este contexto, es posible encontrar numerosos problemas a resolver en dicha conversión. Algunos de ellos son la expansión de abreviaturas (*etc. = etcétera*), siglas (*DC = de cè*) y acrónimos (*UBA = uba*), y las conversiones numéricas, como naturales (*1600 = mil seiscientos*), fechas (*04/10/2010 = cuatro de octubre de dos mil diez*), horas (*04:10 = cuatro horas y diez minutos*), importes (*$410 = cuatrocientos diez pesos*) y números romanos (*IV = cuatro o cuarto*).

En el siguiente ejemplo, se muestra cómo sería la conversión de números según su significado:

- Texto de entrada: "El vuelo *257* sale el *1/10/2009* a las *21:50*"
- Texto preprocesado: "El vuelo *doscientos cincuenta y siete* sale el *primero de octubre de dos mil nueve* a las *veintiuna horas y cincuenta minutos*".

1.1 Metas

Para las múltiples tareas de normalización se propone utilizar traductores de estados finitos (en inglés *finite-state transducers* o FST), y sus variantes con pesos (weighted FST o WFST). Los traductores ofrecen flexibilidad para la representación de los distintos problemas a resolver: dado un texto de entrada permiten "traducirlo" a su correspondiente texto preprocesado. Por ejemplo, es relativamente sencillo construir FSTs específicos para convertir textos al texto equivalente en su forma oral siglas (*"DNI"* a *"de ene i"*) o números romanos (*"XIV"* a *"catorce"*). Sin embargo, la dificultad del problema aparece cuando se permite cualquier texto de entrada, con tasas altas de ambigüedad, debido a que no es trivial determinar qué FST aplicar en cada parte del texto. En el ejemplo anterior "El vuelo *257* sale el *1/10/2009* a las *21:50*", los números se deberían convertir usando distintos FSTs: *"257"* a *"doscientos cincuenta y siete"*, *"1"* a *"primero"*, *"21"* a *"veintiuna"*. Es entonces que los WFST entran en escena, asignando pesos a las diferentes traducciones, de modo que sea elegida la traducción que mejor aplique. Por ejemplo, para *"$40"* pueden existir dos traducciones posibles, *"cuarenta pesos",* o bien *"cuatro pesos cero"*, siendo esperable que la primera versión tenga un peso mayor en el WFST.

La principal meta consiste, entonces, en construir un WFST que reciba cualquier texto de entrada, devuelva otro texto con un conjunto de traducciones posibles equivalentes en su forma oral, cada una con su peso correspondiente. Esta meta impone los siguientes desafíos: en primer lugar, la construcción manual de traductores para atacar los numerosos problemas individuales; y en segundo lugar, la combinación de los traductores, asignando pesos en base al contexto. Esta última tarea resulta demasiado compleja para ser realizada manualmente, por lo que se la automatizará mediante técnicas de optimización. En otras palabras, se buscará la asignación de pesos del WFST que maximice la performance del sistema. Esta solución se podrá beneficiar de la existencia de un

previo reconocimiento de entidades (Ej. <IMPORTES>$40</ IMPORTES>), para orientar la traducción a realizar.

La evaluación del sistema se desarrollará sobre datos reales. Se elegirán oraciones al azar, se convertirán manualmente a su correspondiente texto preprocesado y se verificará que el sistema devuelva el mismo resultado. Para el desarrollo y evaluación del sistema, se utilizará un corpus de textos en español compuesto por diarios on-line y artículos de Wikipedia y otros de la Web.

1.2 Contribuciones

El módulo desarrollado será de público acceso y formará parte de una serie de módulos para un futuro sistema TTS en desarrollo en el Departamento de Computación (DC) de la Facultad de Ciencias Exactas (Universidad de Buenos Aires). Una de las aplicaciones posibles de este sistema es la lectura de páginas Web para personas con problemas visuales, brindado la posibilidad de que dichas personas de acceder a los contenidos de las páginas en español.

Los resultados del presente trabajo podrán ser publicados en conferencias regionales, para describir y detallar cómo construir este módulo desde cero. En primer lugar, ofrecerá a la comunidad académica la posibilidad de profundizar y/o ampliar el tema en futuras investigaciones. En segundo lugar, ofrecerá a la industria una herramienta útil para el desarrollo de aplicaciones de procesamiento del habla.

1.3 Estructura de la tesis

La tesis se encuentra dividida en 6 capítulos y 4 apéndices. En el resto de este capítulo se presentan generalidades de la teoría del preprocesamiento o normalización del texto de entrada dentro del contexto de un sistema de síntesis del habla, sus distintas problemáticas y desafíos para resolverlas. También se presentan metodologías para la normalización del texto, como texto de entrada sin tags y sin reconocimiento de entidades, o bien, texto con tags, identificación de palabras y el enfoque de post procesamiento con modelo de lenguaje, para resolver casos ambiguos.

En el **capítulo 2** se describe el Open-FST y las operaciones utilizadas de la librería para *traductores de estados finitos* (en inglés *finite-state transducers*, o FST), y sus variantes con pesos (weighted FST, o WFST).

En el **capítulo 3** se explica el desarrollo implementado para la solución propuesta al preprocesamiento del texto de entrada (o normalización del texto).

En el **capítulo 4** se detalla la evaluación y el test de la solución implementada, los pasos para realizarla, los resultados obtenidos, el análisis y las conclusiones de las traducciones realizadas por el sistema.

Finalmente en el **capítulo 5** se discuten los comentarios finales de los resultados obtenidos junto con los temas de investigación abiertos y para trabajo futuro.

En el **apéndice A** se brindan los detalles técnicos de desarrollo de la solución implementada y en el **apéndice B** se detallan los resultados de los test previos a la evaluación y herramientas utilizadas durante las misma.

En los **apéndices C y D** finales se incluyen el glosario para describir términos específicos y las referencias a la bibliografía utilizada.

1.4 Sistemas de texto-a-habla

Como se mencionó anteriormente, la síntesis del texto al habla, en inglés *text-to-speech* (TTS), es un proceso donde se convierte un texto escrito a su equivalente oral. Un sistema usado con este propósito recibe el nombre de sistema TTS. Este proceso [JUR 09] es realizado en las fases que se indican en el diagrama de la Figura 1.1:

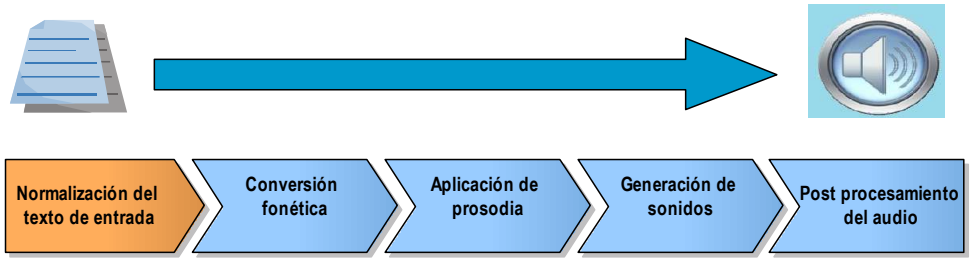


Figura 1.1 – Fases de la síntesis del texto al habla

La primera fase, también llamada normalización del texto, convierte el texto en palabras pronunciables como serían leídas por el humano (Ejemplo: *el 4º piso = el cuarto piso*). La siguiente fase, es la asociación de sonidos para las palabras definidas produciendo una secuencia de fonemas (Ejemplo: *hoguera = /ogera/*). A continuación se seleccionan los atributos prosódicos, incluyendo entonación, velocidad y volumen entre otros, para evitar que la voz suene monótona o "robotizada". Acto seguido, se generan los sonidos correspondientes, donde se convierte los fonemas y su prosodia en el audio digital. Por último, se aplican filtros acústicos para mejorar la calidad del audio resultante [MAN 01].

Como inicio del proceso de síntesis del habla se toma el texto en algún formato: plano, HTML, email, etc. y como finalización de dicho proceso se obtiene un archivo de audio asociado al texto correspondiente. Asimismo, se debe considerar que la escritura es una representación imperfecta e incompleta del lenguaje hablado. En la escritura se presume que el lector conoce el lenguaje en cuestión y puede completar los detalles cuando la información provista en la misma es incompleta o confusa. También se debe tener en cuenta el hecho de que el texto escrito no tiene representación sistemática de prosodia [SPR 08].

La entrada básica de un sistema TTS es un texto codificado en la ortografía estándar del lenguaje a ser sintetizado (en el caso del actual trabajo el texto se encuentra en español). Por otro lado, es importante tener en mente que el texto escrito puede tener alguna codificación de entonación (por ejemplo signos de admiración ¡!) o alguna otra información lingüística (guiones para indicar citas), no necesariamente fácilmente recuperable para el sistema TTS. La escritura está diseñada para personas que conocen el lenguaje y pueden completar los detalles del texto que se encuentran implícitos. Luego es un desafío para los sistemas TTS modelar el lenguaje que el hablante nativo conoce.

En el presente trabajo se desarrollará la primera etapa, la normalización del texto de entrada para textos en español, dado que hasta el momento no se encuentran herramientas de libre acceso que realicen esta tarea en este idioma. No obstante, existen unos pocos estudios previos que aportan ideas para la elaboración de este proceso [BON 00] [JIM 99] [MON 03].

1.4.1 Normalización del texto de entrada

La **normalización del texto de entrada** es el componente del sistema TTS que se encarga de analizar el mismo y devolver la traducción del texto en otro texto con su equivalente en su forma oral. Para modelar la tarea realizada por este componente, en la Figura 1.2 se muestra un ejemplo donde se encuentran resaltadas las distintas traducciones a realizar En dicho ejemplo se pueden observar las traducciones necesarias para expresiones numéricas como **naturales** (*2011 = dos mil once*), **ordinales** (*1° = primer*) y **romanos** (*XXI = veintiuno, II = dos*), **fechas** (*3/02/11 = tres de febrero de dos mil once*, *4/03/11 = cuatro de marzo de dos mil once*), **siglas** (*CBC = ce be ce*) y **abreviaturas** (*etc. = etcétera*).

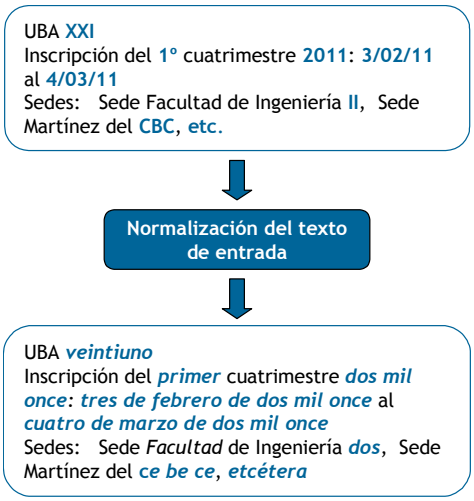


Figura 1.2 – Normalización del texto de entrada

Si bien en el capítulo de desarrollo (Ver Capítulo III) se explica cada una de las traducciones realizadas para esta etapa por el sistema implementado, aquí se explican brevemente algunos casos típicos de la normalización del texto de entrada como la **expansión** de **abreviaturas** y de **números** para introducir conceptos que deben ser tenidos en cuenta en esta etapa.

En el caso de las **abreviaturas**, primero se debe identificar la abreviatura, considerando que una palabra seguida de un punto no siempre se trata de una abreviatura sino que también indica fin de frase (Ej. la precede un espacio). Luego la expansión de abreviaturas se puede realizar buscando la traducción en un listado (*etc. | etcétera, Bs.As. |Buenos Aires*). También se debe tener en cuenta que existen casos ambiguos que deben ser cuidadosamente identificados según el contexto (*1 kg.|un kilo (singular), 2 kg.|dos kilos (plural), Lic.|Licenciado / Licenciada (género)*). Otro tema a considerar es que la lista de abreviaturas se incrementa frecuentemente con lo que no siempre es posible que se incluyan todas ellas, en particular, en el presente trabajo se incluye una lista general basada en la Real

Academia Española e incrementada con otras usualmente usadas en textos en español argentino y quedan fuera de alcance las abreviaturas relacionadas con clasificados, recetas, etc. En el capítulo de desarrollo se explica el manejo de la lista de abreviaturas, que es realizada por el uso de un archivo de configuración que las incluyen.

Con respecto a la expansión de **números**, como se puede observar en el ejemplo, existen varias maneras de traducir los mismos de acuerdo al caso que se trate según su representación y su contexto (natural, ordinal, fecha, etc.). Una vez que se conoce el tipo correspondiente a realizar es relativamente simple la expansión. Una manera de expandir números típicamente usada en sistemas TTS sería por reglas [SPR 08]. Otro enfoque posible, es el uso de WFST, que se desarrolla en el presente trabajo, donde se elige la solución más adecuada según el peso (Ver Capítulo III). En el caso del lenguaje español, existen casos como el género (*21 hs. = veintiuna horas*) que agregan una complejidad adicional a la hora de analizar cual sería la traducción correcta, entre las opciones posibles (*21 = veintiuno o veintiuna*), ya que la misma depende del contexto en donde se encuentre el número a traducir.

Existen otras entidades en el lenguaje español que requieren su traducción (Ej.: romanos, siglas, grados, teléfonos, etc.). Las mismas son explicadas en el capítulo de desarrollo (Ver Capítulo III), donde se indica cada una de las traducciones cubiertas para esta etapa junto a los detalles correspondientes para dichas traducciones.

1.4.2 Soluciones para la normalización del texto

Existen numerosas maneras de implementar soluciones para resolver las traducciones de texto escrito a otros textos con su equivalente oral, tal como herramientas desarrolladas con este propósito específico, algoritmos que implementan en el código (Ej: C++) las reglas de traducción, esquemas Open-Source (Ej: sistema Festival [FES 01]), etc. [SPR 08]. Un enfoque que ha ganado popularidad son los métodos basados en *traductores de estados finitos con peso* (WFST = *Weighted Finite-State Transducers*). Los WFST ofrecen un marco común para manejar una amplia variedad de problemas de traducción. A través de operaciones como unión, concatenación, clausura y composición es posible construir fácilmente los componentes para resolver cada tipo de traducción. En el siguiente capítulo se explica en mayor detalle las operaciones mencionadas (Ver Capítulo II).

En el presente trabajo se presentará esta solución para realizar la tarea de la traducción del texto escrito a otro texto con su equivalente de cómo se leería en voz alta. En el capítulo de desarrollo se explica los detalles de la implementación de dicha solución (Ver Capítulo III). Luego, en el capítulo de evaluación se realiza un análisis de las ventajas y desventajas del uso de este enfoque para realizar las traducciones en el lenguaje español (Ver Capítulo IV).

1.4.3 Resolución de casos ambiguos

Existen numerosos casos donde en los sistemas TTS se encuentra el problema de desambiguar el sentido de una palabra (Ejemplo: Talle XL / XL Aniversario), ya que varias traducciones son posibles para la misma (*XL = equis ele* si es una sigla o *cuarenta* si se trata de un número romano). Luego, dada que varias interpretaciones pueden ser válidas y no hay manera sin información adicional de elegir cuál es la más apropiada, el desafío es resolver estos casos ambiguos considerando el contexto en que se encuentran.

El escritor de un texto conoce cuál es el sentido de cada palabra que expresa. Sin embargo cuando lo está produciendo, lo hace a través de una representación ortográfica donde distintos sentidos de una palabra pueden aplicar. Es la tarea del lector (o del algoritmo TTS) reconstruir el sentido original de la misma [SPR 08].

La desambiguación automática es una importante área de la investigación en el procesamiento del lenguaje natural, con numerosas recetas y desarrollos de corpus y recursos tal como Wordnets en varios lenguajes. Este es un problema esencial de modelado del lenguaje. Para manejar estos casos, se usan típicamente **modelos estadísticos del lenguaje**, tomando en cuenta las probabilidades de la aparición de las palabras involucradas. También son usadas técnicas de **categorización en clases de palabras** indicadas con marcas especiales, como se describe a continuación.

Entonces, una de las técnicas que ayudan a mejorar el pre-procesamiento es la de indicar (en esta tesis, manualmente) las categorías de palabras que correspondan para seleccionar la mejor solución. Ejemplo:

- <SIGLA>XL</SIGLA> = equis ele
- <ROMANO_NAT> XL </ROMANO_NAT > = cuarenta

Luego estas marcas (tags) favorecen a la selección de la solución correcta. Esta es la solución abordada en el presente trabajo. En el capítulo de desarrollo se encuentran los detalles de implementación de la misma (Ver Capítulo III) y en el capítulo de evaluación se muestra cómo dicha solución favorece a la elección de las traducciones correctas (Ver Capítulo IV).

Por último, otra técnica útil y en algunos casos necesaria, también utilizada en el presente trabajo, es preprocesar ciertas palabras para llevarlas a un formato definido para el TTS (Ejemplo fechas: 1/V/2012 = 1/5/2012). En el capítulo de evaluación se indican los casos donde se debe realizar un preprocesamiento de palabras (Ver Capítulo IV).

2 Capítulo II - OpenFST

2.1 Introducción

En el presente trabajo se propone utilizar *traductores de estados finitos* (en inglés *finite-state transducers*, o FST), y sus variantes con pesos (weighted FST, o WFST).

Los WFSTs han sido usados en áreas de reconocimiento y síntesis del habla, traducción automática, reconocimiento óptico de caracteres, reconocimiento de patrones (pattern matching), procesamiento de cadenas, aprendizaje automático, recuperación y extracción de información, entre otras.

Los WFTs se encuentran formados por estados y transiciones (arcos). Las transiciones contienen etiquetas de entrada, etiquetas de salida y sus pesos [ALL 07]. A continuación en la figura 2.1 se muestra un ejemplo donde se pueden observar cada uno de los conceptos mencionados. En las siguientes secciones y capítulos se explicará la utilización de los mismos en el presente trabajo.

Ejemplo:

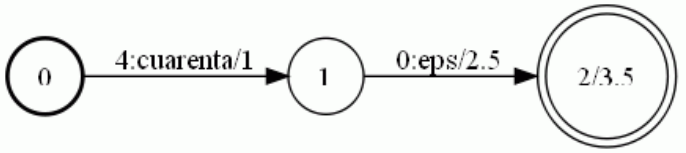


Figura 2.1 – Definición de WFST

El estado inicial es el 0, el estado final es el 2 con un peso de 3.5. Existe una transición desde el estado 0 (inicial) al estado 1, con la etiqueta de entrada "4", la etiqueta de salida "cuarenta" con un peso de 1. Existe otra transición desde el estado 1 al estado 2 (final), con la etiqueta de entrada "0", la etiqueta de salida "epsilon" (cadena vacía) con un peso de 2.5. Los pesos de las transiciones y del estado final serán utilizados para conocer el peso del camino recorrido desde el estado inicial al final siguiendo dichas transiciones. El estado final es el único estado que tiene peso ya que el mismo será utilizado si se realizan operaciones de clausura que se explicarán mas adelante en el presente capítulo. Las etiquetas de estados son enteros de 32-bit y los pesos son punto flotante precisión simple debido a que en el presente trabajo se utilizan pesos del tipo Tropical Weight [ALL 07] [FST 05].

2.2 OpenFst: Definición, representación y construcción

OpenFst es una librería que se encuentra desarrollada en C++ para la construcción, combinación, optimización y búsqueda en los WFSTs. [FST 01]. Es una librería compresible y flexible, donde el código se puede adquirir libremente. Se encuentra disponible en http://www.openfst.org . [GRA 00]

En la librería OpenFst, un WFST puede ser construido desde el código C++ usando las clases constructoras, o bien a través de la línea de comando (shell-level) usando una representación de

archivo de texto plano. En el presente trabajo se utilizó la librería desde código disponible, agregando las extensiones para generar las traducciones requeridas.

Para visualizar el resultado de los WFST obtenidos, se pueden utilizar operaciones del OpenFST para generarlos en formato de texto, o bien, en formato gráfico y luego utilizar herramientas para mostrarlos [GRA 00]. Las mismas fueron utilizadas para los ejemplos que se incluyen a continuación.

2.2.1 Construcción de WFST – Línea de comando (shell-level)

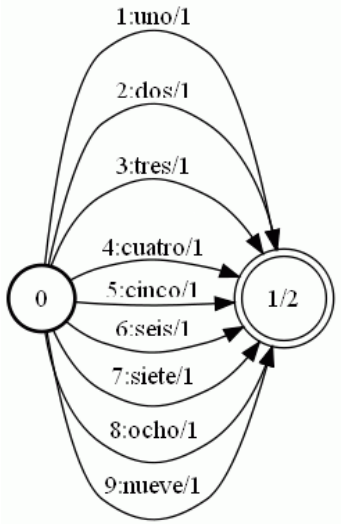
Dado que es muy simple de utilizar y para presentar el tema con un ejemplo, a continuación se describen los pasos / componentes utilizados para crear y visualizar un WFST desde la línea de comando.

Descripción:	**Ejemplo**
Archivo de entrada del WFST Archivo con los arcos del WFST. Formato: Estado desde / Estado hasta / Símbolo de entrada / Símbolo de Salida / Peso del arco. El estado final se indica con su peso. * Ejemplo: input.txt	0 1 1 uno 1.0 0 1 2 dos 1.0 0 1 3 tres 1.0 0 1 4 cuatro 1.0 0 1 5 cinco 1.0 0 1 6 seis 1.0 0 1 7 siete 1.0 0 1 8 ocho 1.0 0 1 9 nueve 1.0 1 2.0
Símbolos de Entrada Archivo con los símbolos de entrada. El símbolo numerado como 0 debe asociarse al valor "epsilon". El símbolo de la primera columna es representado internamente con el número indicado en la segunda columna. * Ejemplo: nat.isyms	eps 0 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10
Símbolos de Salida Archivo con los símbolos de salida. El símbolo numerado como 0 debe asociarse al valor "epsilon". El símbolo de la primera columna es representado internamente con el número indicado en la segunda columna. * Ejemplo: nat.osyms	eps 0 cero 1 uno 2 dos 3 tres 4 cuatro 5 cinco 6 seis 7 siete 8 ocho 9 nueve 10
Generación del WFST Compilar el archivo de entrada con los símbolos de entrada / salida. Se genera el WFST en un archivo binario. * Ejemplo: fstcompile -isymbols=nat.isyms -osymbols=nat.osyms -keep_isymbols -keep_osymbols input.txt nUnidad.fst	nUnidad.fst (archivo binario)
Impresión del WFST Visualizar en formato texto el WFST generado. * Ejemplo: fstprint nUnidad.fst nUnidad.txt	0 1 1 uno 1 0 1 2 dos 1 0 1 3 tres 1 0 1 4 cuatro 1 0 1 5 cinco 1 0 1 6 seis 1 0 1 7 siete 1

<table>
<tr><td></td><td>0 1 8 ocho 1
0 1 9 nueve 1
1 2</td></tr>
<tr><td>Diagrama del WFST

Visualizar en formato gráfico el WFST generado.

* Ejemplo:
fstdraw –portrait nUnidad.fst > nUnidad.dot

* Nota:
Luego de que el archivo .dot es generado, este debe ser visualizado con una herramienta para tal fin. (Ej.: Graphiz [GRA 00], dot2ps)</td><td> 
</td></tr>
</table>

2.2.2 Construcción de WFST – Código C++

A continuación se describe el ejemplo anterior para generar un WFST desde código C++.

```
// Declaración del vector fst
VectorFst<StdArc> fst;

//Agregar el estado 0 y definirlo como inicial
fst.AddState();
fst.SetStart(0);

// Agregar las transiciones desde el estado 0 al 1
// Argumentos de la transición:
// etiqueta de entrada, etiqueta de salida, peso, estado de destino
fst.AddArc(0, StdArc(1, 1, 1.0, 1)); //El primer argumento es el estado de origen
fst.AddArc(0, StdArc(2, 2, 1.0, 1));
fst.AddArc(0, StdArc(3, 3, 1.0, 1));
fst.AddArc(0, StdArc(4, 4, 1.0, 1));
fst.AddArc(0, StdArc(5, 5, 1.0, 1));
fst.AddArc(0, StdArc(6, 6, 1.0, 1));
fst.AddArc(0, StdArc(7, 7, 1.0, 1));
fst.AddArc(0, StdArc(8, 8, 1.0, 1));
fst.AddArc(0, StdArc(9, 9, 1.0, 1));

// Agregar estado 1 y definirlo como final con su peso
fst.AddState();
fst.SetFinal(1, 2); // 1st arg is state ID, 2nd arg weight
```

A continuación se describe un ejemplo para leer / escribir un WFST desde / hacia un archivo a través de código C++.

Lectura	`Fst<Arc> *fst = Fst<Arc>::Read(''T.fst'')`
Escritura	`fst.Write(''T.fst'')`

2.3 OpenFst: Operaciones

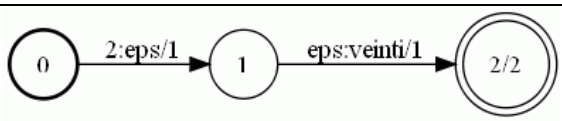
La librería provee más de veinticinco (25) operaciones que son generalmente empleadas para gestión de los WFSTs que pueden ser de utilidad en desarrollos particulares. Las siguientes son las operaciones del OpenFst utilizadas en el presente trabajo. [FST 00] [FST 02] [FST 03]:

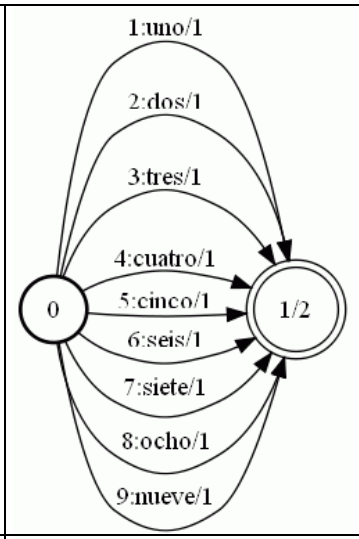
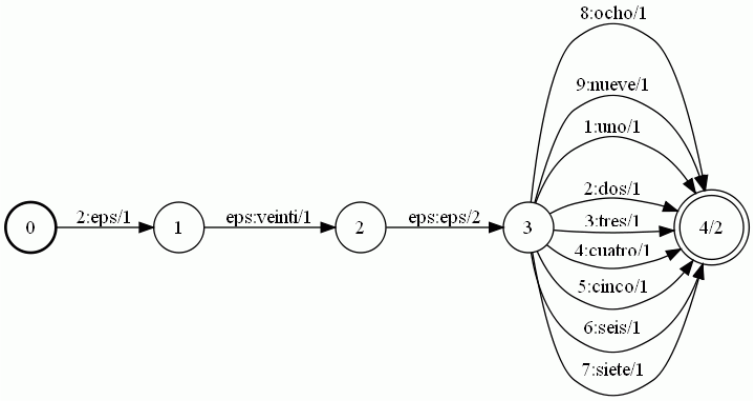
1. Concat (Producto)
2. Union (Suma)
3. RmEpsilon
4. Determinize
5. Minimize
6. Closure
7. Compose
8. ShortestPath

A continuación se detallan cada una de ellas, indicando la descripción de la operación, los casos donde se aplica dicha operación, sus diferentes usos de acuerdo a los objetos usados en el código para implementar los WFST intervinientes en cada operación (siendo siempre el último ítem el uso desde la línea de comandos) y también el cálculo de los pesos luego de aplicar las mismas.

2.3.1 Concat (Producto)

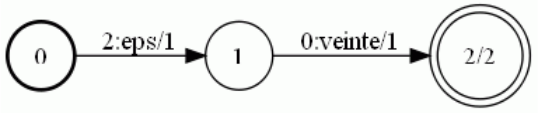
- **Descripción:** Contiene las cadenas de A seguidas de las de B (concatenación de cadenas).
- **Uso:**
  - Concat(&A, B);
  - Concat(A, &B);
  - ConcatFst<Arc>(A, B);
  - fstconcat a.fst b.fst out.fst
- **Aplicación:** Los WFST que traducen cadenas se encuentran modularizados. La concatenación se usa para unir cadenas previamente traducidas, una a continuación de la otra.
- **Peso:** Si *A* traduce la cadena *x* a *y* con peso *a* y *B* traduce la cadena *w* a *v* con peso *b*, entonces su concatenación traduce la cadena *xw* a *yv* con peso *a* + *b*.
- **Ejemplo:**

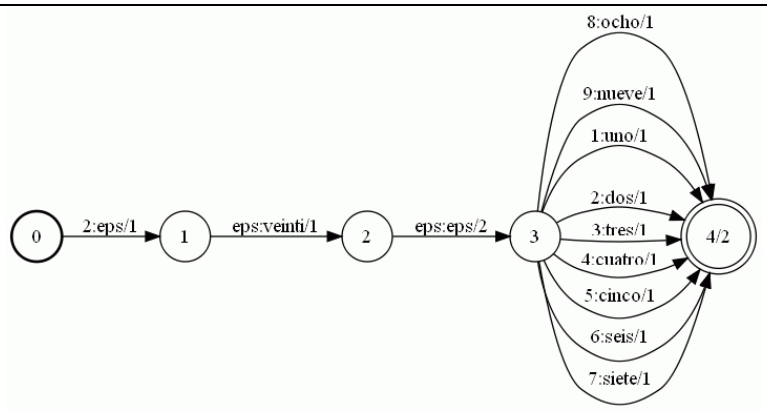
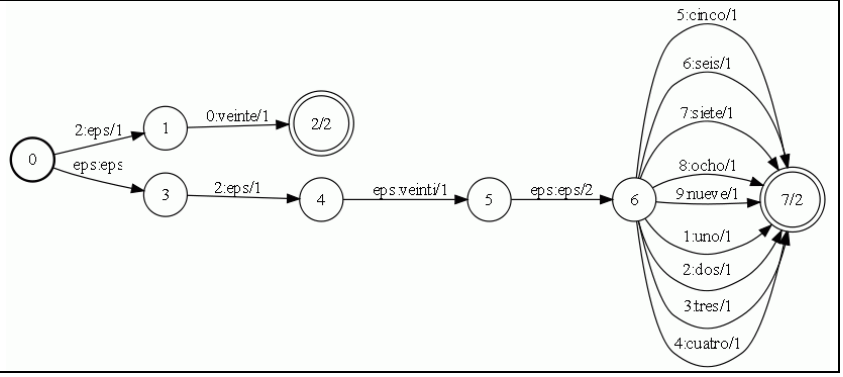
nVeinti.fst (A)	

`nUnidad.fst (B)`	
`Fstconcat nVeinti.fst nUnidad.fst outConcat.fst`	

2.3.2 Union (Suma)

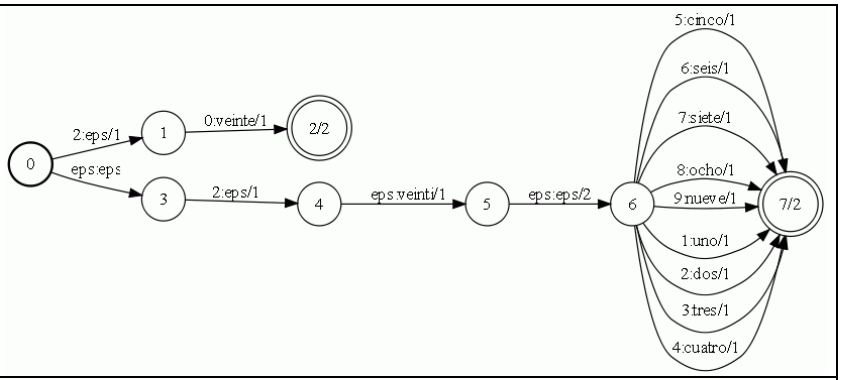
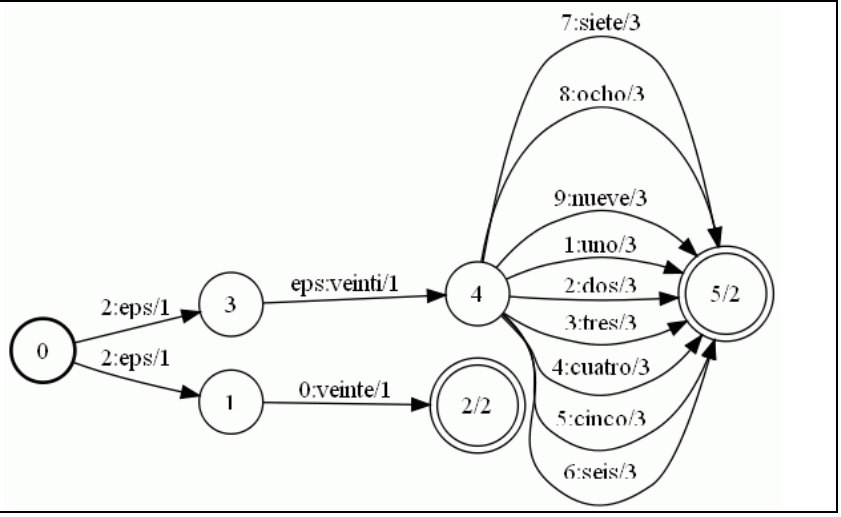
- **Descripción:** Contiene el conjunto de cadenas provenientes de hacer la unión de las cadenas de A y las de B.
- **Uso:**
  - Union(&A, B);
  - UnionFst<Arc>(A, B);
  - fstunion a.fst b.fst out.fst
- **Aplicación:** Se utiliza la unión para juntar cadenas previamente traducidas, para retornar las soluciones posibles dado un texto de entrada.
- **Peso:** Si *A* traduce la cadena *x* a *y* con peso *a* y *B* traduce la cadena *w* a *v* con peso *b*, entonces su unión traduce *x* a *y* con peso *a* y *w* a *v* con peso *b*.
- **Ejemplo:**

`nVeinte.fst (A)`	

`OutConcat.fst (B)`	
`fstunion nVeinte.fst outConcat.fst outUnion.fst`	

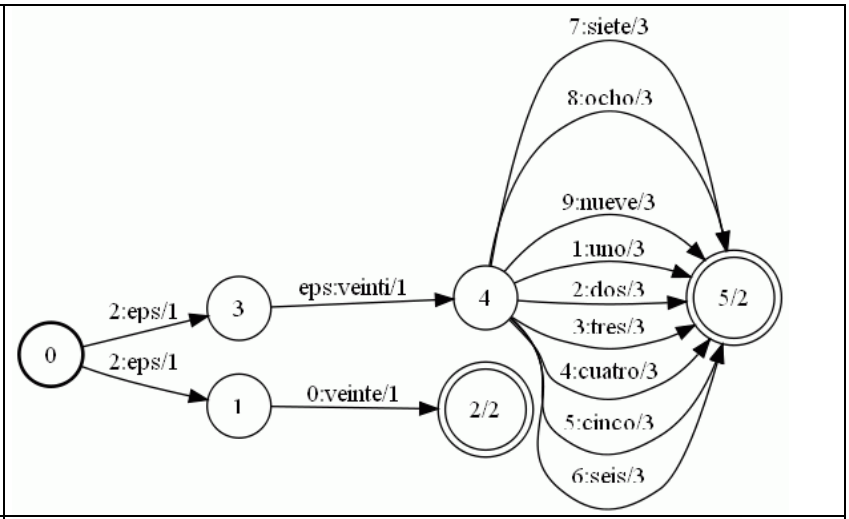
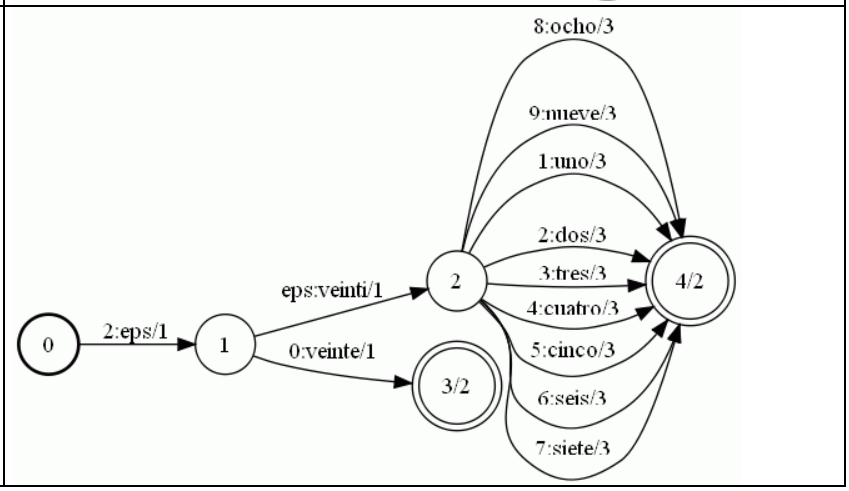
2.3.3 RmEpsilon

- **Descripción:** WFST equivalente al original con las transiciones "epsilon" removidas (ε-free transducer).
- **Uso:**
  - RmEpsilon(&A);
  - RmEpsilonFst<Arc>(A);
  - fstrmepsilon in.fst out.fst
- **Aplicación:** Se utiliza la operación para remover los arcos innecesarios que contienen traducciones ε:ε, para obtener WFST con menor cantidad de estados.
- **Peso:** Si *A* traduce la cadena *x* a *y* con peso *a* y se remueven sus transiciones "epsilon", los pesos de las transiciones removidas son sumados entre los estados y/o transiciones del WFST resultante, obteniendo el mismo peso del camino completo para traducir *x* a *y*.
- **Ejemplo:**

`outUnion.fst`	
`fstrmepsilon outUnion.fst outEps.fst`	

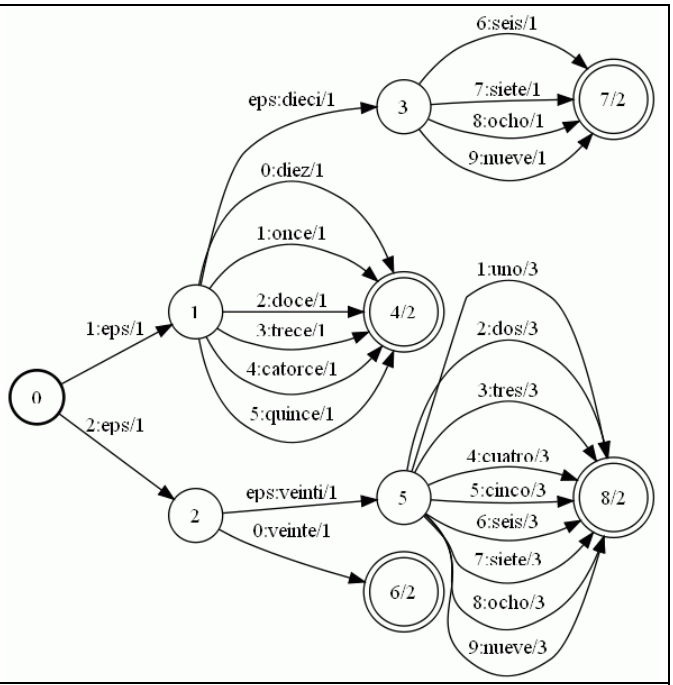
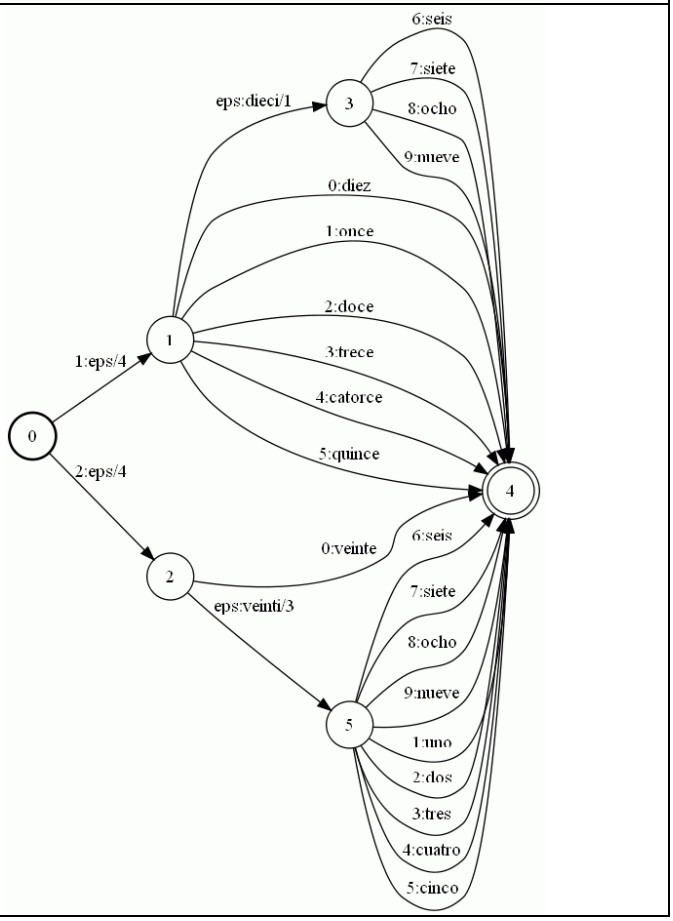
2.3.4 Determinize

- **Descripción:** WFST determinístico equivalente al original.
- **Uso:**
  - Determinize(A, &B);
  - DeterminizeFst<Arc>(A);
  - fstdeterminize in.fst out.fst
- **Aplicación:** Se utiliza la operación para determinizar en los casos que sea posible, para reducir la cantidad de arcos / estados a recorrer.
- **Peso:** Los pesos se distribuyen en las transiciones resultantes de manera tal que el peso de la traducción de las cadenas en el WFST original se mantengan en el WFST determinizado.
- **Ejemplo:**

`outEps.fst`	
`Fstdeterminize outEps.fst outDet.fst`	

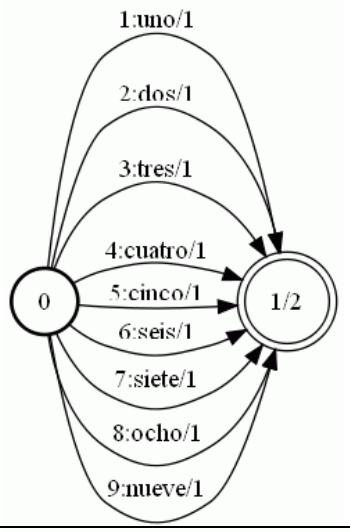
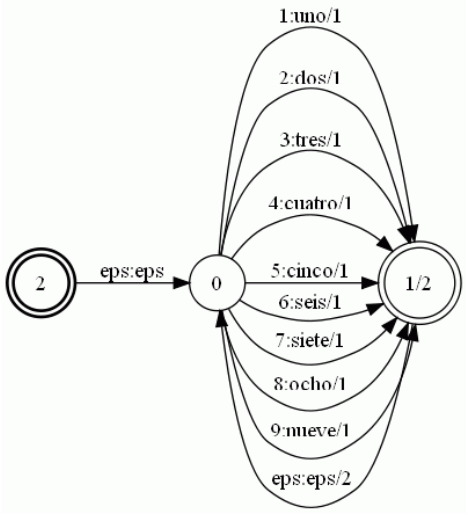
2.3.5 Minimize

- **Descripción:** WFST con la menor cantidad de estados posibles y que sea equivalente al original.
- **Uso:**
  - Mininize(&A);
  - Mininize(&A, &B);
  - fstminimize in.fst out1.fst [out2.fst]
- **Aplicación:** Se utiliza la operación para minimizar en los casos que sea posible, para reducir la cantidad de arcos / estados a recorrer.
- **Ejemplo:**

`nDecenas.fst`	
`fstminimize nDecenas.fst outMin.fst`	

2.3.6 Closure

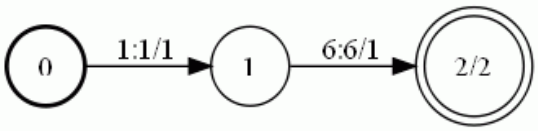
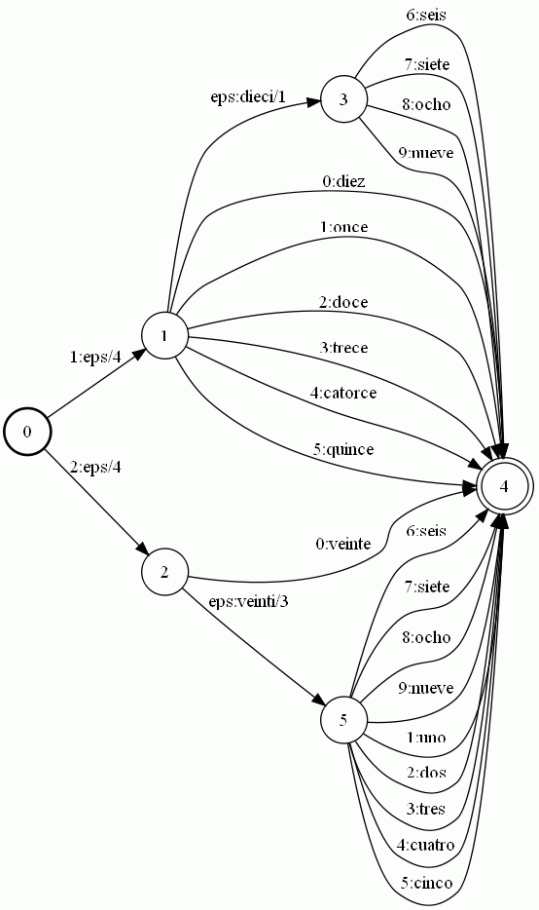
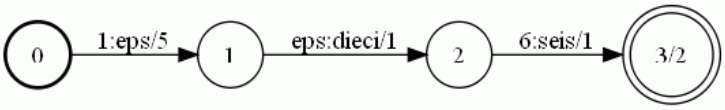
- **Descripción:** Clausura del WFST **(**A* = {ε } [A [AA [. . .)
- **Uso:**
  - Closure(&A, type);
  - ClosureFst<Arc>(A, type);
  - fstclosure in.fst out.fst
- **Aplicación:** Se utiliza la clausura para traducir una cadena iterando, para cada uno de los caracteres de dicha cadena, sobre el mismo WFST.
- **Peso:** Los pesos se distribuyen en las transiciones resultantes de manera tal que el peso de la traducción de las cadenas en el WFST original se mantengan en el WFST minimizado.
- **Ejemplo:**

`nUnidad.fst`	
`fstclosure nUnidad.fst outClosure.fst`	

2.3.7 Compose

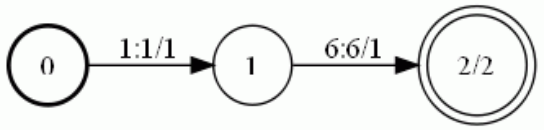
- **Descripción:** Composición de WFST (A o B).

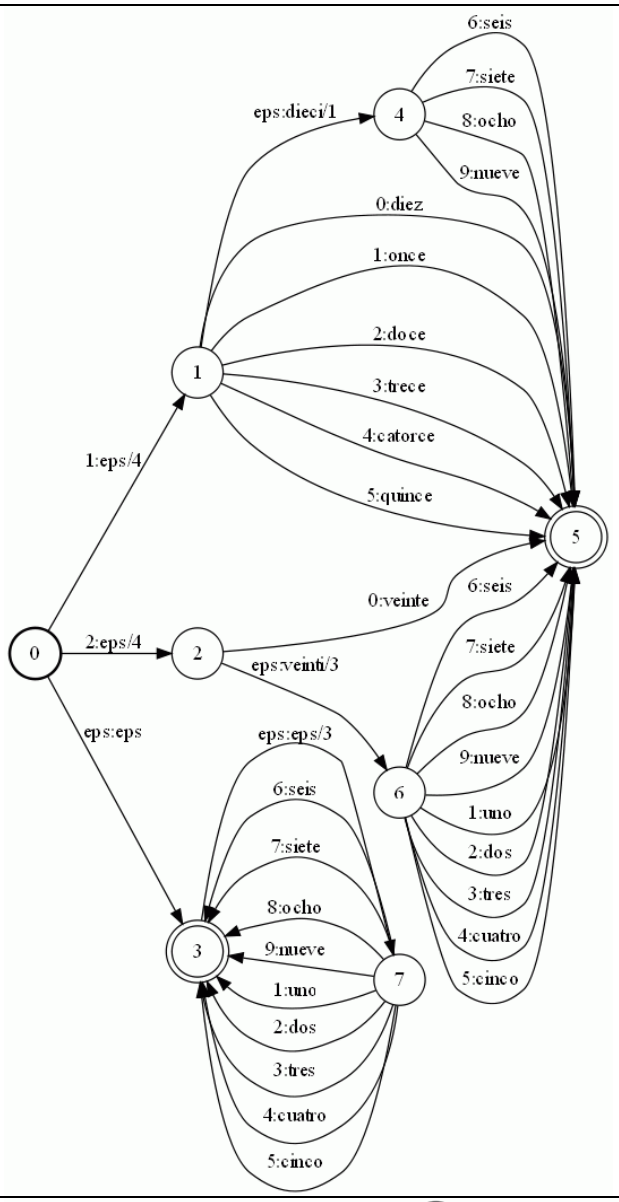
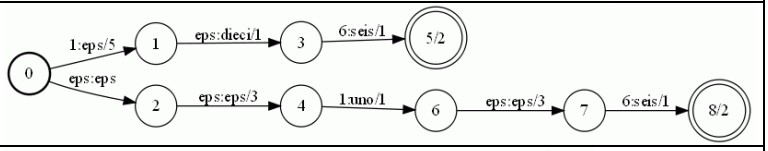
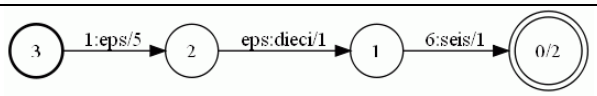
- **Uso:**
  - Compose(A, B, &C);
  - ComposeFst<Arc>(A, B);
  - fstcompose a.fst b.fst out.fst
- **Aplicación:** Se compone el WFST que contiene la cadena de entrada a traducir junto con el WFST de las traducciones para obtener las traducciones posibles para el texto de entrada.
- **Peso:** Si *A* traduce la cadena *x* a *y* con peso *a* y *B* traduce la cadena *y* a z con peso *b*, entonces su concatenación traduce la cadena *x* a z con peso *a* + *b*.
- **Ejemplo:**

`test.fst (A)`	
`nDecenas.fst (B)`	
`fstcompose test.fst nDecenas.fst result.fst (A o B)`	

2.3.8 ShortestPath

- **Descripción:** N-Caminos mas cortos (N-shortest paths)
- **Uso:**
  - ShortestPath(A, &B, nshortest=1);
  - fstshortestpath [–nshortest=$n] in.fst out.fst
- **Aplicación:** Se utiliza para obtener el camino más corto para obtener la solución más apropiada cuando existen varias traducciones posibles. El camino más corto es el de menor peso, siendo el peso del camino la suma de los pesos de las transiciones y estados del mismo.
- **Peso:** Los pesos corresponden al camino (o los caminos, en caso de N > 1) con menor peso del WFST original.
- **Ejemplo:**

test.fst	

`nDecenasU.fst`	
`fstcompose test.fst nDecenasU.fst result.fst`	
`fstshortestpath result.fst outShort.fst`	

3 Capítulo III - Desarrollo

Para la implementación de la solución al *preprocesamiento del texto de entrada* (o *normalización del texto*) descripto en los capítulos anteriores, se desarrolló un sistema **Traductor de Texto Escrito a Texto Oral** para el idioma español, basado en la librería de OpenFst (detallada en el Capítulo II).

En dicha implementación se realiza la conversión del texto escrito a otro texto equivalente, donde figuran las palabras pronunciables cómo serían leídas por el humano en español, utilizando *traductores de estados finitos con peso* (WFST = *Weighted Finite-State Transducers*), descriptos en el capítulo II.

En la Figura 3.1, se describe el contexto del sistema que realiza la traducción mencionada, donde se muestra el ingreso del texto a traducir (texto escrito) y como quedaría traducido (texto oral, es decir cómo se leería en voz alta) luego de utilizar el sistema **Traductor de Texto Escrito a Texto Oral**.

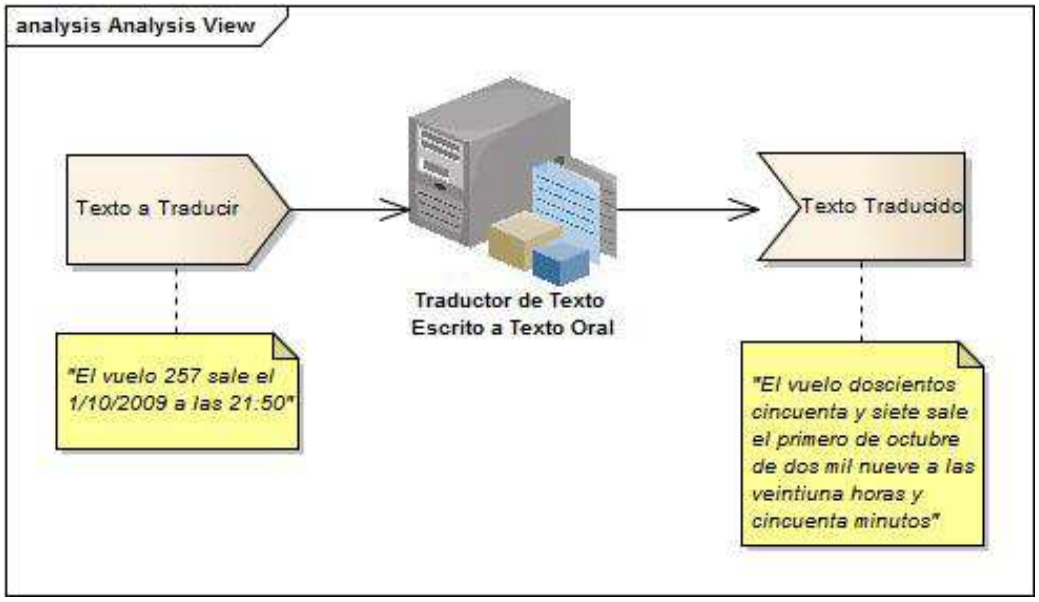


Figura 3.1 – Contexto del sistema Traductor de Texto Escrito a Texto Oral

Para desarrollar la solución se implementaron los siguientes módulos:

1) Generar WFST completo
2) Traducir texto

Inicialmente se genera el *traductor de estados finitos con peso* que incluye todas las traducciones posibles a realizar por el sistema (*WFST completo*) y luego se realiza la *traducción del texto* de entrada propiamente dicha, utilizando el traductor generado previamente.

A continuación se describen los detalles de implementación de cada uno de los requerimientos.

3.1 Generar WFST completo

El *WFST completo* se encuentra compuesto por varios WFST específicos que realizan las traducciones particulares.

En la Figura 3.2 se muestran los WFST específicos que se encuentran dentro del alcance de la implementación del sistema, donde se incluyen la generación de los WFST para realizar la traducción de **letras** (literales: *ABC = ABC*), **iniciales** (*ABC = a be ce*), **abreviaturas** (*Av. = avenida*), números **naturales** (*2 = dos*), **reales** (*2,5 = dos coma cinco*), **ordinales** (*2° = segundo / segunda*), **romanos** (*II = dos*), **teléfonos** (*4576-3390 = cuatro cinco siete seis tres tres nueve cero*), **fechas** (*2/2/2002 = dos de febrero de dos mil dos*), **horas** (*2:20 = dos horas y veinte minutos*), **fracciones** (*1/2 = un medio*) y **grados** (*2° = dos grados*). Existen casos donde más de una traducción es posible según el contexto de la palabra, por lo cual se incluyen los WFST con las distintas alternativas, como por ejemplo, abreviaturas por género (*Lic. = Licenciada o Licenciado*), números ordinales por género (*3° = tercer, tercero o tercera*), etc. para que en dichos casos cuando se encuentre indicado, mediante tags asignados manualmente (Ver Tags en el presente capítulo), se seleccione la solución correcta. En las siguientes secciones se explican en detalle cada uno de ellos.

Asimismo, existen casos que se podrían incluir para extender las traducciones dentro del idioma español, como por ejemplo las **URL** (*https://www.dc.uba.ar/= hache te te pe ese dos puntos barra barra doble ve doble ve doble ve punto de ce punto uba punto ar*) o **símbolos** según el contexto (por ejemplo el símbolo – que se lee como *menos, guión, a (para períodos o resultados deportivos), etc.*). Los mismos no son cubiertos en el presente trabajo, ya que se entiende que los casos existentes son los más comúnmente encontrados y serían suficientes para evaluar la solución propuesta. De todas formas no sería complejo como trabajo futuro agregarlos (debido a la modularización de los WFST específicos) para ampliar la cobertura de las traducciones.

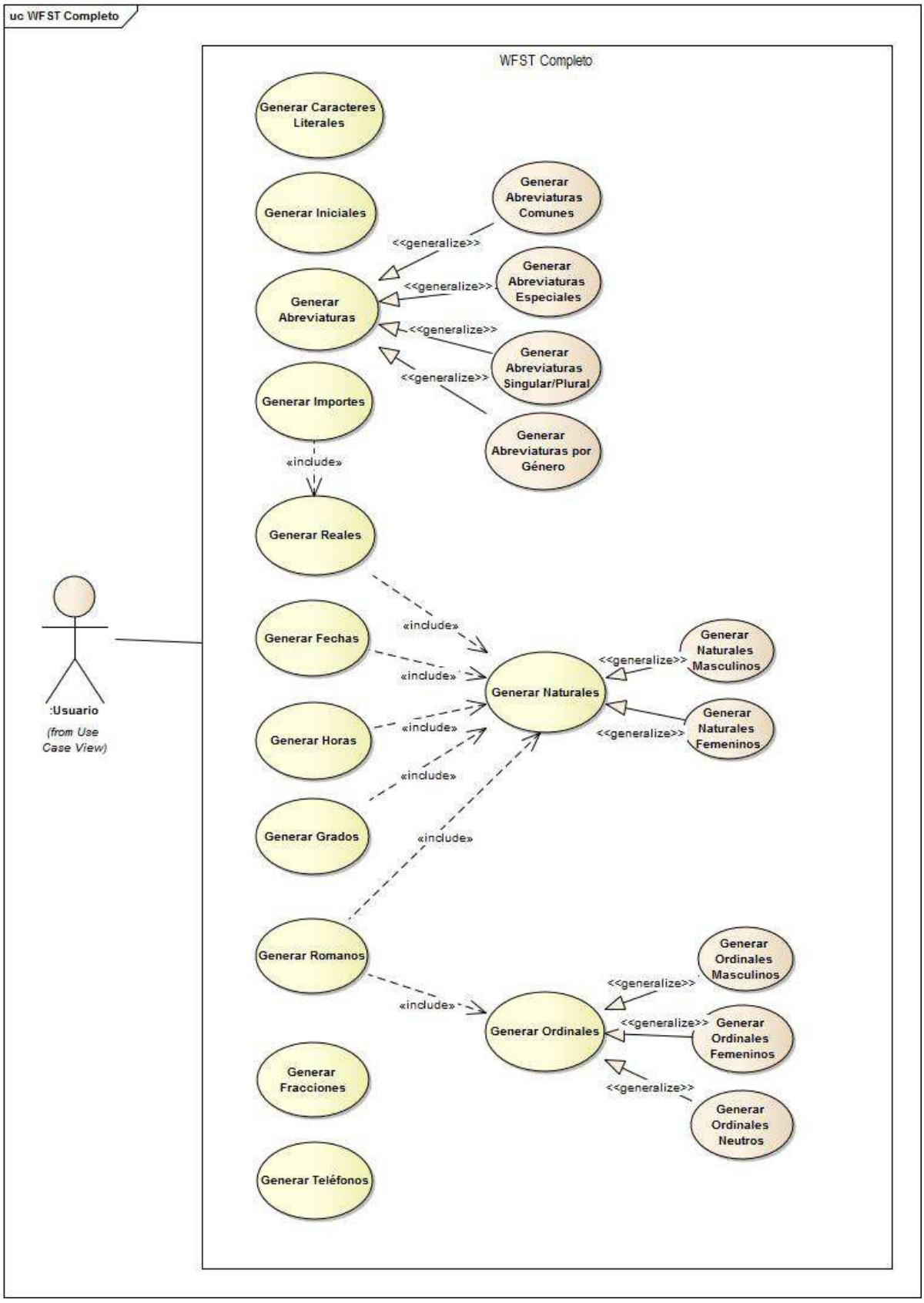

uc WFST Completo
WFST Completo
Generar Caracteres Literales
Generar Iniciales
Generar Abreviaturas
Generar Abreviaturas Comunes
Generar Abreviaturas Especiales
Generar Abreviaturas Singular/Plural
Generar Abreviaturas por Género
<<generalize>>
Generar Importes
«include»
Generar Reales
Generar Fechas
Generar Horas
Generar Grados
Generar Naturales
Generar Naturales Masculinos
Generar Naturales Femeninos
Generar Romanos
Generar Ordinales
Generar Ordinales Masculinos
Generar Ordinales Femeninos
Generar Ordinales Neutros
Generar Fracciones
Generar Teléfonos
:Usuario
(from Use Case View)

Figura 3.2 - Casos de Uso del WFST Completo

Luego de definir cada uno de los WFST específicos, se realizó **la unión** de todos ellos para generar el *WFST completo*, que es el utilizado para realizar las traducciones del texto de entrada. Por ejemplo, si los WFTS específicos A y B realizan las traducciones de naturales y de siglas respectivamente, la unión (A U B) realizará ambas traducciones. Luego, el WFST con la unión, aceptará secuencias de caracteres tanto de naturales como de siglas.

Para cada uno de los WFST específicos, como también para el *WFST completo*, se realizó **la clausura** de cada uno de ellos para permitir la iteración en las búsquedas de las traducciones. Por ejemplo, si se tiene una secuencia de palabras donde la primera es una sigla y la segunda es un número natural (Ej.: *DNI 44123456*), la primera palabra quedaría iterando en el WFST de siglas debido a la clausura del WFST específico y luego la segunda palabra se consume en el WFST de naturales por la clausura de dicho WFST. La clausura del WFST completo permite realizar la traducción de la segunda palabra luego de haber concluido la primera.

Para cada uno de los WFST, en cada **transición** figura un solo carácter como entrada o como salida para ser traducido. En el diagrama de la Figura 3.3 se muestra un ejemplo de como sería la secuencia para traducir el número *2 = dos,* incluida en los números naturales. Luego cuando el WFST es utilizado para la conversión del texto, la misma es realizada carácter por carácter.

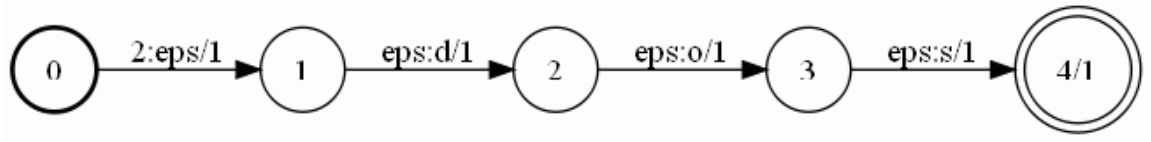


Figura 3.3 - WFST para la traducción del número 2

La palabra *eps* significa *Epsilon (ε)*, que en la teoría de autómatas representa al valor vacío, lo que indicaría que no existe carácter a ser traducido o traducción (según corresponda) de un carácter para dicha transición.

Los WFST específicos son **determinísticos**, con las **transiciones *epsilon* eliminadas** y se encuentran **minimizados**. Para construir los distintos WFST se utilizan operaciones de unión y concatenación para reutilizar WFST más pequeños. Durante dichas operaciones se generan nuevas transiciones y/o estados en los WFST resultantes. Es por ello que se realizan tareas de optimización para favorecer el recorrido de los mismos durante la traducción. Para optimizar los WFSTs, en principio se remueven las transiciones epsilon, luego se realiza la determinación del mismo y a continuación se minimiza para lograr un WFSTs, que acepta el mismo lenguaje, con la menor cantidad de estados / transiciones para que sea más fácil recorrerlo y en menor tiempo. Asimismo, se debe tener en cuenta que dichas operaciones de optimización tienen un costo durante la generación del WFST completo, pero siendo que el mismo se genera una sola vez y que las traducciones que hacen uso del mismo son numerosas, es preferible asumir ese costo en la generación y no en la traducción [FST 03].

Con respecto a **los pesos de los traductores**, estos son asignados de acuerdo a los criterios para beneficiar ciertas traducciones con respecto a otras, ya que se seleccionarán las traducciones que tengan menor peso en el camino recorrido para obtenerlas. Los mismos son parametrizables a fines de posibilitar al usuario seleccionar las preferencias de búsquedas.

Los pesos que se encuentran implementados son los siguientes:

- **Peso Interno de Caracteres Literales / Iniciales** (*char_p*): es el peso asignado para las transiciones de los WFST de caracteres literales (ej. letras como se encuentran ingresadas: *XX* = *XX*) e iniciales (ej.: *XX* = *equis equis*), debido a que estos serían el resultado trivial de la búsqueda de la traducción. Entonces, la posibilidad de diferenciarlos permite beneficiar a la solución específica por sobre la trivial (*XX* = *veinte* en número romano vs. *XX* para carácter literal o *equis equis* para iniciales).
- **Peso Final de Caracteres Literales / Iniciales** (*char_pf*): es el peso asignado para el estado final de los WFST de caracteres literales e iniciales. El peso final es utilizado especialmente en operaciones de clausura (Ver capítulo II), ya que es el asignado para la transición que sale desde este estado final al inicio del WFST.
- **Peso Interno** (*p*): es el peso asignado para las transiciones de cada uno de los WFST (con excepción de los caracteres literales e iniciales).
- **Peso Final** (*pf*): es el peso asignado para el estado final de cada uno de los WFST con excepción de los caracteres literales e iniciales.
- **Peso Interno de Tags** (*tag_p*): es el peso asignado para las transiciones de los WFST de tags, utilizados para seleccionar una traducción específica. En principio, están asignado como peso nulo para que se prefiera la traducción con tags a la traducción general, ya que con los mismos se permite indicar cual sería el WFST indicado para realizarla.
- **Peso Final de Tags** (*tag_pf*): es el peso asignado para el estado final de los WFST de tags.

En la Figura 3.4 se muestra un modelo reducido de los conceptos explicados anteriormente. Sólo se incluyen algunos WFST específicos parcialmente con fines de ejemplificar el modelo implementado (con las operaciones de unión, clausura, pesos, etc.), debido a que incluir a todos ellos y en forma completa haría que el diagrama no sea legible.

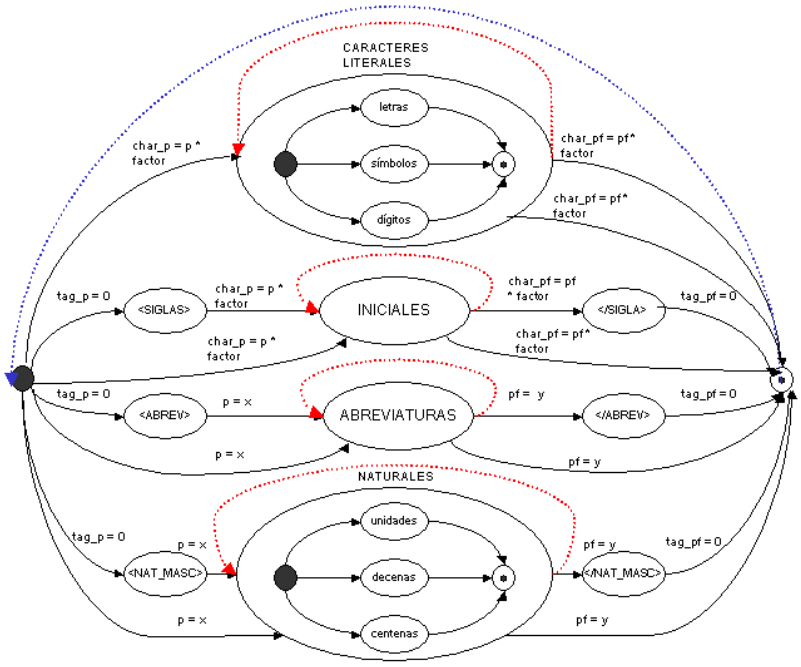


Figura 3.4 - WFST Modelo Reducido

Entonces, se incluye una versión reducida de los WFST de **naturales**, **literales** (caracteres que se dicen tal como se leen), **iniciales** (para siglas) y **abreviaturas** para ejemplificar el comportamiento del traductor.

Con respecto a los pesos modelados en el diagrama de la Figura 3.4, se le asigna a las variables *p* y *pf* los valores de *x* e *y* respectivamente (*p* = *x* y *pf* = *y*) debido a que los mismos son parametrizables y para mostrar que los mismos pueden cambiar según sea conveniente. La asignación de los valores *x* e *y* correspondiente se verá en el siguiente capítulo durante la evaluación, donde se justificará su elección.

Luego los valores de los pesos para los WFST de caracteres literales e iniciales (*char_p* y *char_pf*) se calculan en base a los valores asignados previamente a los pesos del resto de los WFST específicos (*p* y *pf* respectivamente) y un *factor* que también es un valor positivo, variable y parametrizable. El *factor* es un número permite distanciar los valores de ambos pesos para priorizar las soluciones de los WFST específicos, ya que sus pesos (*p* y *pf*) son menores que los de la solución trivial (*char_p* y *char_pf*) debido a la multiplicación de *p* y *pf* por el *factor*. La asignación adecuada del valor del *factor* correspondiente también se verá en el siguiente capítulo y será seleccionada en conjunto con los valores de los pesos.

Por otro lado, el peso de los WFST que generan las traducciones con tags (*tag_p*), son asignados con el valor nulo, debido a que se busca favorecer el uso de tags frente a otras soluciones cuando los mismos se encuentran presentes para desambiguar las traducciones posibles, como se explica mas adelante en el presente capítulo (Ver la sección 3.3 Tags). Esto es que cuando el traductor encuentre el tag correspondiente en el texto de entrada, se prefiera la solución con tag debido a que tiene menor peso que la solución sin tag y las soluciones de menor peso son las que se priorizan.

Las clausuras que permiten seguir iterando en los WFST como se mencionó anteriormente, se indican con líneas punteadas, *roja* para los WFST específicos y *azul* para el WFST completo.

A continuación se detalla cada una de las traducciones mencionadas.

3.1.1 Caracteres Literales

- **Texto de Entrada**: Secuencia de letras, símbolos y números que se leen literalmente (ej.: *¿Pabellón 1?*).
    - **Rango**: El rango contemplado incluye letras en mayúsculas y minúsculas del alfabeto español, dígitos del 0 al 9 y símbolos utilizados en el idioma español.
    - **Formato**: Letras, dígitos y símbolos tal como se encuentran ingresados en el texto (¡ABC! ¿123?).
- **Texto de Salida**: Letras (palabras), símbolos expresados literalmente y números expresados como dígitos (ej.: *¿Pabellón uno?*)

Los símbolos en general son representados como se encuentran ingresados (ej.: símbolos de puntuación *¿?* .), pero existen casos particulares donde se representa cómo se leen según su significado (ej.: @ = *arroba*). (Ver Apéndice A – Detalles de Implementación – Caracteres del Sistema y Símbolos del Sistema)

Ejemplos:

Texto de entrada	Texto de salida
¡Hola mundo!	¡Hola mundo!
UBA	UBA
@mundo	arroba mundo

3.1.2 Iniciales

- **Texto de Entrada**: Secuencia de letras que representan una sigla (ej.: *DC*). [RAE Sig]
  - **Rango**: El rango contemplado incluye letras en mayúsculas y minúsculas del alfabeto español y dígitos del 0 al 9.
  - **Formato**: Letras y dígitos tal como se encuentran ingresados en el texto.
- **Texto de Salida**: Iniciales y dígitos (ej.: *de ce*)

Las iniciales son utilizadas para deletrear las siglas, por sus iniciales cuando estas no son acrónimos (una sigla que se pronuncia como una palabra [RAE Acr][WIK Acr]). En el caso de ser acrónimos se leen como palabras, es decir se leen literalmente, esto es carácter por carácter literal (Ej: OVNI, UBA, AFIP, etc.) y para generar Caracteres Literales se utiliza el WFST descripto anteriormente.

Ejemplos:

Texto de entrada	Texto de salida
AFJP	a efe jota pe
DNI	de ene i

Existen casos de siglas que no son deletreadas estrictamente por sus iniciales (ej.: *CD-ROM* = *ce de rom*), en ese caso son traducidas análogamente a las abreviaturas (Ver sección 3.1.3 Abreviaturas para mayor detalle). En el caso que la sigla especial se encuentre en el listado de abreviaturas especiales, se retorna antes que las opciones de iniciales o caracteres, ya que la asignación de pesos favorece a las abreviaturas especiales.
Para diferenciar los acrónimos de las siglas que se deletrean como iniciales se utilizan marcas especiales (tag = `<SIGLA>`) para estas últimas (Ver Tags en el presente capítulo), debido que la complejidad del problema para distinguir a priori cual sería la solución correcta entre ambas opciones queda fuera del alcance del trabajo de la tesis.

3.1.3 Abreviaturas

Las abreviaturas son conjuntos de letras que en general no se pueden pronunciar y que representan una palabra [JIM 99]. Para encontrar si un conjunto de letras corresponde a una abreviatura, se busca en un archivo (*abreviaturas.txt*) conformado inicialmente con la lista de abreviaturas de la Real Academia Española [RAE Abr]. Dicho archivo puede ser dinámicamente actualizado para contemplar nuevas abreviaturas que no se encuentren incluidas en el listado original (Ver Apéndice A – Ejecutables - buildFstTesis para la construcción de WFST completo que incluye el listado de abreviaturas). En el caso que el conjunto de letras se encuentre en la lista, se devuelve su expansión; en caso contrario, es tratada como una palabra común (Ver sección 3.1.1 Caracteres Literales). Si la palabra es encontrada en la lista de abreviaturas, la traducción es devuelta como su expansión

asociada, ya que el peso para las abreviaturas en menor que el de caracteres literales como se explicó anteriormente.

- **Texto de Entrada**: Conjunto de letras que representan una abreviatura. (ej.: *etc.*)
    - **Rango**: Se incluye la lista de abreviaturas desde un archivo que debe encontrarse disponible para realizar la generación del traductor correspondiente.
    - **Formato**: Cada línea del archivo es de la siguiente forma: *abreviatura*|*palabra* asociada (ej.: *etc.*|*etcétera*).
- **Texto de Salida:** Palabra asociada a la abreviatura (ej.: *etcétera*)

Para que una secuencia de letras sea considerada como una abreviatura, la misma debe estar precedida por un espacio en blanco, para indicar que se trata de una palabra aislada y evitar la expansión de abreviaturas que se encuentran incluidas dentro de otra palabra y que coinciden con el fin de la oración (Ej.: *a.= área*, en el caso de las palabras que terminan con a y finalizan la oración se realizaría una expansión que no sería correcta, como: *Argentina. = Argentinárea*).

Es importante destacar que los listados de las abreviaturas incluidas en los archivos, pueden actualizarse y en ese caso el traductor debe generarse nuevamente para contemplar dichas actualizaciones (Ver sección de Ejecutables en el Apéndice A).

Ejemplos:

Texto de entrada	Texto de salida
Av.	Avenida
Bs.As.	buenos aires
FF.CC.	ferrocarriles
S.A.	sociedad anónima

3.1.3.1 Abreviaturas Especiales

Existen casos de abreviaturas que tienen asociadas distintas palabras según el contexto en donde se usan. Este es el caso del plural (ej.: *kg. = kilo o kilos*) y el género (ej.: *Lic. = licenciado o licenciada*), por lo cual estas son tratadas especialmente. Para distinguir cuál sería la traducción correcta se utilizan tags para cada uno de los tipos de abreviaturas especiales que se indican a continuación (Ver Tags más adelante en el presente capítulo).

- **Texto de Entrada**: Conjunto de letras que representan una abreviatura. (ej.: *pág.*)
    - **Rango**: Se incluye la lista de abreviaturas desde el archivo correspondiente (*abrevEspeciales.txt*, *abrevGenero.txt, abrevPlural.txt*).
    - **Formato**: Cada línea del archivo es de la siguiente forma: *abreviatura*|*palabra asociada* (ej.: *pág.*|*página*).

- **Texto de Salida:** Palabra asociada a la abreviatura según el contexto donde se indica género (femenino / masculino) y plural/singular según corresponda (ej.: *página o páginas*)

Los traductores que se generan para contemplar las abreviaturas especiales y sus correspondientes expansiones son los siguientes:

- Especiales: Se realiza la expansión para la abreviatura correspondiente similar a la realizada para las abreviaturas comunes, pero se realiza desde una lista distinta a fines de diferenciarlas (m3 = *metros cúbicos*). La lista se encuentra en el archivo *abrevEspeciales.txt*.
- Plural / Singular: Se realiza la expansión para la abreviatura correspondiente y en el caso de ser plural se agrega la **s** (pag. = *página o* pag. = *páginas*). La lista se encuentra en el archivo *abrevPlural.txt*
- Por Género: Se realiza la expansión para la abreviatura correspondiente, en el caso de ser femenino se agrega la **a** y en caso masculino la **o** (Lic. = *Licenciada o* Lic.. = *Licenciado*). La lista se encuentra en el archivo *abrevGenero.txt*.

Ejemplos:

Texto de entrada	Texto de salida
km.	kilómetro o kilómetros
Pág.	página o páginas
n/	nuestro o nuestra
Km2	Kilómetros cuadrado
RH+	erre hache positivo

3.1.4 Naturales

- **Texto de Entrada**: Conjunto de dígitos que representan números naturales (ej.: *4*).
  - **Rango**: El rango contemplado es de 0 a 999999999999
  - **Formato**: Dígitos de 0 a 9 consecutivos sin separador de miles y/o decimales.
- **Texto de Salida**: Número expresado en letras (ej.: *cuatro*)

El separador de miles no es reconocido en el texto de entrada y el separador de decimales no se considera, ya que se tratan de números naturales.

Los números con separador de miles quedan fuera del alcance de la traducción, debido a que existen diversos símbolos de puntuación numérica (punto o coma según se encuentre configurado), y asumiendo que estos podrían ser fácilmente normalizados a los formatos especificados. Las expansiones contempladas por el traductor implementado han sido seleccionadas con este criterio, quedando la implementación de otras opciones para trabajo futuro.

El símbolo 0 a la izquierda no es tomado como dígito significativo, por lo que no es traducido como natural si es seguido de otros dígitos ya que no se considera como tal sino como una secuencia de dígitos (Ej. *012* se lee como *cero uno dos* en lugar de *doce*).

Los traductores que se generan para contemplar los distintos casos de los números naturales son los siguientes:

- Masculino: Los naturales generados que incluyen al 1 expandido como **uno** (Ej.: *21 = veintiuno*).
- Femenino: Los naturales generados que incluyen al 1 expandido como **una** (Ej.: *21 = veintiuna*).

Los mismos serán convenientemente seleccionados mediante el uso de los tags correspondientes (Ver la sección 3.3 Tags) de acuerdo al contexto cuando corresponda indicar la traducción correcta de acuerdo al contexto (ej: *21 Olimpíada = veintiuna Olimpíada*). Es importante destacar que si bien existen casos donde existe diferencia de género (ej.: uno / una) y otros que no (ej.: dos), se genera un conjunto de traducciones para los naturales masculinos donde se incluyen ambos casos (ej.: uno, dos, etc.) y otro análogo para los naturales femeninos (ej.: una, dos, etc.). Luego los tags deberían utilizarse sólo en los casos donde es necesario distinguir el género (ej.: *1 = uno / 1 = una*), siendo que en los otros casos es indistinto el uso del tag debido a que se realiza la misma traducción en ambos conjuntos (ej.: *2 = dos*).

Ejemplos:

Texto de entrada	Texto de salida
0	Cero
16	dieciséis
123	ciento veintitrés
31000	treinta y un mil
999999999	novecientos noventa y nueve millones novecientos noventa y nueve mil novecientos noventa y nueve.

3.1.5 Reales

- **Texto de Entrada**: Conjunto de dígitos que representan números reales (ej.: *4,5*).
  - **Rango**: El rango contemplado es de –9999999999,99 a 9999999999,99
  - **Formato**: [Opcional: Símbolo positivo (+) /negativo (-)] + Números del 0 al 9 + separador de decimales (, o .) + números del 0 al 9 (Ej.: nnnn,nn o +nnnn.nn o -nnnn.nn).
- **Texto de Salida**: Número expresado en letras (ej.: *cuatro coma cinco*)

El separador de miles no es reconocido en el texto de entrada y el separador de decimales puede ser coma (,) o bien punto (.). Al igual que con los naturales, los números con separador de miles quedan fuera del alcance de la traducción, asumiendo que estos podrían ser normalizados previamente a los formatos especificados.

El símbolo para indicar si el número es positivo (+) o negativo (-) es opcional, es decir, es posible especificar los números reales con o sin símbolo.

El símbolo 0 a la izquierda no es tomado como dígito significativo, por lo que no es traducido como natural si es seguido de otros dígitos ya que no se considera como tal sino como una secuencia de dígitos, excepto que preceda al símbolo de separador de decimales.

Los decimales se expanden dígito por dígito, excepto a en los casos que los decimales correspondan a una decena que se expande como tal (Ej: *2,50 = dos coma cincuenta*) ya que se prioriza esta solución.

Ejemplos:

Texto de entrada	Texto de salida
1.012	uno coma cero uno dos
+2,012	más dos coma cero uno dos
0,25	cero coma veinticinco / cero coma dos cinco
-5,99	menos cinco coma nueve nueve

3.1.6 Ordinales

[RAE Ord] [VIT Ord] [WIK Ord]

- **Texto de Entrada**: Conjunto de dígitos que representan números naturales seguidos del símbolo de grado (ej.: *4°*).
    - **Rango**: El rango contemplado es de 1° a 9999999°
    - **Formato**: Dígitos de 0 a 9 consecutivos sin separador de miles y/o decimales + símbolo de grado (°).
- **Texto de Salida**: Número ordinal expresado en letras (ej.: *cuarto*)

Dado que los ordinales son números naturales seguidos del símbolo de grado, se aplican las mismas consideraciones con respecto al separador de miles y de decimales. Asimismo, el cero no se incluye en el rango debido a que no se considera como ordinal.

Se permite identificar el género del ordinal ya que dependiendo del contexto podría ser masculino (Ej.: *cuarto*) o femenino (Ej.: *cuarta*). También se considera el caso de que primero (*1°*) y tercero (*3°*) presentan apócope delante de un nombre masculino singular (*primer* / *tercer*).

Las distintas ocurrencias del símbolo (4ª) y abreviaturas (4to. / 4ta.) que acompañan al número quedan fuera del alcance de la traducción, asumiendo que estos podrían ser normalizados previamente al símbolo de grado especificado (°).

Los traductores que se generan para contemplar los distintos casos de los números ordinales son los siguientes.

- Masculino: Los ordinales generados se expanden finalizando como género masculino (Ej.: *1° = primero, 4° = cuarto*).
- Femenino: Los ordinales generados se expanden finalizando como género femenino (Ej.: *1° = primera, 4° = cuarta*).
- Neutro: Los ordinales generados incluyen al 1 y 3 expandido como *primer* y *tercer* respectivamente y el resto de los números se expanden como género masculino (Ej.: *1° = primer, 4° = cuarto*).

Para distinguir cuál sería la traducción correcta se utilizan tags para cada uno de los tipos de ordinales mencionados (Ver Tags más adelante en el presente capítulo).

Ejemplos:

Texto de entrada	Texto de salida
1°	primero / primera / primer
16°	décimo sexto / sexta
123°	centésimo vigésimo tercero / tercera / tercer

3.1.7 Romanos

- **Texto de Entrada**: Combinaciones de letras que representan números romanos (ej.: *IV*).
  - **Rango**: El rango contemplado es de I (1) a MMMCMXCIX (3999) en mayúsculas.
  - **Formato**: Combinaciones de las siguientes letras: I, V, X, L, C, D y M, escritas en mayúsculas (Ej.: *IV*).
- **Texto de Salida:** Número natural u ordinal (ej.: *cuatro* / *cuarto/a*)

Para que un conjunto de letras sea considerado como un número romano, el mismo debe estar precedido y seguido por un espacio en blanco, para indicar que se trata de una palabra aislada y evitar la expansión de romanos que se encuentran incluidas dentro de otra (Ej.: *MALVINAS= Macincuenta y seisNAS*).

El rango llega al valor de MMMCMXCIX (3999), debido a que los mayores a este número se deben representar con una línea por sobre las letras, quedando esto fuera del alcance del traductor ya que serían caracteres gráficos. Por otra parte, existen pocos casos de números romanos representados mayores a este valor.

Los traductores que se generan para contemplar los distintos casos de los números romanos son los siguientes:

- Naturales Masculinos: Los romanos naturales generados que incluyen al 1 expandido como **uno** (Ej.: *XXI = veintiuno*).
- Romanos Ordinales Masculinos: Los romanos ordinales generados se expanden finalizando como género masculino (Ej.: *I = primero, IV = cuarto*).
- Romanos Ordinales Femeninos: Los romanos ordinales generados se expanden finalizando como género femenino (Ej.: *I = primera, IV = cuarta*).
- Romanos Ordinales Neutros: Los romanos ordinales generados incluyen al 1 y 3 expandido como **primer** y **tercer** respectivamente y el resto de los números se expanden como género masculino (Ej.: *I = primer, IV = cuarto*).

Para distinguir cuál sería la traducción correcta se utilizan tags para cada uno de los tipos de números romanos mencionados (Ver Tags más adelante en el presente capítulo).

Ejemplos:

Texto de entrada	Texto de salida
I	uno / Primero / primera / primer

Texto de entrada	Texto de salida
XVI	dieciséis / Décimo sexto / sexta
CXXIII	ciento veintitrés / Centésimo vigésimo tercero / tercera / tercer

3.1.8 Fechas

- **Texto de Entrada**: Conjunto de números que representan fechas en uno de los formatos definidos (ej.: *16/12/1993*).
    - **Rango**: El rango contemplado es del 01/01/01 al 31/12/9999
    - **Formato**: La fecha reconocida se encuentra dentro de los siguientes formatos:
        - Número del 1 al 31 + (- o /o .) + número del 1 al 12 + (- o /o .) + grupo de 1 a 4 dígitos del 0 al 9 (Ej.: dd/mm/yyyy o dd.mm.yyyy o dd-mm-yyyy).
        - Número del 1 al 31 + (- o /o .) + número del 1 al 12 + (- o /o .) + grupo de 2 dígitos del 0 al 9 (Ej.: .d/mm/yy o dd.mm.yy o dd-mm-yy)
- **Texto de Salida**: Fecha expresada en letras (ej.: *dieciséis de diciembre de mil novecientos noventa y tres*).
    - El mes en número es expandido a su correspondiente en letras (ej.: *12* como *diciembre*).
    - Si el año es representado con 2 dígitos (ej.: *16/12/93*), el separador previo es traducido como **del** (Ej.: *dieciséis de diciembre del noventa y tres*)
    - Si el año es representado con 1, 3 o 4 dígitos (ej.: *16/12/1993*), el separador previo es traducido como **de** (Ej.: *dieciséis de diciembre de mil novecientos noventa y tres*)

Los separadores de fecha se permiten combinar ya que no es el propósito de la traducción estandarizar el formato de la misma y se asume que normalmente no van a ser combinados (ej.: *16/12-93* sería una fecha valida). Los meses representados en letras (ej.: *16/diciembre/1993* o *16/dic/1993* y en romanos (ej.: *16/XII/93*), o bien las fechas sin año (ej.: *16/12*), quedan fuera del alcance de la traducción, asumiendo que estos podrían ser normalizados previamente a los formatos especificados.

Otro aspecto que queda fuera del alcance de la tesis es el chequeo de fechas válidas (ej.: *30/02/2012* puede ser traducida como *treinta de febrero de dos mil doce*) dado que se asume que las fechas encontradas dentro de los textos a traducir normalmente son correctas, o bien, estos podrían ser validadas previamente.

El símbolo 0 a la izquierda es reconocido pero es ignorado a la hora de traducción (ej.: *06/02/01* es leído como *6/2/1* para su correspondiente traducción).

Ejemplos:

Texto de entrada	Texto de salida
16.12.1993	dieciséis de diciembre de mil novecientos noventa y tres
9/7/1816	nueve de julio de mil ochocientos dieciséis

Texto de entrada	Texto de salida
09-07-16	nueve de julio del dieciséis
1/1/1	uno de enero del uno

3.1.9 Horas

- **Texto de Entrada**: Conjunto de números que representan horas en uno de los formatos definidos (ej.: *16:30*).
    - **Rango**: El rango contemplado es de 0:00:00..23:59:59
    - **Formato**: La hora reconocida se encuentra dentro de los siguientes formatos:
        - Número del 0 al 23 + (: o .) + número del 0 al 59 (ej.: hh:mm o hh.mm).
        - Número del 0 al 23 + (: o .) + número del 0 al 59 + (: o .) + número del 0 al 59 (ej.: hh:mm:ss o hh.mm.ss)
- **Texto de Salida**: hora expresada en letras (ej.: *dieciséis horas y treinta minutos*).

Los separadores de horas (: y .) se permiten combinar ya que no es el propósito de la traducción estandarizar el formato de la misma y se asume que normalmente no van a ser combinados (ej.: *16:12.50* sería una hora valida). Además, existen numerosas maneras de leer la hora, siendo muchas de estas correctas. Las expansiones contempladas por el traductor implementado han sido seleccionadas entre estas opciones correctas, quedando la implementación de otras opciones para trabajo futuro.

El símbolo 0 a la izquierda si es seguido de otro dígito, en las horas es reconocido pero es ignorado a la hora de traducción (ej.: *06:02* es leído como *6:02* para su correspondiente traducción). En el caso de la hora *00:00* es traducido como *cero* horas.

Ejemplos:

Texto de entrada	Texto de salida
4:10	cuatro horas y diez minutos
04.10	cuatro horas y diez minutos
01:00	una horas
23:59:59	veintitrés horas cincuenta y nueve minutos cincuenta y nueve segundos

3.1.10 Importes

- **Texto de Entrada**: Conjunto de números reales o naturales precedidos por el símbolo de importe (ej.: *$2,50*).
    - **Rango**: El rango contemplado es de $0 a $999999999999,99
    - **Formato**: El importe reconocido se encuentra dentro de los siguientes formatos:
        - Símbolo de importe + [Opcional: Espacio] + Número natural (ej.: $nn o $ nn).

        - Símbolo de importe + [Opcional: Espacio] + Número real, con distintas cantidad de decimales (ej.: $nn,nn o $nn.nn o $ nn,nn o $ nn.nn).

- **Texto de Salida**: importe expresado en letras (ej.: *dos coma cincuenta pesos*).

Los símbolos de importe soportados para considerar una secuencia de caracteres como un importe son los siguientes: $, u$d, us, us, UD, US, US$ ya que son los mas usualmente utilizados. Quedan fuera del alcance otros símbolos de distintas monedas (ej.: AR$) quedando éstos para ser incluidos en trabajos futuros.

Los espacios entre el símbolo y el número se consideran (en el caso que se encuentren incluidos) para realizar la traducción.

Ejemplos:

Texto de entrada	Texto de salida
$ 1,99	uno coma noventa y nueve pesos
$2.50	dos punto cincuenta pesos
$ 1	un peso

3.1.11 Teléfonos

- **Texto de Entrada**: Conjunto de números que representan un teléfono de Capital Federal / Gran Buenos Aires (ej.: *4576-3390*).
    - **Rango**: El rango contemplado es de 0000-0000 a 9999-9999
    - **Formato**: El teléfono reconocido se encuentra dentro del siguiente formato:
        - Número del 0000 al 9999 + (-) + número del 0000 al 9999 (Ej.: dddd-dddd).
- **Texto de Salida**: Teléfono expresado en números leídos como dígitos (ej.: *cuatro cinco*...).

Los formatos de telefonía de larga distancia nacional e internacional y de telefonía móvil no se encuentran soportados en el presente trabajo, quedando su ampliación para aplicaciones futuras de este trabajo.

Ejemplos:

Texto de entrada	Texto de salida
4510-1100	Cuatro cinco uno cero uno uno cero cero
4576-3300	Cuatro cinco siete seis tres tres cero cero

3.1.12 Fracciones

- **Texto de Entrada**: Conjunto de expresiones numéricas que representan fracciones más comunes (ej.: *1/4*).

- **Rango**: Se incluye la siguiente lista de fracciones en la inicialización: 1/2, 1/4, 3/4, 1/3 y 2/3
- **Formato**: Número natural + (/) + número natural (Ej.: n/n).

- **Texto de Salida**: Fracción expresada en letras (ej.: *un cuarto*).

Otras traducciones de fracciones (ej.: 1/6, 1/7, etc.) quedan fuera del alcance de la presente tesis, dado que las traducciones son de dominio general, y traducciones de propósito específico (ej.: dominio matemático) son propuestas como trabajo futuro.

Ejemplos:

Texto de entrada	Texto de salida
1/2	Un medio
1/4	Un cuarto
3/4	Tres cuartos
1/3	Un tercio

3.1.13 Grados

- **Texto de Entrada** Conjunto de dígitos que representan números naturales seguido del símbolo de grado (ej.: *4°*).
  - **Rango**: El rango contemplado es el definido para los naturales y reales ya que los mismos son utilizados en la conversión.
  - **Formato**: Los grados reconocidos se encuentran dentro de los siguientes formatos:
    - Número natural + Símbolo ° (ej.: 16°).
    - Número real + Símbolo ° (ej.: -2.5°).
- **Texto de Salida**: Grados expresado en letras (ej.: cuatro grados).

Ejemplos:

Texto de entrada	Texto de salida
1°	Un grado
16°	Dieciséis grados
-2.5°	Menos dos punto cinco grados

3.2 Traducir texto

La implementación de las traducciones del *texto escrito* al *texto oral* (como se leería en voz alta), se realiza de la siguiente manera: a partir de un texto de entrada (obtenido de distintas fuentes, como noticias de diarios, artículos de la Web, Wikipedia, etc.), se utiliza el *WFST completo* generado previamente para obtener las traducciones posibles y el resultado de la conversión se devuelve en un texto de salida.

El **texto de entrada** se toma de un archivo de texto, que contiene las frases a traducir en cada línea del mismo, en formato ASCII extendido (codificación Windows-1252 (ANSI)) [WIK ASC].

Las palabras pueden contener o no tags para indicar la traducción deseada (Ver los tags disponibles en la sección Tags). A continuación se muestran ejemplos de archivos de entrada con y sin tags:

Texto de entrada sin Tags:

```
[WIKIPEDIA] El Club Atlético San Lorenzo de Almagro es una institución deportiva
y social centenaria, con sede en la ciudad de Buenos Aires, Argentina, cuya
principal actividad es el fútbol profesional. Es uno de los cinco grandes del
fútbol argentino.

[WIKIPEDIA] Hacia diciembre de 2010, San Lorenzo de Almagro es clasificado por la
IFFHS como el octavo mejor equipo sudamericano de la primer década del siglo XXI.

[WIKIPEDIA] Luego de varios meses, se realizó la primera asamblea extraordinaria,
el 1 de abril de 1908, fecha establecida como el día de su fundación. En ese
momento se decidió elegir el nombre del flamante club.

[WIKIPEDIA] Durante este período se crea el barrio de Boedo mediante la Ordenanza
N° 23698 del 11 de junio de 1968.

[WIKIPEDIA] Posee 162 peñas, de las cuales 124 se encuentran distribuidas a lo
largo y a lo ancho de toda la Argentina, y 28 en el exterior, ubicadas en:
Alemania, Andorra, Australia, Chile, 10 en España (Andalucía, Madrid, Barcelona,
Fuengirola, Oliva, Santander, Tarragona, Tenerife, Valencia y Zaragoza), 5 en
Estados Unidos (Arizona, Hawái, Miami, Nueva York y Texas), Grecia, Inglaterra,
Italia, Israel, Japón, México, Perú, Puerto Rico y Uruguay. En la actualidad
cuenta con 33000 socios aproximadamente.
```

Texto de entrada con Tags:

```
[WIKIPEDIA] El Club Atlético San Lorenzo de Almagro es una institución deportiva
y social centenaria, con sede en la ciudad de Buenos Aires, Argentina, cuya
principal actividad es el fútbol profesional. Es uno de los cinco grandes del
fútbol argentino.

[WIKIPEDIA] Hacia diciembre de 2010, San Lorenzo de Almagro es clasificado por la
IFFHS como el octavo mejor equipo sudamericano de la primer década del
siglo<ROM_NAT> XXI</ROM_NAT>.

[WIKIPEDIA] Luego de varios meses, se realizó la primera asamblea extraordinaria,
el <NAT_MASC>1</NAT_MASC> de abril de 1908, fecha establecida como el día de su
fundación. En ese momento se decidió elegir el nombre del flamante club.

[WIKIPEDIA] Durante este período se crea el barrio de Boedo mediante la
Ordenanza<ABREV_SING> N°</ABREV_SING> 23698 del 11 de junio de 1968.

[WIKIPEDIA] Posee 162 peñas, de las cuales 124 se encuentran distribuidas a lo
largo y a lo ancho de toda la Argentina, y 28 en el exterior, ubicadas en:
Alemania, Andorra, Australia, Chile, 10 en España (Andalucía, Madrid, Barcelona,
Fuengirola, Oliva, Santander, Tarragona, Tenerife, Valencia y Zaragoza), 5 en
Estados Unidos (Arizona, Hawái, Miami, Nueva York y Texas), Grecia, Inglaterra,
Italia, Israel, Japón, México, Perú, Puerto Rico y Uruguay. En la actualidad
cuenta con 33000 socios aproximadamente.
```

El **texto de salida** se guarda en archivos de texto, que contienen las frases traducidas en cada línea de los mismos. El formato de salida es UTF8. Los dos archivos de salida generados para evaluar los resultados de la traducción son los siguientes:

- **Mejor Resultado**: se retorna la mejor traducción encontrada para la frase de entrada. La mejor es considerada por tener menor peso con respecto a las restantes (si es que existen). Este archivo, permite comparar fácilmente las frases retornadas con las de entrada, para

verificar las expansiones realizadas, o bien, con el de referencias con las traducciones realizadas por humanos, para evaluar si las traducciones son correctas.

- **N-Mejores Resultados** (donde N es configurable): se retornan las N mejores traducciones encontradas para la frase de entrada, ordenadas de menor peso a mayor peso, incluyendo la de mejor encontrada devuelta en el archivo mencionado anteriormente (Mejor Resultado). Este archivo permite evaluar si la frase esperada se encuentra dentro de las N primeras soluciones. Por otra parte, se agrega el peso de cada una de las traducciones al finalizar cada frase, para permitir un mejor análisis de los resultados y pueden existir soluciones duplicadas debido a como se realizan los recorridos de los distintos caminos[1]. También pueden existir distintas traducciones con el mismo peso. Luego, en el archivo se salida figuran todas éstas posibles soluciones.

A continuación se muestran ejemplos de archivos de salida de Mejor Resultado y N-Mejores Resultados:

Texto de salida "Mejor Resultado":

```
[WIKIPEDIA] El Club Atlético San Lorenzo de Almagro es una institución deportiva
y social centenaria, con sede en la ciudad de Buenos Aires, Argentina, cuya
principal actividad es el fútbol profesional. Es uno de los cinco grandes del
fútbol argentino.

[WIKIPEDIA] Hacia diciembre de dos mil diez, San Lorenzo de Almagro es
clasificado por la IFFHS como el octavo mejor equipo sudamericano de la primer
década del siglo vigésimoprimer.

[WIKIPEDIA] Luego de varios meses, se realizó la primera asamblea extraordinaria,
el uno  de abril de mil novecientos ocho , fecha establecida como el día de su
fundación. En ese momento se decidió elegir el nombre del flamante club.

[WIKIPEDIA] Durante este período se crea el barrio de Boedo mediante la Ordenanza
números veintitres mil seiscientos noventa y ocho del once de junio de mil
novecientos sesenta y ocho.

[WIKIPEDIA] Posee ciento sesenta y dos peñas, de las cuales ciento veinticuatro
se encuentran distribuidas a lo largo y a lo ancho de toda la Argentina, y
veintiocho  en el exterior, ubicadas en: Alemania, Andorra, Australia, Chile,
diez  en España (Andalucía, Madrid, Barcelona, Fuengirola, Oliva, Santander,
Tarragona, Tenerife, Valencia y Zaragoza), cinco en Estados Unidos (Arizona,
Hawái, Miami, Nueva York y Texas), Grecia, Inglaterra, Italia, Israel, Japón,
México, Perú, Puerto Rico y Uruguay. En la actualidad cuenta con treinta y tres
mil socios aproximadamente.
```

Texto de salida "N-Mejores Resultados":

```
/* ====================================== */

ENTRADA: [WIKIPEDIA] El Club Atlético San Lorenzo de Almagro es una institución
deportiva y social centenaria, con sede en la ciudad de Buenos Aires, Argentina,
cuya principal actividad es el fútbol profesional. Es uno de los cinco grandes
del fútbol argentino.

SALIDA:

[WIKIPEDIA] El Club Atlético San Lorenzo de Almagro es una institución deportiva
y social centenaria, con sede en la ciudad de Buenos Aires, Argentina, cuya
principal actividad es el fútbol profesional. Es uno de los cinco grandes del
fútbol argentino. COSTO: 1.386e+007
```

[1] El hecho de que existían soluciones duplicadas fue advertido por uno de los jurados durante la defensa de esta tesis. La documentación de la operación *ShortestPath* (utilizada para retornar los primeros N resultados) indica que se puede usar un parámetro para evitar los duplicados quedando su uso para trabajo futuro [FST 03].

```
[dobleve IKIPEDIA] El Club Atlético San Lorenzo de Almagro es una institución
deportiva y social centenaria, con sede en la ciudad de Buenos Aires, Argentina,
cuya principal actividad es el fútbol profesional. Es uno de los cinco grandes
del fútbol argentino. COSTO: 1.386e+007

[dobleve i ka i PEDIA] El Club Atlético San Lorenzo de Almagro es una institución
deportiva y social centenaria, con sede en la ciudad de Buenos Aires, Argentina,
cuya principal actividad es el fútbol profesional. Es uno de los cinco grandes
del fútbol argentino. COSTO: 1.386e+007

[Wi ka IPEDIA] El Club Atlético San Lorenzo de Almagro es una institución
deportiva y social centenaria, con sede en la ciudad de Buenos Aires, Argentina,
cuya principal actividad es el fútbol profesional. Es uno de los cinco grandes
del fútbol argentino. COSTO: 1.386e+007

[dobleve i KIPEDIA] El Club Atlético San Lorenzo de Almagro es una institución
deportiva y social centenaria, con sede en la ciudad de Buenos Aires, Argentina,
cuya principal actividad es el fútbol profesional. Es uno de los cinco grandes
del fútbol argentino. COSTO: 1.386e+007

[dobleve i ka IPEDIA] El Club Atlético San Lorenzo de Almagro es una institución
deportiva y social centenaria, con sede en la ciudad de Buenos Aires, Argentina,
cuya principal actividad es el fútbol profesional. Es uno de los cinco grandes
del fútbol argentino. COSTO: 1.386e+007

[Wi ka i pe e de i a ] El Club Atlético San Lorenzo de Almagro es una institución
deportiva y social centenaria, con sede en la ciudad de Buenos Aires, Argentina,
cuya principal actividad es el fútbol profesional. Es uno de los cinco grandes
del fútbol argentino. COSTO: 1.386e+007

[Wi ka i pe EDIA] El Club Atlético San Lorenzo de Almagro es una institución
deportiva y social centenaria, con sede en la ciudad de Buenos Aires, Argentina,
cuya principal actividad es el fútbol profesional. Es uno de los cinco grandes
del fútbol argentino. COSTO: 1.386e+007

[Wi KIPEDIA] El Club Atlético San Lorenzo de Almagro es una institución deportiva
y social centenaria, con sede en la ciudad de Buenos Aires, Argentina, cuya
principal actividad es el fútbol profesional. Es uno de los cinco grandes del
fútbol argentino. COSTO: 1.386e+007

[dobleve i ka i pe e de i a ] El Club Atlético San Lorenzo de Almagro es una
institución deportiva y social centenaria, con sede en la ciudad de Buenos Aires,
Argentina, cuya principal actividad es el fútbol profesional. Es uno de los cinco
grandes del fútbol argentino. COSTO: 1.386e+007


/* ======================================== */

ENTRADA: [WIKIPEDIA] Hacia diciembre de 2010, San Lorenzo de Almagro es
clasificado por la IFFHS como el octavo mejor equipo sudamericano de la primer
década del siglo XXI.

SALIDA:

[WIKIPEDIA] Hacia diciembre de dos mil diez , San Lorenzo de Almagro es
clasificado por la IFFHS como el octavo mejor equipo sudamericano de la primer
década del siglo veintiuno . COSTO: 8.47328e+006

[WIKIPEDIA] Hacia diciembre de dos mil diez , San Lorenzo de Almagro es
clasificado por la IFFHS como el octavo mejor equipo sudamericano de la primer
década del siglo vigésimoprimer. COSTO: 8.47328e+006

[WIKIPEDIA] Hacia diciembre de dos mil diez , San Lorenzo de Almagro es
clasificado por la IFFHS como el octavo mejor equipo sudamericano de la primer
década del siglo vigésimoprimer. COSTO: 8.47328e+006

[WIKIPEDIA] Hacia diciembre de dos mil diez , San Lorenzo de Almagro es
clasificado por la IFFHS como el octavo mejor equipo sudamericano de la primer
década del siglo veintiuno . COSTO: 8.47328e+006

[WIKIPEDIA] Hacia diciembre de dos mil diez , San Lorenzo de Almagro es
clasificado por la IFFHS como el octavo mejor equipo sudamericano de la primer
década del siglo vigésimaprimera. COSTO: 8.47328e+006
```

```
[WIKIPEDIA] Hacia diciembre de dos mil diez , San Lorenzo de Almagro es
clasificado por la IFFHS como el octavo mejor equipo sudamericano de la primer
década del siglo vigésimoprimero. COSTO: 8.47328e+006

[WIKIPEDIA] Hacia diciembre de dos mil diez , San Lorenzo de Almagro es
clasificado por la IFFHS como el octavo mejor equipo sudamericano de la primer
década del siglo vigésimoprimero. COSTO: 8.47328e+006

[WIKIPEDIA] Hacia diciembre de dos mil diez , San Lorenzo de Almagro es
clasificado por la IFFHS como el octavo mejor equipo sudamericano de la primer
década del siglo vigésimaprimera. COSTO: 8.47328e+006

[WIKIPEDIA] Hacia diciembre de dos mil diez , San Lorenzo de Almagro es
clasificado por la IFFHS como el octavo mejor equipo sudamericano de la primer
década del siglo  vigésimoprimer. COSTO: 8.47329e+006

[WIKIPEDIA] Hacia diciembre de dos mil diez , San Lorenzo de Almagro es
clasificado por la IFFHS como el octavo mejor equipo sudamericano de la primer
década del siglo  vigésimoprimer. COSTO: 8.47329e+006
```

Finalmente, si existen casos en que el traductor no encuentra la solución para la línea de entrada a traducir (por ejemplo: caracteres disponibles en alfabetos en otro idioma diferente al español o inglés, no contemplados en la solución), se retorna la siguiente línea:

```
Traducción no encontrada para: + Línea de entrada a traducir
```

3.2.1 Composición

Para obtener la traducción de cada una de las frases, se realizan los siguientes pasos:

1. Se lee la frase (oración) de entrada
2. Se genera un WFST con dicha frase de entrada a traducir (con sólo un carácter por transición), donde la traducción de cada carácter es el mismo carácter. Este WFST es generado para luego componerlo con el *WFST completo* para obtener la traducción asociada.
3. Se realiza la **composición** del *WFST de la frase a traducir* con el *WFST completo* generado previamente. (Ver detalles de la composición en la sección de Compose del Capítulo II de OpenFST). La composición entre ambos WFST es realizada carácter a carácter. Como resultado de la composición se obtiene un *WFST con las frases traducidas*.
4. Se genera el texto asociado a cada frase traducida desde el WFST obtenido en el paso anterior. Para ello se recorre el camino desde el estado inicial hasta llegar al estado final, tomando cada uno de los caracteres traducidos en las transiciones de un estado al otro (Ver Capítulo II para más detalle). Para obtener la mejor frase se recorre el camino más corto (Shortest path), es decir el de menor peso. Para obtener las N-mejores frases se recorren cada uno de los N primeros caminos de menor peso (si es que existen ya que el WFST podría tener menos de N caminos, caso contrario devuelve todos los existentes).
5. Se guardan el/los textos traducidos en los archivos de salida mencionados (Mejor Resultado / N-Mejores Resultados)

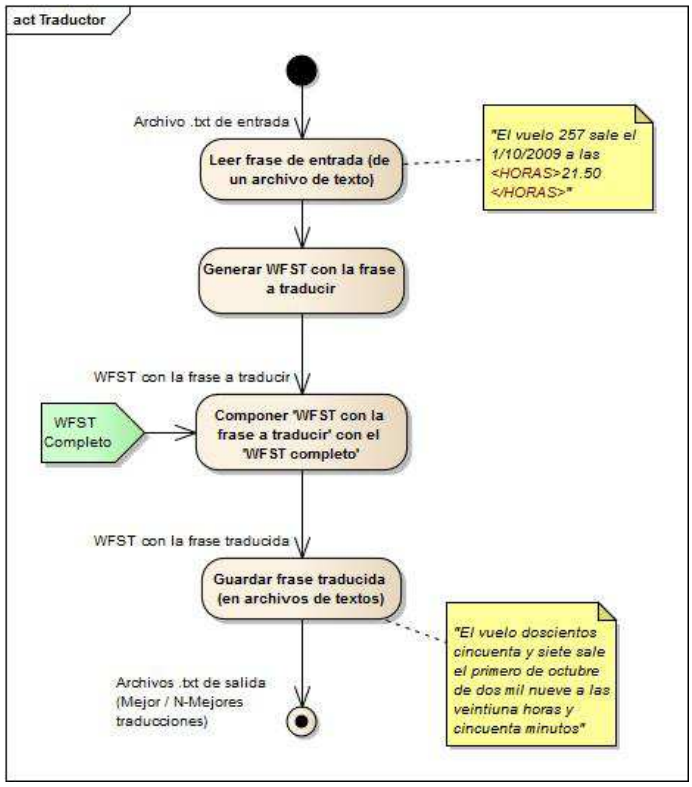


Figura 3.5 - Pasos de la Composición de WFSTs

3.3 Tags

Un *tag* es una secuencia de caracteres predefinida, para indicar un tipo de traducción a realizar. Los tags permiten favorecer el uso de un WFST específico en lugar de otro, para discriminar en los casos donde varias traducciones serían posibles. Para utilizar esta solución, se marca una determinada palabra entre dos tags, lo que permite identificar el tipo de traducción a realizar.

Por ejemplo, el siguiente caso tiene varias traducciones donde todas ellas serían correctas:

- 1° = primer, primero, primera o un grado

Entonces, en este caso de acuerdo al contexto, se asignaría previamente a la realizar la traducción, el tag correspondiente en el texto de entrada para que luego se traduzca correctamente. Luego, el caso planteado anteriormente se resolvería, como se detalla a continuación:

- `<ORD_NEUTRO>`1°`</ORD_NEUTRO>` Aniversario = primer Aniversario
- `<ORD_FEM>`1°`</ORD_FEM>` Edición = primera Edición
- `<ORD_MASC>`1°`</ORD_MASC>` B= primero be
- Temperaturas de `<GRADOS>`1°`</GRADOS>` = Temperaturas de un grado

Para realizar las traducciones utilizando los tags se debe incluir la siguiente secuencia en el texto de entrada:

1. Tag de Apertura (`<GRADOS>`)

2. Texto a Traducir (1°)
3. Tag de Cierre (</GRADOS>)

Los tags implementados comienzan con el símbolo reservado **<** y finalizan con el símbolo reservado **>**. Dichos símbolos no se encuentran incluidos en la lista de símbolos comunes a traducir, para evitar la expansión del tag en si mismo.

Cuando se identifica la secuencia de caracteres que indica un tag de apertura, el traductor debido a los pesos asignados, quedaría iterando en el WFST específico en lugar de ir a buscar a otra solución hasta que encuentra la secuencia de caracteres que indica un tag de cierre. Esto es, la secuencia que comienza con un tag, es consumida por el WFST correspondiente, debido al carácter reservado de apertura. Luego queda iterando en el WFST correspondiente al tag, en lugar de ir a buscar otra solución, debido a que los pesos seleccionados para los caminos con tags son menores, con lo cual lo favorece con respecto a otras soluciones posibles, hasta finalizar con el tag de cierre por el mismo motivo.

Cada WFST específico, como se mencionó anteriormente, tiene una clausura propia, que es lo que permite la iteración para procesar la misma parte del WFST entre dos tags.

En la siguiente lista se incluyen los tags soportados por el traductor para contemplar las traducciones particulares:

Descripción	Tag de Apertura	Tag de Cierre	Ejemplo
Naturales Masculinos	<NAT_MASC>	</NAT_MASC>	<NAT_MASC>21</NAT_MASC> = veintiuno <NAT_MASC>2</NAT_MASC> = dos
Naturales Femeninos	<NAT_FEM>	</NAT_FEM>	<NAT_FEM>21</NAT_FEM> = veintiuna <NAT_FEM>2</NAT_FEM> = dos
Ordinales Femeninos	<ORD_FEM>	</ORD_FEM>	<ORD_FEM>1°</ORD_FEM> = primera
Ordinales Masculinos	<ORD_MASC>	</ORD_MASC>	<ORD_MASC>1°</ORD_MASC> = primero
Ordinales Neutros	<ORD_NEUTRO>	</ORD_NEUTRO>	<ORD_NEUTRO>1°</ORD_NEUTRO> = primer
Grados	<GRADOS>	</GRADOS>	<GRADOS>1°</GRADOS> = un grado
Reales	<REAL>	</REAL>	<REAL>3,14</REAL> = tres coma catorce
Romanos Naturales	<ROM_NAT>	</ROM_NAT>	<ROM_NAT> XXI </ROM_NAT> = veintiuno
Romanos Ordinales Femeninos	<ROM_ORD_FEM>	</ROM_ORD_FEM>	<ROM_ORD_FEM> XXI </ROM_ORD_FEM> = vigésimo primera
Romanos Ordinales Masculinos	<ROM_ORD_MASC>	</ROM_ORD_MASC>	<ROM_ORD_MASC> XXI </ROM_ORD_MASC> = vigésimo primero
Romanos Ordinales Neutros	<ROM_ORD_NEUTRO>	</ROM_ORD_NEUTRO>	<ROM_ORD_NEUTRO> XXI </ROM_ORD_NEUTRO> = vigésimo primer
Siglas (Iniciales)	<SIGLA>	</SIGLA>	<SIGLA>DNI</SIGLA> = de ene i
Abreviaturas Comunes	<ABREV>	</ABREV>	<ABREV> Sr.</ABREV> = señor
Abreviaturas Singulares	<ABREV_SING>	</ABREV_SING>	<ABREV_SING> kg.</ABREV_SING> = kilogramo
Abreviaturas Plurales	<ABREV_PLURAL>	</ABREV_PLURAL>	<ABREV_PLURAL> kg.</ABREV_PLURAL> = kilogramos
Abreviaturas Masculinas	<ABREV_MASC>	</ABREV_MASC>	<ABREV_MASC> max.</ABREV_MASC> = máximo
Abreviaturas Femeninas	<ABREV_FEM>	</ABREV_FEM>	<ABREV_FEM> max.</ABREV_FEM> = máxima
Abreviaturas Especiales	<ABREV_ESP>	</ABREV_ESP>	<ABREV_ESP> ASCII</ABREV_ESP> = ASQUI
Fechas	<FECHA>	</FECHA>	<FECHA>04/07/12</FECHA> = cuatro de julio de dos mil doce
Horas	<HORAS>	</HORAS>	<HORAS>13:30</HORAS> = trece horas y treinta minutos
Teléfonos	<TELEFONOS>	</TELEFONOS>	<TELEFONOS>4747-1234</TELEFONOS> =

Descripción	Tag de Apertura	Tag de Cierre	Ejemplo
			cuatro siete cuatro siete uno dos tres cuatro
Importes	<IMPORTES>	</IMPORTES>	<IMPORTES>u$d 100</IMPORTES> = cien dólares
Fracciones	<FRACCIONES>	</FRACCIONES>	<FRACCIONES>3/4</FRACCIONES> = tres cuartos

Nota:

Los nombres de los tags fueron definidos arbitrariamente para el propósito del sistema, y en base a las traducciones disponibles. Queda fuera de alcance y como trabajo futuro, estudiar su analogía y/o conversión a otros sistemas de tagging semántico, como por ejemplo el provisto por Freeling [FreeLing].

3.4 Resumen del desarrollo

El sistema **Traductor de Texto Escrito a Texto Oral** desarrollado para implementar la solución al *preprocesamiento del texto de entrada* (o *normalización del texto*) para el idioma español, permite la conversión del texto escrito a su equivalente texto oral, es decir, como se leería en voz alta.

Para la implementación del sistema se utilizan dos programas: el primero para *generar el WFST completo* que incluye las traducciones posibles, y el segundo para *realizar las traducciones* utilizando el *WFST completo* generado previamente.

El *WFST completo* se encuentra compuesto por WFST específicos para realizar traducciones particulares (por ejemplo: naturales, abreviaturas, fechas, etc.).

El *traductor de texto* utiliza un texto de entrada en español con o sin tags, que pueden provenir de noticias de diarios, artículos en la Web, Wikipedia, etc. Los tags son secuencias de caracteres que se utilizan para favorecer o seleccionar una traducción específica, en particular cuando existe más de una solución correcta. En el presente trabajo, estos tags son asignados manualmente cuando es requerido indicar cual sería la traducción correcta según el contexto.

Para realizar la traducción se realiza la composición del WFST correspondiente al texto de entrada y del WFST completo construido anteriormente. Con esta operación de composición de traductores de autómatas finitos se obtiene un WFST con todas las traducciones posibles.

Los resultados de la traducción se guardan en dos archivos de texto. El primero con la mejor traducción y otro con las N-mejores traducciones y sus pesos para cada frase, donde las mejores son seleccionadas en base a su menor peso.

Los programas fueron desarrollados en Visual C++ importando las librerías de OpenFST para MS-Windows, para ejecutar en línea de comandos en DOS. Asimismo, los fuentes podrían ser compilados para Linux utilizando las librerías de OpenFST correspondientes para dicho entorno (Ver Apéndice A – Detalles de Implementación).

Se generaron dos ejecutables para cada uno de los programas: *buildFstTesis.exe,* utilizado para generar el WFST completo y *openFstTesisTest.exe,* utilizado para realizar la traducción. Los mismos utilizan archivos de entrada, como el archivo de texto con las frases a traducir, como así también generan archivos de salida como el *WFST completo* y los archivos de textos donde finalmente se obtiene un texto con el resultado de las frases traducidas a su equivalente oral.

4 Capítulo IV - Evaluación

En la presente sección se describe la evaluación del sistema **Traductor de Texto Escrito a Texto Oral** para el idioma español (detallado en el capítulo anterior), desde su preparación y su ejecución hasta el análisis de los resultados obtenidos luego de la misma. El objetivo de la evaluación es verificar el módulo de normalización del texto de entrada en un sistema de síntesis del habla, midiendo el grado de eficacia de las traducciones obtenidas de dicho módulo, teniendo como referencia (o *gold standard*) cómo leería el texto escrito un humano.

En el diagrama de la Figura 4.1 se muestra el contexto de la evaluación del sistema, donde se toman textos de diferentes fuentes en la Web, se consolidan en un archivo de texto plano para luego procesar en el **Traductor de Texto Escrito a Texto Oral** y obtener como resultados el texto equivalente para ser leído en voz alta (Mejor Resultado) y analizar las posibles traducciones del sistema (N-Mejores Resultados). En las siguientes secciones se describen los detalles de la misma.

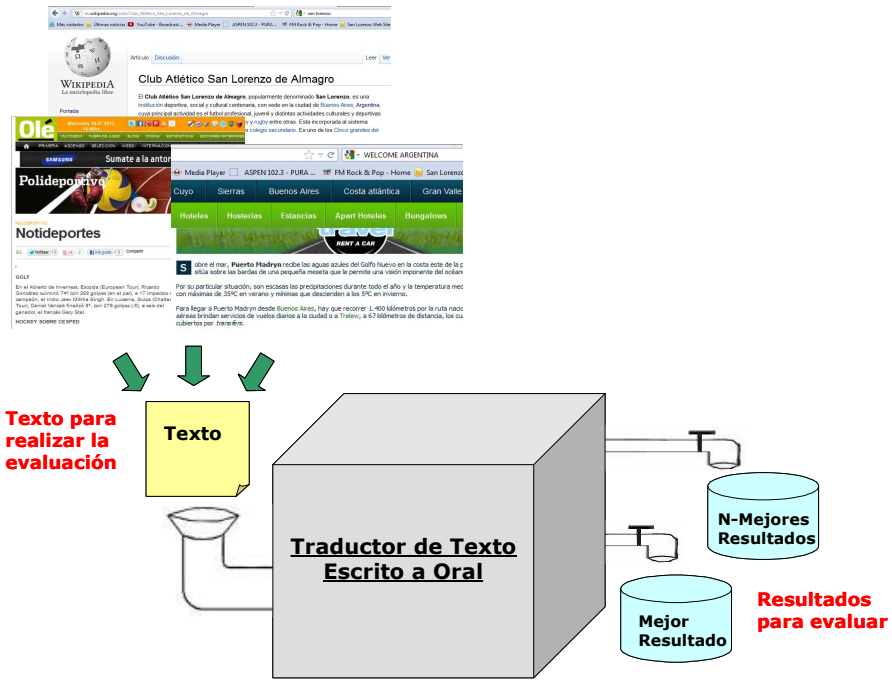


Figura 4.1 – Contexto de la evaluación

Como guía para realizar la evaluación, se utilizó la *Receta de los siete pasos* (definido por el grupo Eagles [EAG 99]), ya que aplica para sistemas PLN como el desarrollado en el presente trabajo. De acuerdo con la *Receta de los siete pasos* [MIL 01], para realizar la evaluación se debe conocer inicialmente el usuario y el propósito del sistema a evaluar. Luego se deben definir las tareas a realizar durante la misma, las características que serán evaluadas en base al propósito mencionado y las métricas para medir dichas características. Considerando estos lineamientos, se realiza la

ejecución de la evaluación y el análisis de los resultados. En las siguientes secciones se explica en detalle cada uno de los puntos mencionados del enfoque propuesto.

4.1 Descripción de usuarios y tareas

Para iniciar el proceso de la evaluación, la idea es identificar los tipos de usuarios y tareas a realizar por el sistema [EST 10] [MIL 01]. Los usuarios del sistema **Traductor de Texto Escrito a Texto Oral** a evaluar, serán lectores u oyentes de habla hispana de los textos (en español) de las distintas fuentes provistas. La fuente de la información a leer serán diarios en línea, artículos en la Web (Ej.: Wikipedia), documentos y reportes específicos, es decir información general, teniendo en cuenta que la información a traducir sería el texto incluido en dichas fuentes y no las imágenes.

Una aplicación posible del sistema será la lectura de páginas Web para personas con problemas visuales (no videntes o con visibilidad reducida), brindando la posibilidad a dichas personas de *escuchar* los contenidos de las páginas en español. Si bien el sistema a evaluar será un módulo que formará parte de una serie de módulos para un futuro sistema TTS, el mismo será evaluado en forma independiente, con respecto a los usuarios descriptos.

Entonces, para el contexto de la evaluación, a través del sistema se deben obtener las traducciones de texto oral correspondientes al texto escrito que desearía leer u oír el usuario. Dichos textos escritos, son provistos como fuente de información de entrada al sistema, obteniendo como salida del sistema el texto equivalente de cómo se leería en voz alta.

Esta tarea del sistema (traducir del texto escrito a su correspondiente texto oral) tiene un fuerte impacto en las características a evaluar, considerando que las traducciones deben ser correctas de acuerdo a lo que se debería leer, ya que el *lector* debe confiar en la salida que *escucharía* del sistema, fundamentalmente porque podría no estar *leyendo* el texto escrito, debido a dificultades visuales como se mencionó anteriormente. En este caso es importante que se traduzcan todas las palabras que se deben traducir, como así también que dichas traducciones sean correctas (Ej.: *atte.* debe leerse como *atentamente* y no como *atte*). Se deben tener en cuenta que en muchos casos la correctitud está relacionada con el contexto de las palabras a traducir, ya que varias traducciones podrían ser posibles (Ej.: *1°* sería *un grado* para el caso en que se encuentre en un texto que se habla de temperatura, y *primer* si se habla de puestos de una competencia).

Asimismo, el tiempo en el cual obtiene la traducción es un factor importante a considerar ya que sería deseable que la lectura sea prácticamente en línea.

Luego, se evaluará el sistema con estas necesidades en mente. A continuación se describen las características a ser evaluadas.

4.2 Características a ser evaluadas

Como se mencionó anteriormente existen características a ser evaluadas para verificar que las tareas del sistema sean cumplidas de manera deseable para el usuario mencionado en la sección previa.

Las principales características a ser evaluadas son las siguientes [FEMTI]: *Correctitud*, *Terminología* y *Tiempo de respuesta*. A continuación se describe cada una de ellas.

La idea de evaluar la característica de *correctitud* es medir el grado de distorsión con el cual la información contenida en el texto original ha sido reproducida en la traducción. Las distorsiones pueden ser provocadas por los siguientes factores:

- Ausencia de información (silencio): Palabras no traducidas (Ej.: siglo *XX* = siglo *XX* en lugar de siglo *veinte*)
- Interferencia (ruido): Palabras agregadas por el sistema (Ej.: *Di* Pieri = *quingentésimoprimer* Pieri en lugar de *Di* Pieri)
- Combinación entre ausencia e interferencia: Palabras mal traducidas (Ej.: a. C. = *área compañía* en lugar de siglo *antes de cristo*)

Por otra parte, evaluar la característica de *terminología* permite verificar que la traducción sea correcta para los términos específicos del dominio. Como se describió anteriormente, el dominio es general. Para evaluar esta característica es importante la completitud de los archivos de abreviaturas, siglas / palabras especiales, etc. ya que incide en los resultados, debido a que si existen palabras que no están incluidas en dichos diccionarios, las mismas no serían traducidas. Esta característica es evaluada en conjunto con la correctitud, ya que la amplia cobertura del vocabulario en español, impacta en los resultados de las traducciones correctas. No obstante, existen dominios específicos que no serán cubiertos como por ejemplo la expansión de abreviaturas de clasificados, recetas de cocina, términos técnicos, etc.

La evaluación de la característica de *tiempo de respuesta* permite verificar la capacidad del sistema de proveer un apropiado tiempo de respuesta para realizar las traducciones dado un texto de entrada. Es importante considerar que idealmente la traducción debería ser en línea, con lo cual el tiempo debería acercarse a cero.

4.3 Métricas para evaluar las características

La efectividad de un sistema de recuperación depende de la relevancia de las respuestas obtenidas. Para evaluar las características mencionadas en la sección anterior, se utilizarán métricas ampliamente conocidas en el área de la recuperación de la información: **precisión, recall** y **F-measure** [MAN 09] [GOD 12]. A continuación se describe cada una de ellas, ajustando la definición al contexto de las traducciones (que es lo que se quiere evaluar).

La Tabla 4.1 muestra la correspondiente matriz de confusión, donde figuran los datos que deberían ser traducidos (o no) por el sistema (columnas) y los que efectivamente fueron traducidos (o no) por el mismo (filas), independientemente de si son traducidos correctamente. Estos datos permiten tomar las mediciones previamente mencionadas.

	A Traducir	**A No Traducir**
Traducidas	Verdaderos Positivos (vp)	Falsos Positivos (fp)
No Traducidas	Falsos Negativos (fn)	Verdaderos Negativos (vn)

Tabla 4.1– Matriz de Confusión

La **precisión** es la cantidad de respuestas recuperadas que son relevantes; es decir, de las *palabras traducidas* por el sistema, es el porcentaje que fueron traducidas y que debían ser traducidas.

$$precisión = \frac{vp}{(vp + fp)} = \frac{ATraducir\_Traducidas}{Traducidas}$$

El **recall** es la cantidad de respuestas relevantes que son recuperadas; es decir, de las *palabras a traducir* por el sistema, es el porcentaje que fueron traducidas y que debían ser traducidas.

$$recall = \frac{vp}{(vp + fn)} = \frac{ATraducir\_Traducidas}{ATraducir}$$

La precisión y recall tienen poca utilidad por separado, ya que se puede tener un recall alto recuperando toda la información del conjunto aunque con muy baja precisión, como así también no recuperándola la precisión es alta y el recall tiende a cero. Es por ello que se usan medidas de balanceo entre ambos valores, como por ejemplo **F-measure**.

$$F - measure = \frac{(2 * precisión * recall)}{(precisión + recall)}$$

Luego de evaluar la cantidad de traducciones esperadas y efectuadas por el sistema, se presenta la necesidad medir la calidad de dichas traducciones. Esto es, de las palabras traducidas por el sistema (y que debía traducir), cuál fue el porcentaje de las palabras *correctamente* traducidas. Esta medida es conocida como **accuracy** y es frecuentemente usada para evaluar la efectividad de problemas de clasificación. Su fórmula se describe a continuación.

$$accuracy = \frac{Traducidas\_Correctas}{ATraducir\_Traducidas}$$

Por otra parte, para medir el *tiempo de respuesta* del sistema, se medirán los tiempos en realizar las traducciones, siendo que los mismos deberían ser minimizados ya que sería deseable realizarlas prácticamente en tiempo real. Es importante destacar, que se medirán los tiempos concretos para realizar las traducciones, ya que el tiempo esperable es una percepción muy subjetiva acerca de si el tiempo es adecuado o no.

4.4 Diseño de la evaluación

Antes de ejecutar la evaluación, se deben realizar una serie de tareas para su preparación. Las mismas fueron diseñadas y validadas a través de distintos los tests del sistema (Ver Apéndice B - Test).

Las siguientes son las tareas previas a la ejecución de la evaluación:

1. Selección del texto de entrada (Corpus)
2. Preprocesamiento del texto de entrada

3. Generación del texto de referencia
4. Tagging del texto de entrada
5. Selección de pesos
6. Selección de herramientas para la evaluación

A continuación se describe cada una de ellas.

4.4.1 Selección del texto de entrada (Corpus)

El **corpus** o **texto de entrada** para realizar la evaluación, ha sido seleccionado al azar desde diversas fuentes, como ser *diarios on-line* (El mundo, Critica, Crónica, Clarín, Ámbito, Infobae, La Prensa, La Nación, Página 12, El Eco Digital, Línea Capital y Diario Libre), *Wikipedia* (de los distintos tópicos mencionados a continuación) y otros *artículos de la Web* (Wikipedia, Ecolink, Biology Cabinet, Nasa, Literatura Argentina, LT10Digital, La Tercera, Agritotal, Todo Noticias, Campo en Acción, Infocampo, Arquitecto On-Line, Arquimaster y revista Mercado).

Los textos fueron elegidos de diez tópicos diferentes, para diversificar el vocabulario. Los tópicos seleccionados son los siguientes: ciencia, cultura, deportes, economía, espectáculos, interés general, lugares, política, tecnología y turismo. Los mismos se encuentran distribuidos en diez archivos (uno por cada tópico) de cien frases (aproximadamente) cada uno. Las frases son de distinta longitud y se encuentran separadas por un fin de línea. A continuación se incluye como ejemplo un fragmento de uno de ellos (turismo):

```
[DIARIO LIBRE]
Vacaciones en CHILE, Un 50% mas caras en dolares para los Argentinos
El dolar en chile esta a 500 pesos chilenos, el año pasado estaba a 650.
En Argentina el dolar cuesta 3,82 pesos hace un año costaba 3,20.
Las vacaciones en Chile cuestan un 50% mas caras en dolares este año.
[LUGARES DE VIAJE]
Para picar, almorzar o tentarse con algo dulce, 7 buenas razones para frenar en la
ruta.
Este lugar existe desde hace 44 años, cuando empezó siendo carnicería.
Todos los días, de 8.30 a 19.
En la salida de Mercedes, dirección a Lobos, hay una estación de servicio a mano
izquierda: siga 1.5 km, y a mano derecha, en un monte de eucaliptos sale el camino
que lleva a lo de los hermanos Hugo y Alberto Di Pieri.
Hace 30 años que elaboran chorizos, bondiolas, longanizas y el dos veces célebre
salame: por identidad mercedina y por el sello de calidad con nombre y apellido.
Todos los días de 7.30 a 18.
Carlos Gabba y su familia hacen chocolates muy ricos, una labor artesanal que
reproducen en 20 variedades con cacao importado de Ecuador.
La novedad: un chocolate 100% amargo.
Todos los días de 10 a 13 y 14:30 a 19:30.
[LUGARES DE VIAJE]
Los anteojos de Godard y los de Woody Allen, la baguette y el bagel, Quasimodo y
King Kong, los perfiles de JFK y CDG.
El libro no está a la venta en Argentina.
[WIKIPEDIA]
Walter Hunziker - Kurt Krapf, 1942 dicen que "El Turismo es el conjunto de
relaciones y fenómenos producidos por el desplazamiento y permanencia de personas
fuera de su domicilio, en tanto que dichos desplazamientos y permanencia no están
motivados por una actividad lucrativa".
El término "turismología" surgió en los años '60.
Pero fue el yugoslavo Ivadin Jovicic (geógrafo en su formación académica), el
científico considerado "padre de la turismología", quién lo popularizó cuando
fundó la revista del mismo nombre en 1972.
Son famosas las expediciones desde Venecia a Tierra Santa y las peregrinaciones
por el Camino de Santiago (desde el 814 en que se descubrió la tumba del santo),
```

```
fueron continuas las peregrinaciones de toda Europa, creándose así mapas, mesones
y todo tipo de servicios para los caminantes).
En Roma mueren 1500 peregrinos a causa de una plaga de peste bubónica.
A finales del siglo XVI surge la costumbre de mandar a los jóvenes aristócratas
ingleses a hacer el Grand Tour al finalizar sus estudios con el fin de
complementar su formación y adquirir ciertas experiencias.
Era un viaje de larga duración (entre 3 y 5 años) que se hacía por distintos
países europeos, y de ahí proceden las palabras: turismo, turista, etc.
Es posible afirmar que los viajes de placer tuvieron sus inicios en los últimos
años del siglo XIX y los primeros del siglo XX.
El siglo XIX fue testigo de una gran expansión económica, seguida de una
revolución industrial y científica incluso mayor en la segunda mitad del siglo XX.
El turismo fue uno de los principales beneficiarios, para llegar a ser a finales
del siglo XX la mayor industria del mundo.
```

4.4.2 Preprocesamiento del texto de entrada

Como se explicó anteriormente (Ver Capítulo III), existen textos que deben ser **preprocesados** antes de realizar la traducción. Esto es para realizar la conversión a un formato predefinido aceptado por el traductor. Dichas conversiones se encuentran fuera del alcance del sistema, como se detalló en los WFSTs específicos en el capítulo mencionado.

Las conversiones a ser realizadas (manualmente para el alcance de la evaluación del presente trabajo) son las siguientes:

- **Naturales, reales, importes**: Remover símbolos separadores de miles.
- **Ordinales**: Modificar por el símbolo ° (símbolo de grado) las distintas ocurrencias del símbolo y abreviaturas que acompañan al número (Ej.: *4ª, 4to.* y *4ta.* se reemplazan como *4°*).
- **Fechas**: Modificar por el número de mes correspondientes para los meses representados en letras y en romanos (Ej.: *16/diciembre/1993*, *16/dic/1993* y *16/XII/93* se reemplazan como *16/12/1993*).
- **Símbolos**: Reemplazar los que se encuentran fuera la lista de símbolos reconocidos por el sistema (Ver 6.3 Caracteres del Sistema) por uno equivalente (Ej.: º por °).

En el siguiente ejemplo se muestra una de las conversiones realizadas:

Texto de origen:

```
La 4ª edición de la maratón fue realizada el 16/dic/1993 donde se recorrieron
5.716 km. y el 1er puesto fue alcanzado por el campeón olímpico.
```

Texto convertido:

```
La 4° edición de la maratón fue realizada el 16/12/1993 se recorrieron 5716 km. y
el 1° puesto fue alcanzado por el campeón olímpico.
```

4.4.3 Generación del texto de referencia

Para validar la correctitud de las traducciones realizadas por el sistema se usan **textos de referencia** análogos a los de entrada pero con las traducciones realizada por humanos. Los textos de referencia fueron traducidos manualmente por diferentes personas, que fueron seleccionadas con distintos perfiles a modos de ampliar el espectro de vocabulario para generar referencias heterogéneas y representativas de la población.

A cada una de las mismas se les solicitó que, para un texto del corpus de entrada dado, generen su equivalente según cómo lo leerían en voz alta. Para realizar dicha traducción se los guió con algunas

técnicas a modo de no distorsionar el texto de entrada (por ejemplo, que se respeten mayúsculas, minúsculas, idioma de origen, etc.) pero no se influyó en las decisiones tomadas para realizar las traducciones.

En el siguiente ejemplo se muestra una de las referencias generadas:

Texto de origen:

```
Vacaciones en CHILE, Un 50% mas caras en dolares para los Argentinos
El dolar en chile esta a 500 pesos chilenos, el año pasado estaba a 650.
En Argentina el dolar cuesta 3,82 pesos hace un año costaba 3,20.
Las vacaciones en Chile cuestan un 50% mas caras en dolares este año.
```

Texto de referencia:

```
Vacaciones en CHILE, Un cincuenta por ciento mas caras en dolares para los
Argentinos
El dolar en chile esta a quinientos pesos chilenos, el año pasado estaba a
seiscientos cincuenta.
En Argentina el dolar cuesta tres pesos con ochenta y dos centavos hace un año
costaba tres con veinte
Las vacaciones en Chile cuestan un cincuenta por ciento mas caras en dolares este
año.
```

4.4.4 Tagging del texto de entrada

El proceso de **tagging del texto de entrada**, es el proceso de añadir los tags definidos en el capítulo III (ver 3.3 Tags) a cada una de las palabras según los tipos definidos en el mismo capítulo. Este proceso fue realizado manualmente, agregando según el contexto de cada palabra el tag correspondiente. Para facilitar la tarea se comparó el texto de entrada con el texto análogo de referencia (descriptas en secciones anteriores), siendo las diferencias las palabras candidatas para añadir los tags.

En el siguiente ejemplo se muestra un fragmento de un archivo con sus correspondientes tags:

Texto de origen:

```
Vacaciones en CHILE, Un 50% mas caras en dolares para los Argentinos
El dolar en chile esta a 500 pesos chilenos, el año pasado estaba a 650.
En Argentina el dolar cuesta 3,82 pesos hace un año costaba 3,20.
Las vacaciones en Chile cuestan un 50% mas caras en dolares este año.
```

Texto con tags:

```
Vacaciones en CHILE, Un 50% mas caras en dolares para los Argentinos
El dolar en chile esta a 500 pesos chilenos, el año pasado estaba a 650.
En Argentina el dolar cuesta <REAL>3,82</REAL> pesos hace un año costaba
<REAL>3,20</REAL>.
Las vacaciones en Chile cuestan un 50% mas caras en dolares este año.
```

4.4.5 Selección de pesos

Para realizar la **selección de pesos** adecuada para correr la evaluación del sistema (Ver 3.1 Generar WFST completo), se ejecutaron una serie de tests preliminares con distintas combinaciones de pesos para el mismo archivo de entrada. Luego se compararon los resultados de las traducciones y las referencias para el mismo en búsqueda de la solución de pesos óptima. Los tests fueron ejecutados en un laboratorio del Departamento de Computación de la Facultad de Ciencias Exactas

y Naturales de la UBA, en quince PCs en forma simultánea, con el fin de optimizar los tiempos para ejecutar numerosas combinaciones de pesos y factor.

El algoritmo empleado se detalla a continuación. La idea del mismo es explorar distintas combinaciones de pesos internos, pesos finales y factores para encontrar la mejor combinación que permita optimizar el tiempo y minimizar las diferencias entre las traducciones esperadas y realizadas.

Algoritmo para Selección de Pesos

Sean los pesos p1, p2, ... pn con una asignación inicial elegida arbitrariamente (pi != 0 para todo i), con n = 7

Sean los pesos finales pf1, pf2, ... pfm con una asignación inicial elegida arbitrariamente (pfj != 0 para todo j) con m = 7

Sean los factores f1, f2, ... fq con una asignación inicial elegida arbitrariamente (fk != 0 para todo k) con q = 10 .

```
1   para i = 1..n
2     para j = 1..m
3       para k = 1..q
4         Peso = {pi, pfj, fk}
5         evaluar sobre los datos
6         si el resultado es mejor, seleccionar el nuevo Peso
7         si el resultado no es mejor, seleccionar el Peso original.
8       fin
9     fin
10 fin
```

Los pesos fueron elegidos arbitrariamente, con el conocimiento adquirido durante el desarrollo de cuales serían mejores candidatos. La ejecución de los tests con la aplicación de pesos mencionados en el algoritmo, fue realizada sobre un mismo texto de entrada, utilizado para este fin, compuesto por cien frases de diferentes tópicos (turismo, espectáculos, deportes, humor, etc.) de diversos artículos y periódicos on-line para obtener traducciones de distinto tipo (Ver los valores y rangos de pesos utilizados en el algoritmo en Apéndice B - Test de Selección de Pesos).

Asimismo la traducción obtenida por el sistema para la combinación de pesos empleada en cada ejecución fue comparada con el mismo texto de referencia con las traducciones realizadas por un humano, con el fin de encontrar cuales eran las diferencias entre ambas traducciones.

Finalmente, la mejor combinación de pesos sería la que obtiene menor cantidad de diferencias entre ambas traducciones, ya que este hecho indica la similitud entre ambas traducciones. Dicha combinación de pesos es la que luego es utilizada para construir el *WFST completo*.

Los archivos utilizados para el test no participan de la evaluación, sólo fueron utilizados como muestra para la selección de pesos.

Para ejecutar cada uno de los tests se corrió un script en Python, que se describe a continuación. Para más detalle de cada uno de los pasos ver la sección 4.5 Ejecución de la evaluación, más adelante en el presente capítulo.

Algoritmo para Ejecución de Test

1 Asignación de peso interno (p), peso final (pf) y factor seleccionados para el test:

   p = pi

   pf = pfj

   factor = fk

2 Construir el WFST completo para {p, pf, factor}

3 Traducir el texto de entrada con el WFST completo

4 Comparar resultados obtenidos con los esperados y registrar las diferencias

5 fin

Luego de realizar aproximadamente setenta ejecuciones de los tests (Ver Apéndice B - Test de Selección de Pesos), el resultado de combinación de pesos seleccionado para realizar la evaluación es la siguiente, porque tiene la menor cantidad de diferencias entre las frases de referencias (traducidas por humanos) y las frases traducidas por el sistema y que además las retorna en el menor tiempo posible con respecto a las otras pruebas corridas en PCs con configuraciones similares.

- p = 1
- pf = 10
- factor = 5000

4.4.6 Herramienta de evaluación

Para ser evaluados y medir su efectividad, los resultados de la traducción realizada por el sistema deben ser comparados con los textos de entrada (a traducir) y sus correspondientes referencias (traducciones realizadas por humanos). Para ello es utilizada una **herramienta de comparación de archivos** desarrollada especialmente, que permite cuantificar las diferencias entre los archivos con el resultado de la traducción realizada, el del texto a traducir y el archivo de referencias. La herramienta de comparación de archivos mencionada se encuentra desarrollada en C++ standard y está basada en la biblioteca *diff* [Diff Code], proyecto de Google que provee algoritmos para realizar operaciones para archivos de texto plano.

Esta herramienta toma como entrada dos archivos de texto a comparar y como resultado devuelve un archivo de texto con las diferencias marcadas y con la cantidad total de diferencia encontradas (Ver sintaxis en Apéndice B - de Comparación de archivos (Diff)). Este archivo con las diferencias permite analizar de los resultados obtenidos, así como establecer las mediciones y conclusiones que se detallan en las siguientes secciones.

En el siguiente ejemplo se muestra un archivo con las diferencias encontradas:

```
Vacaciones en CHILE, Un <50%,cincuenta por ciento> mas caras en dolares para los
Argentinos
El dolar en chile esta a <500,quinientos> pesos chilenos, el año pasado estaba a
<650,seiscientos cincuenta>.
En Argentina el dolar cuesta <3,82,tres> pesos <con ochenta y dos centavos >hace
un año costaba <3,20.,tres con> <veinte>
Las vacaciones en Chile cuestan un <50%,cincuenta por ciento> mas caras en dolares
este año.
```

Si bien la herramienta de *diff* es altamente efectiva para establecer las diferencias entre archivos de textos, existen numerosos errores que pueden distorsionar la cantidad de diferencias encontradas. Dichos errores están relacionados en el análisis de las secuencias de caracteres entre ambos archivos, dado que dicha secuencia puede no corresponder con la secuencia de palabras respectiva que se desea establecer realmente como diferencia. Para realizar la evaluación estos errores son corregidos manualmente para permitir establecer las diferencias correctamente. La cantidad de errores presente en los archivos depende de los textos comparados y el orden de las secuencias de caracteres, con lo cual no es posible establecer claramente una tasa de errores de este tipo. A continuación se muestran algunos ejemplos de ello:

Diferencias encontradas:

```
<k, kiló>m<.,etros>
<14:30,catorce  horas  y> <treint>a <19:30, minutos  a diecinueve  horas  y
treinta  minutos >
<4200,cuatro> m<il do>s<cie>n<tos  >m<etros sobre el nivel del mar>
```

Diferencias esperadas:

```
<km., kilómetros>
<14:30,catorce  horas  y treinta minutos> a <19:30, diecinueve  horas  y treinta
minutos >
<4200,cuatro mil doscientos> <msnm, metros sobre el nivel del mar>
```

4.5 Ejecución de la evaluación

Para la ejecución de la evaluación se realizó una serie de pasos que permitieron obtener las traducciones y el análisis de la correctitud de las mismas. Como se muestra en la Figura 4.2, en primer lugar se **construye el *WFST Completo*** (por única vez), con todas las traducciones posibles del sistema. Luego se utiliza dicho *WFST Completo* para **traducir el texto** correspondiente a cada uno de los tópicos seleccionados. Este paso se ejecuta sucesivas veces, una para cada texto de entrada que se desea traducir. Una vez generadas las traducciones por el sistema se **comparan los resultados** obtenidos con los esperados, es decir, se comparan las traducciones del sistema con las traducciones de referencia (realizadas por humanos) y se toman mediciones para analizar la eficacia del sistema. Este paso se ejecuta una vez por cada texto de salida obtenido.

Figura 4.2 – Ejecución de la evaluación

El primer paso consistió en construir el *WFST completo* con los pesos seleccionados descriptos en la sección anterior (Ver 4.4.5 Selección de pesos). Este paso se ejecutó una sola vez, debido a que con el mismo WFST fue posible realizar todas las traducciones para los distintos tópicos. Para construir el *WFST completo* se ejecutó el programa correspondiente (BuildFstTesis.exe). [2]

Luego de generar el *WFST completo* se realizó la traducción para cada tópico, es decir ejecutar el programa que realiza la traducción para cada uno de los diez archivos de entrada descriptos en la sección anterior (Ver 4.4.1 Selección del texto de entrada (Corpus)). Por otra parte, dicho proceso fue también ejecutado para los archivos análogos con tags. Una vez ejecutado el programa traductor se obtuvieron dos archivos con los resultados para cada uno de los tópicos: el que contiene la Mejor

[2] Para mayor detalle sobre la ejecución del mismo ver en el Apéndice A la sección 8.2.1 buildFSTTesis.exe.

traducción y el que contiene las N-Mejores traducciones (Ver 3.2 Traducir texto). Para realizar la traducción se ejecutó el programa correspondiente (OpenFstTesisTest.exe). [3]

Una vez generados los resultados de las traducciones se realizaron las distintas comparaciones para analizar los mismos y obtener las métricas de la evaluación. Las comparaciones que se realizaron son las siguientes:

- **Texto de Entrada vs. Referencias**: para obtener las traducciones a realizar (palabras a traducir).
- **Texto de Entrada vs. Mejor Resultado**: para obtener las traducciones realizadas por el sistema (palabras traducidas).
- **Texto de Referencias vs. Mejor Resultado**: para medir la correctitud de las traducciones realizadas por el sistema (palabras traducidas correctamente).

En la Figura 4.3 se muestra la comparación de los textos mencionados, donde se puede observar en las relaciones entre los mismos cuales son las mediciones obtenidas de acuerdo a lo explicado previamente.

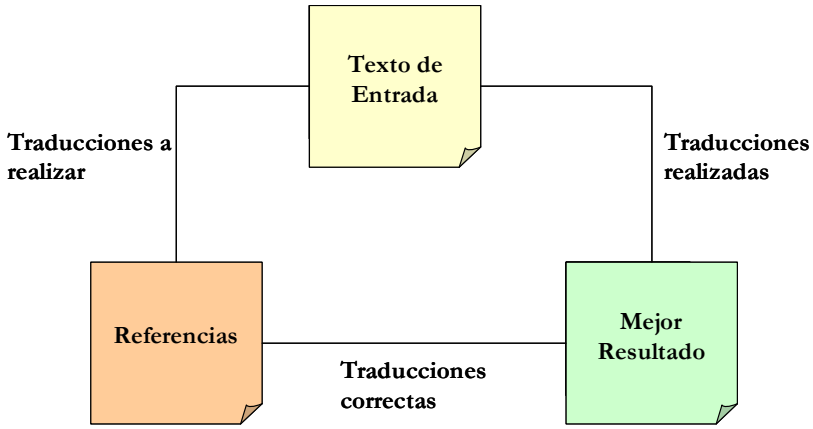


Figura 4.3 – Comparación de Textos

Para realizar las comparaciones se ejecutó el programa correspondiente (Diff.exe). [4]

4.6 Resultados de la evaluación

En la presente sección, se detallan los resultados obtenidos luego de la ejecución de la evaluación previamente descripta y el análisis de los mismos junto a las métricas para evaluar su efectividad. El análisis se enfoca a evaluar las diferencias entre los archivos de entrada, los archivos de salida generados por el sistema y las referencias realizadas por humanos.

En primer lugar, es importante destacar que el sistema respondió correctamente para las palabras que no deben ser traducidas, es decir las que se encontraron escritas de la misma manera en que se

[3] Para mayor detalle sobre la ejecución del mismo ver en el Apéndice A la sección 8.2.2 openFSTTesisTest.

[4] Para mayor detalle sobre la ejecución del mismo ver en el Apéndice B la sección 9.3.1. Comparación de Archivos.

leen. Esto significa que no se encontraron casos donde palabras que se encontraban escritas tal como se leen en el texto de entrada, hayan sido traducidas y/o cambiadas en el texto de salida obtenido por el sistema. Luego, las traducciones realizadas por el sistema siempre correspondían a palabras que debían ser traducidas.

En la Tabla 4.2 se pueden observar las traducciones para cada uno de los tópicos elegidos, donde se muestran la cantidad de **palabras a traducir** por el sistema (obtenidas de la comparación entre el texto de entrada y las referencias), la cantidad de **palabras traducidas** por el sistema (obtenidas de la comparación entre el texto de entrada y el de salida), y las **correctamente traducidas** (obtenidas de la comparación entre el texto de salida y las referencias) tanto para textos de entrada **con y sin tags**, dado que como se observa el uso de tags permite obtener mayor cantidad de traducciones correctas ya que se le puede indicar el tipo de traducción deseada para resolver los casos ambiguos (donde mas de una traducción es posible). No obstante existen casos que no fueron traducidos dado que no han sido contemplados para el presente trabajo, aún con el uso de tags, como se describe más adelante en el presente capítulo (Ver Análisis de errores).

Tópicos	Palabras a Traducir	Palabras Traducidas (sin Tags)	Correctas Traducidas (sin Tags)	Palabras Traducidas (con Tags)	Correctas Traducidas (con Tags)
Ciencia	131	121	68	131	130
Cultura	162	150	120	162	160
Deportes	158	140	124	157	156
Economía	205	192	167	205	179
Espectáculos	108	98	80	108	97
Interés General	218	211	195	216	202
Lugares	188	162	127	175	158
Política	182	165	150	175	163
Tecnología	144	117	106	139	131
Turismo	156	140	116	146	136

Tabla 4.2 – Traducciones por tópicos

4.6.1 Traducción sin Tags

Los resultados de realizar la traducción sin agregar tags (para desambiguar casos particulares según el contexto) se pueden observar en la Tabla 4.3, donde figuran los porcentajes obtenidos de comparar las traducciones realizadas y esperadas, tanto para las métricas de precisión y recall, como F-Measure para balancear las mismas (ver sección 4.3 Métricas para evaluar las características).

Traducción sin Tags	Precisión	Recall	F-Measure
Ciencia	99,2%	92,4%	95,7%

Traducción sin Tags	Precisión	Recall	F-Measure
Cultura	98,7%	92,6%	95,5%
Deportes	95,9%	88,6%	92,1%
Economía	100,0%	93,7%	96,7%
Espectáculos	95,1%	90,7%	92,9%
Interés General	99,5%	96,8%	98,1%
Lugares	100,0%	86,2%	92,6%
Política	99,4%	90,7%	94,8%
Tecnología	100,0%	81,3%	89,7%
Turismo	98,6%	89,7%	94,0%

Tabla 4.3 – Precisión / Recall para Traducción sin Tags

En la Figura 4.4 se muestran los porcentajes previamente calculados y las variaciones existentes entre los tópicos para las mediciones tomadas. Gráficamente se puede observar los valores obtenidos son altos para las medidas consideradas y estables entre los tópicos, lo que muestra un buen comportamiento del sistema.

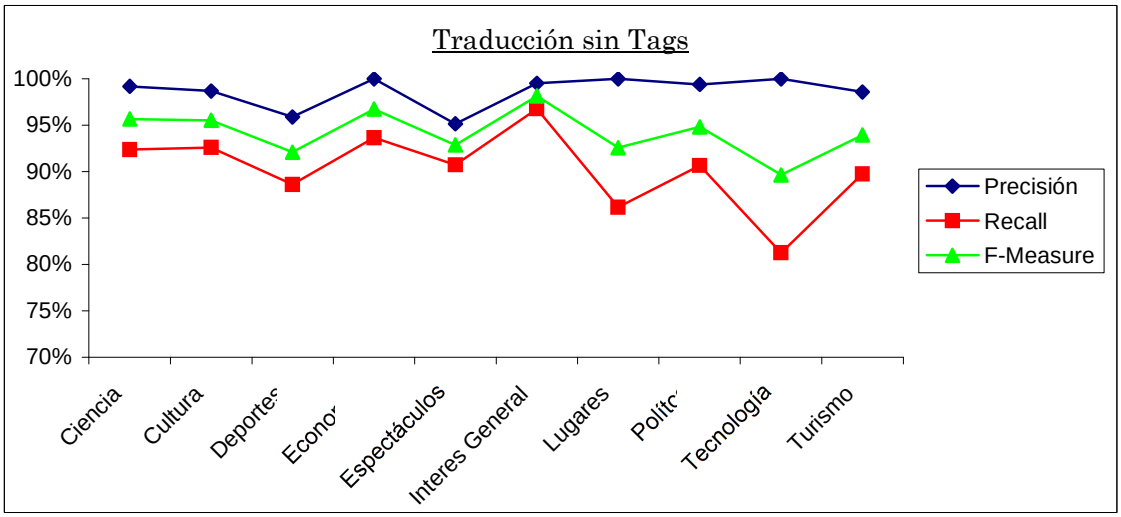


Figura 4.4 – Precisión / Recall / F-Measure para Traducción sin Tags

4.6.2 Traducción con Tags

Los casos de siglas, romanos, horas, abreviaturas y números por género se resolvieron desambiguando con tags. En Tabla 4.4 se muestran los resultados obtenidos luego de realizar las traducciones para un texto de entrada con tags. Las medidas consideradas son análogas a las tomadas en la traducción sin tags explicadas en la sección anterior.

Traducción con Tags	Precisión	Recall	F-Measure
Ciencia	99,2%	100,0%	99,6%
Cultura	98,8%	100,0%	99,4%
Deportes	100,0%	99,4%	99,7%
Economía	100,0%	100,0%	100,0%
Espectáculos	95,6%	100,0%	97,7%
Interés General	99,5%	99,1%	99,3%
Lugares	100,0%	93,1%	96,4%
Política	99,4%	96,2%	97,8%
Tecnología	100,0%	96,5%	98,2%
Turismo	98,6%	93,6%	96,1%

Tabla 4.4 – Precisión / Recall para Traducción con Tags

Análogamente a lo explicado en la sección anterior en la Figura 4.5 se muestra las variaciones entre las medidas tomadas entre los tópicos.

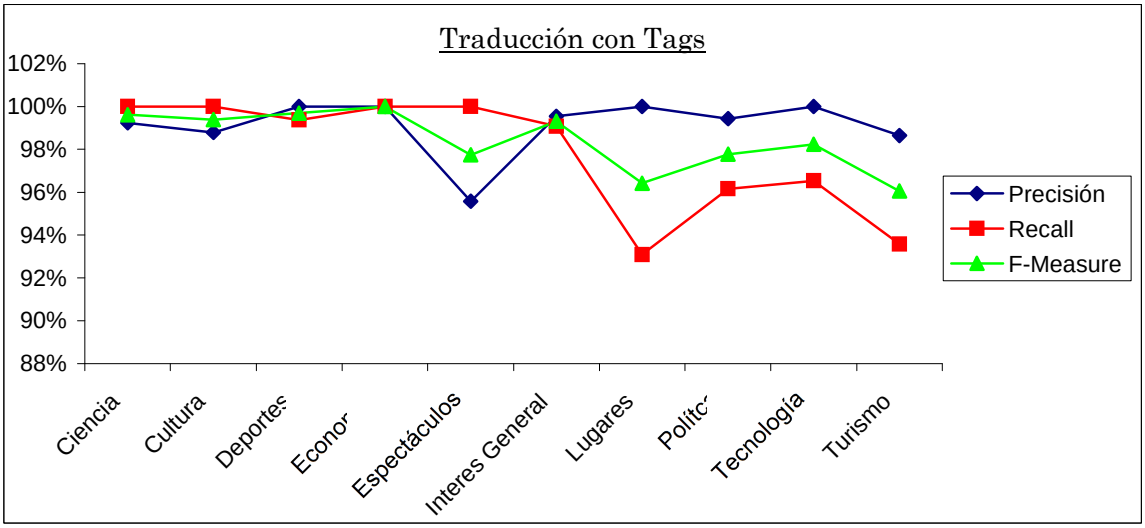


Figura 4.5 - Precisión / Recall / F-Measure para Traducción con Tags

4.6.3 Traducción sin Tags vs. con Tags

Para ver en más detalle la efectividad del uso de tags, que como se mencionó anteriormente, resuelven muchos de los casos donde se presenta ambigüedad, cuando el contexto permite indicar cuál sería la traducción correcta, se utiliza la medida conocida como **accuracy** que permite conocer cuál fue el porcentaje de las palabras *correctamente* traducidas (ver sección 4.3 Métricas para evaluar las características).

En la Tabla 4.5 se observan los resultados de aplicar la medida de accuracy para los textos de entrada con y sin tags para cada uno de los tópicos.

Accuracy	Sin Tags	Con Tags
Ciencia	56,2%	99,2%
Cultura	80,0%	98,8%
Deportes	88,6%	99,4%
Economía	87,0%	87,3%
Espectáculos	81,6%	89,8%
Interés General	92,4%	93,5%
Lugares	78,4%	90,3%
Política	90,9%	93,1%
Tecnología	90,6%	94,2%
Turismo	82,9%	93,2%

Tabla 4.5 – Comparación de traducciones con y sin Tags

En la Figura 4.6 se muestra la comparación de la medida de accuracy (calculada en la tabla anterior) entre traducciones realizadas con y sin tags para el texto de entrada. Se puede observar que la correctitud de las traducciones mejora notablemente con la incorporación de tags y que se mantiene dicha tendencia para cada uno de los tópicos.

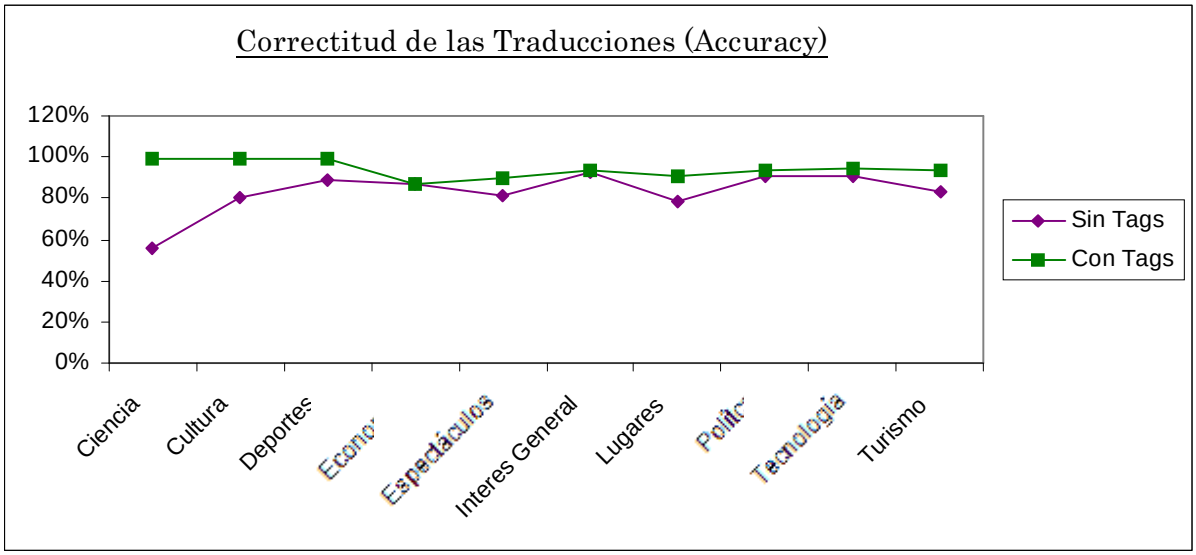


Figura 4.6 – Comparación de traducciones con y sin Tags

4.6.4 N-Mejores Resultados

Las N-mejores traducciones se obtienen utilizando la operación de *n-shortest paths (n-caminos más cortos)* en el WFST con todas las traducciones posibles. Los *n-caminos más cortos* son los *n* caminos que tienen los menores pesos para obtener la traducción (Ver Capítulo II para mayor detalle). Existen casos donde la primera opción retornada no fue la traducción esperada, pero evaluando las N-mejores, se pudo observar que se encuentra la traducción correcta.

Entonces, esta operación permite conocer si la solución correcta es devuelta entre las mejores, en particular para evaluar los casos donde hay empates en los pesos de los caminos y solo uno de ellos es considerado como el mejor (arbitrariamente). Es importante considerar que dicha operación tiene un costo de procesamiento y que crece linealmente con el tamaño de N [FST ShortestPaths].

En la Tabla 4.6 figuran los casos donde dicha solución fue encontrada entre las N-mejores (con N = 10) para cada uno de los tópicos. En esa tabla, las **palabras no traducidas correctamente** indican que el sistema no realizó la traducción, o bien, realizó una traducción diferente a la esperada en primer lugar. Las **palabras traducidas en las N-Mejores** son las palabras que se encontraron correctamente traducidas entre la segunda y la décima mejor traducción (para el caso de N = 10). Con estos datos se realizó el cálculo del porcentaje de casos resueltos (donde se encontró la traducción deseada) entre los N-mejores resultados. Por ejemplo, en Turismo se encontraron correctamente traducidas 14 (entre las 10 mejores soluciones) de las 40 palabras no traducidas correctamente en primer lugar, lo que significó que se resolvieron un 35% de los casos entre el segundo y décimo mejor resultado.

Tópicos	Palabras no traducidas correctamente	Palabras traducidas en las N-Mejores	Porcentaje traducido entre las N-Mejores
Ciencia	74	14	18,9%
Cultura	44	23	52,3%
Deportes	39	4	10,3%
Economía	39	0	0,0%
Espectáculos	33	4	12,1%
Interés General	24	0	0,0%
Lugares	61	11	18,0%
Política	32	3	9,4%
Tecnología	38	3	7,9%
Turismo	40	14	35,0%

Tabla 4.6 – Palabras traducidas entre las N-Mejores traducciones

4.6.5 Análisis de errores

En esta sección se analizan las diferencias encontradas entre las traducciones realizadas por el sistema y las referencias. Existen casos que no pudieron ser resueltos con tags debido a su naturaleza, o bien, porque el traductor no contemplaba la traducción esperada según las referencias.

En el caso de las **abreviaturas** se observó como generalidad que cuando la abreviatura coincide con un fin de frase en el texto de entrada, el sistema no conserva el punto en la traducción para indicar el fin de frase como se presenta en las referencias. Por otra parte se encontraron casos particulares que no fueron correctamente traducidos por el sistema, como por ejemplo las abreviaturas compuestas por más de una palabra (*a.C. = antes de Cristo*, *R.P. = ruta provincial*) que el sistema las tradujo por separado (*a.C. = área compañía*, *R.P. = reverendo padre*) ya que figuraban ambas opciones en el listado de abreviaturas, o bien, casos que coincidían con números romanos (*CD* traducido como *cuatrocientos* en lugar de *ce de),* como así también, el caso que en el listado figuraba la abreviatura sin punto y fue expandido dentro de una palabra (*Radio* figuraba como *radiodifución* y *Radiofonía* fue traducida como *Radiodifución fonía*). También existieron casos donde la abreviatura no fue encontrada en dicho listado (*/ton. = por tonelada*, *disp. = disposición*, *m2 = metro cuadrado*), o bien, no figuraban todas las ocurrencias de escritura de la misma abreviatura (*FF.AA. = FFAA = fuerzas armadas*), con lo cual las mismas no fueron expandidas por el sistema. Cabe destacar que esta lista de abreviaturas podría ser actualizada para ampliar la cobertura de las traducciones como se explicó en el capítulo de desarrollo (ver sección 3.1.3 Abreviaturas).

Con respecto a las **fechas** se presenta el caso de que el primer día del mes es usualmente leído como *primero* (así se indica en las referencias) a pesar que se encuentra escrito como *uno*, luego para el caso *1 de mayo* el sistema lo traduce como *uno de mayo* en lugar de *primero de mayo* como se esperaba, ya que no tiene indicación de ordinal que acompañe al número para que el sistema lo reconozca como tal.

En el caso de las **horas** el sistema no puede discriminar a priori si se trata de un número real o de una hora (*22.30* puede ser *veintidos punto treinta* o bien *veintidos horas y treinta minutos*) con lo que ambas traducciones son posibles, siendo que será necesario conocer el contexto para elegir la opción correcta. Otro caso presentado son los períodos, donde en las referencias se agregan palabras que el sistema no contempla (*de 10 a 13 y 14:30 a 19:30* se espera como *de diez a trece y 'de' catorce treinta a diecinueve treinta*).

Para los **importes** se observaron casos donde el orden y la cantidad de palabras esperados no coinciden con la traducción del sistema (*$ 100 millones* es traducido como *cien pesos millones* en lugar de *cien millones de pesos* o *u$s 100 mil* es traducida como *cien dólares mil* en lugar de *cien mil dólares*), debido a que a la cifra se agregan palabras para indicar el número en el texto de entrada. Otro caso es el símbolo de dólar que se encuentra representado de numerosas maneras en el texto de entrada (*u$s*, *USD*, etc.) y si alguna de ellas no está contemplada en el traductor, la misma no es expandida. En el caso de los decimales, el traductor los expande como tal (dígito por dígito) en lugar de indicarlos como centavos como figuraba en algunas referencias.

Los **naturales** presentaron también casos especiales, siendo el más destacado cuando los mismos denotan un período (*1993-2003*) dado que el sistema lo interpreta como número telefónico, ya que tiene el mismo formato, y son traducidos como tal (dígito por dígito) (Ver sección 3.1.11 Teléfonos). También se presentó el caso donde se indica el período *2011/2012* en el texto de entrada, donde el sistema lo expande como *doscientos uno 'un medio' cero doce* ya que interpreta el *1/2* como *un medio.* El caso del género para las centenas no se encuentra contemplado en el sistema con lo cual no es expandido como se encuentra en las referencias (*450 personas* es traducido como *cuatrocientos cincuenta personas* en lugar de *cuatrocientas cincuenta personas*), como así también, el caso del *uno* que se debería expandir como *un* según el contexto (*1 punto* es traducido como *uno punto* en lugar de *un punto*).

Los **reales** por su parte presentaron casos particulares como el de *1.5 km.* que fue expandido como *uno punto cinco kilómetros* y en las referencias se indicaba como *un kilómetro y medio* y otro como *0,414*

que los decimales se esperaban como centenas (*cero coma cuatrocientos catorce*) en lugar de dígito por dígito (*cero coma cuatro uno cuatro*). Cabe destacar que estos casos, ambas traducciones son correctas, tanto las referencias como las del sistema.

Por último con respecto a los **símbolos** se observó que en muchos casos el mismo símbolo puede ser traducido de manera diferente según el contexto (por ejemplo el símbolo /, que se lee *54/175 = barra* o *s/ = sobre* o *2012/2013 = a* o *1/2 = no se lee*) y en algunos casos existen caracteres usados como símbolos que no se encuentran soportados por el sistema y que deberían ser incluidos para mayor cobertura (por ejemplo el *apóstrofo* en *D'Alambert*).

Muchos de los casos se podrían resolver simplemente incorporando tags especiales para indicar el tratamiento apropiado de la traducción, como así también extender la cobertura de las traducciones del sistema, quedando estos para trabajo futuro. Ver el detalle de los casos encontrados y las soluciones propuestas en el Apéndice B – Tests (sección Detalle de las diferencias encontradas en la evaluación).

Como podemos ver, en la mayoría de los casos, el texto traducido por el sistema no es estrictamente incorrecto, ya que al ser sintetizado por un sistema TTS comunicará el mensaje, aunque en dichos casos exista una manera más elegante de traducirlo.

4.6.6 Tiempos de respuesta

Como se indicó en la sección 6.2, una de las características importantes a ser evaluada es el tiempo de respuesta del sistema. Para realizar esta evaluación se tomaron los tiempos de inicio y finalización para traducir el texto de entrada por el sistema para cada uno de los tópicos elegidos.

La evaluación fue ejecutada en un equipo con las siguientes especificaciones técnicas:

- Procesador Intel Core i3 3.06 Ghz
- 2 GB de RAM
- Sistema Operativo Microsoft Windows 7 (32 bits)

En la Tabla 4.7 se muestran las mediciones de tiempo para realizar las traducciones para cada uno de los tópicos. Se incluyen la **cantidad de frases, palabras** y **caracteres** traducidos para mostrar una idea de la magnitud del tiempo insumido para realizar las correspondientes traducciones. Como referencia, se puede considerar que la cantidad de frases a traducir ocupan entre seis y ocho páginas según el tópico, aproximadamente. En las últimas columnas figuran los **tiempos,** especificados en **hora:minutos:segundos,** para obtener las traducciones de los **textos de entrada con** y **sin tags**. Como se puede observar, si bien se obtienen mejores resultados para los textos de entrada con tags, estos demoran más tiempo (debido a que se tiene mayor cantidad de caracteres a traducir, siendo esta cantidad variable de acuerdo a la cantidad de tags que se encuentren incorporados según el tópico).

Tópicos	Cantidad de frases	Cantidad de palabras	Cantidad de caracteres	Tiempo sin Tag	Tiempo con Tag
Ciencia	113	3095	19736	0:01:29	0:05:27

Tópicos	Cantidad de frases	Cantidad de palabras	Cantidad de caracteres	Tiempo sin Tag	Tiempo con Tag
Cultura	116	3216	19110	0:01:14	0:01:42
Deportes	103	2484	13862	0:00:56	0:01:12
Economía	119	3356	19827	0:01:13	0:01:43
Espectáculos	121	2755	15674	0:01:00	0:01:17
Interés General	120	3423	20321	0:01:33	0:01:43
Lugares	116	3363	20485	0:01:23	0:02:07
Política	113	2851	16519	0:01:48	0:01:52
Tecnología	116	3144	18781	0:01:33	0:03:01
Turismo	114	2555	14914	0:02:30	0:04:59

Tabla 4.7 – Mediciones por tiempo

5 Capítulo V - Discusión

El sistema **Traductor de Texto Escrito a Texto Oral** fue evaluado para medir el grado de eficacia de sus traducciones, tomando como gold standard o referencia cómo leería el texto escrito un humano.

Para ejecutar la evaluación se prepararon previamente los archivos de entrada, seleccionando frases al azar de diez tópicos diferentes (turismo, economía, interés general, etc.) para ampliar el espectro del texto a traducir, provenientes de diversas fuentes (diarios en línea, artículos de la Web, etc.). También se prepararon los textos de referencias con traducciones realizadas por humanos, según cómo leerían los textos de entrada en voz alta. Por otro lado, se realizó la selección previa de pesos adecuados en un conjunto de test independiente, que luego se utilizaron en la evaluación.

Para tomar las métricas seleccionadas para evaluar las principales características del sistema (precisión, recall, F-measure y accuracy) y realizar el análisis de los resultados, se utilizó una herramienta de comparación de archivos (diff) desarrollada para tal fin. Esto permitió establecer las diferencias entre los textos a traducir, textos traducidos por el sistema y textos de referencias generados por humanos.

De acuerdo a las mediciones exhibidas, se observó que las traducciones realizadas por el sistema para textos de entrada con tags, mejoran la efectividad notablemente debido a que según el contexto es posible indicar cuál sería la traducción correcta, tal como se había asumido en el desarrollo. No obstante, el tiempo utilizado para realizar dichas traducciones con tags es mayor que para su respectivo texto de entrada sin tags.

Por otra parte, las mediciones de precisión y recall mostraron que los porcentajes de las palabras que fueron traducidas y que debían ser traducidas por el sistema se acercaron al 100% y la medida del F-measure, que balancea ambos valores, fue mayor al 90%, lo que implica un alto grado de cubrimiento en las traducciones realizadas.

En muchos casos, también se observó que si la traducción correcta no se encontraba como primera solución, la traducción deseada podía encontrase entre la segunda y la décima. Esto permite pensar como idea para el futuro, que se podrían analizar los pesos seleccionados según el contexto de la palabra para que la solución deseada mejore el ranking y sea devuelta en primer lugar.

En relación a las diferencias encontradas entre las referencias realizadas por humanos y las traducciones realizadas por el sistema, se observó que en general dichas diferencias estuvieron relacionadas con los números *romanos* (que fueron expandidos como ordinales y se esperaban como numerales), *siglas* (que no fueron expandidas por el sistema) y el género de los *naturales* (cuando se esperaba femenino y el sistema retornaba masculino). Todos estos casos fueron resueltos con tags ya que ayudaron a desambiguar según el contexto. Otro caso fue el de los símbolos especiales como apostrofe ('), ampersand (&), barra (/), guión (-), etc., que según el contexto deberían ser traducidos de manera diferente, quedando esto para resolverlo como trabajo futuro.

También se observó que existen diferencias particulares que ocurren según el tópico. Por ejemplo, en *Tecnología* y *Deportes* hubo ocurrencias de siglas sin expandir; en *Ciencia* ocurrieron casos de períodos de años tomados como teléfonos; en *Turismo* con casos como *camionetas 4 x 4* que debían ser traducidas como *cuatro por cuatro*; en *Cultura* se mostraron casos de romanos que fueron expandidos como se mencionó anteriormente; en *Interés General* con casos de horas expresadas de manera diferente a las referencias; en *Economía* donde se encontraron importes y cifras con

decimales expresados diferente a la esperada; y *Lugares* con abreviaturas específicas no expandidas. Con esto se puede ver que la elección de los distintos tópicos ofreció la posibilidad de abordar distintas problemáticas del lenguaje en español, lo cual permitió un mejor análisis en la evaluación.

Con respecto al tiempo de respuesta, se observó que sería un factor a optimizar, ya que el mismo debería ser mínimo para simular una traducción en línea. Se puede plantear como una idea para trabajo futuro, particionar el WFST completo en varios más pequeños y utilizarlos por separado de acuerdo al contexto de la palabra a traducir, lo que reduciría notablemente el tiempo de búsqueda.

5.1 Temas futuros

Durante el desarrollo de esta tesis se identificaron ciertos aspectos a mejorar y otros para resolverlos como trabajo futuro. A continuación se describen los temas principales que permitirían extender la cobertura y comportamiento del presente trabajo.

Extensión de la cobertura del vocabulario:

Existen casos que si bien fueron cubiertos satisfactoriamente por la tesis, podrían ser extendidos para ampliar la cobertura del vocabulario a traducir. Un caso particular es el de las **abreviaturas** cuando coinciden con un fin de frase. En este caso cuando se realiza la expansión de la misma el punto final se pierde en la traducción (ya que es el que normalmente la acompaña). Luego, se podrían implementar técnicas para conservarlo en este contexto. Otro caso especial es el de las **fechas** escritas con palabras, donde el primer día del mes que normalmente se lee como *primero,* a pesar de que el símbolo escrito sea un *1* (Ej.: *1* de mayo = *primero* de mayo). Para el caso de los **importes**, existen distintas ocurrencias de escritura para tener en cuenta, como la combinación de números y palabras *($50 millones = cincuenta millones de pesos* en lugar de *cincuenta pesos millones*), los diferentes símbolos de dólares (*USD 20 = veinte dólares*), la inclusión de centavos (Ej.: *$3,82 = tres pesos con ochenta y dos centavos*), etc. Los **grados** podrían contemplar los sistemas de temperaturas Celsius y Fahrenheit con sus respectivas representaciones y abreviaturas (*27,1 °C = veintisiete grados Celsius con una décima*). Por último, los **naturales** y **reales** podrían incluir más casos dependientes del contexto, como el género en las centenas (*400 personas = cuatrocientas personas*) y las ocurrencias donde *1* se lee como *un* (*26,1 puntos = veintiseis coma un puntos*). Como se puede observar, cada tema (es decir cada WFST específico construido) tiene su problemática, con lo que un enfoque posible sería abordar cada uno de ellos para ser exhaustivo con el vocabulario del mismo.

Aprendizaje del contexto para casos ambiguos:

Uno de los temas principales identificado y mencionado, es la expansión de los **casos ambiguos**, donde más de una traducción es posible para la misma palabra. Si bien en la presente tesis muchos de ellos fueron tratados con el uso de tags específicos para desambiguar, existen otros enfoques posibles como el **aprendizaje del contexto** de la palabra ambigua. Esto es, de acuerdo a las palabras que acompañan a la palabra ambigua, aprender e inferir cuál sería la traducción correcta para la misma, asignándole un mejor peso al WFST de dicha solución.

Como problemas de ambigüedad a resolver se identificaron los siguientes, entre otros:

- **Género de ordinales:** De acuerdo a las palabras que acompañan al número ordinal se puede inferir el género del mismo, como por ejemplo:
  - 1° puesto -> *primer* puesto
  - 1° edición -> *primera* edición

- 1° A -> *primero* A

- **Número Romanos**: Un caso similar al anterior es cuando se trata de un número romano que según las palabras que lo acompañan se puede tratar de un ordinal o un cardinal, como por ejemplo:
  - Enrique VIII -> Enrique *octavo*
  - Capítulo VIII -> Capítulo *ocho*
- **Símbolos**: Existen casos de símbolos que se leen de diferentes maneras de acuerdo al contexto en que se encuentren (Ej.: apostrofe ('), ampersand (&), barra (/), guión (-), punto (.)) Por ejemplo, el **guión** según el contexto, se lee de manera diferente: en el caso de los resultados deportivos, el guión que se encuentra presente entre los números del encuentro habitualmente se lee como *a,* con lo cual para este caso específico existe un significado diferente del símbolo, que en otros se leería como *guión* o *menos*. Entonces, se debería inferir su significado de acuerdo a las palabras del contexto dado, como por ejemplo:
  - Ganó 4-1 -> Ganó cuatro *a* uno (y no Ganó cuatro *menos* uno, ni tampoco Ganó cuatro *guión* uno)
  - la diferencia 4-1 -> la diferencia cuatro *menos* uno
  - la resolución 207-89 -> resolución doscientos siete *guión* ochenta y nueve / resolución doscientos siete *del* ochenta y nueve
  - en el período (1993-2001) -> en el período mil novecientos noventa y tres *al* dos mil tres

Palabras extranjeras:

En el caso de la identificación de las palabras extranjeras, que quedaron fuera del alcance del presente trabajo, podemos observar que muchas de ellas son incorporadas en textos del idioma español, como por ejemplo: *shopping*, *show*, *quorum*, *déjà vu*, *playoffs*, *jazz*, *windsurf*, etc. y nombres propios en otro idioma.

El tratamiento de las mismas requiere la identificación del idioma de origen y su posterior asociación a la conversión de grafema a fonema según la pronunciación requerida para la misma. Esta tarea quedaría para implementarse como otro módulo (en la fase de conversión fonética) a incorporar en un sistema TTS.

Conversiones a formatos estándar:

Las **fechas** en sus diferentes formatos se podrían detectar, con la implementación de un módulo previo y llevarlo a un formato estándar (Ej.: *16/VI/2011 -> 16/06/2011*, *16/Junio/2011 -> 16/06/2011*, etc.) para luego expandir dicho formato estándar con el sistema implementado en el presente trabajo (*dieciséis de junio de dos mil once*).

Con respecto a los **números con separadores de miles**, los mismos deberían identificarse con el formato de numeración en español (separador de miles = . y separador de decimales = ,) y luego removerse para indicar cuando se trata de la parte entera del número (Ej.: 15.500 -> 15500).

Para el caso de los **ordinales**, existen numerosas ocurrencias para escribirlos, luego es importante llevarlos a un formato común para luego expandir dicho formato con el sistema implementado en el presente trabajo (Ej.: *4to. -> 4°* (ordinal masculino), *4ª -> 4°* (ordinal femenino), etc.).

Traducciones específicas del dominio:

Existen casos donde las traducciones son dependientes del dominio y el vocabulario a ser tratado es muy específico. Estos casos se encuentran fuera del alcance de la presente tesis y se propone su implementación como trabajo futuro, ya que abordarlos es una problemática en sí misma, como por ejemplo: la expansión de **funciones matemáticas** (*log* = *logaritmo*, *sen* = *seno*, etc.), **Urls** (https://www.dc.uba.ar/), **clasificados** (*Dto.* en lugar céntrico en alquiler, 2 *amb.*, con lavarropas *aut.*- heladera con dos puertas *c*/congelador), **recetas de cocina** (Champignones *100 g*, Harina *1/2* kg, Aceite de oliva *10 cc*, sal *1 cda.*) y **tecnología** (1 *Kb., 2 Mb., browsers, WebOS, CD, DVD, iPad, email*) entre otros.

Aprendizaje de cómo incorpora el ser humano el lenguaje:

Un tema interesante que surgió durante el desarrollo de la tesis es cómo los humanos vamos aprendiendo nuestro lenguaje, cómo vamos incorporando nuevos términos que luego vamos a utilizar al leer para hacerlo correctamente. Por ejemplo, un niño de primer grado en su lectura todavía no tiene incorporados los números romanos y las abreviaturas, con lo cual es esperable que estos términos sean leídos literalmente (como se encuentran escritos). Es a partir del tiempo y del aprendizaje que estos nuevos términos son incorporados al vocabulario. Luego, si bien se trata de otra disciplina, sería interesante conocer este proceso para luego intentar imitarlo en un sistema de TTS, es decir, es un desafío plantearnos: ¿Cómo programamos nuestros algoritmos de aprendizaje para incorporar los términos de nuestro lenguaje?

6 Apéndice A – Detalles de Implementación

6.1 Codificación

La solución implementada se desarrolló en *C++* con *MS-Visual C++ 2008 Express Edition* para MS Windows. El código es C++ standard, por lo que funciona en otras plataformas (como por ejemplo Linux)

Para utilizar las operaciones de *traductores de autómatas finitos con peso* (WFST) se importó la librería de *OpenFST*. El paquete se encuentra disponible para bajar desde la página de la librería [FST 04] y se utilizó la versión 1.1. Aunque actualmente existen versiones mas recientes, la versión utilizada soporta adecuadamente las operaciones necesarias para la implementación. Ver detalles de la librería y las operaciones utilizadas en el Capítulo II – OpenFST.

Los programas generados pueden ejecutarse en MS-Windows, en una línea de comandos en DOS. No se requiere la instalación de la librería OpenFST, ni otras librerías, para ejecutar los mismos.

6.2 Ejecutables

Los archivos ejecutables generados para utilizar el sistema son los siguientes:

- **buildFstTesis.exe:** utilizado para generar el *WFST completo*
- **openFstTesisTest.exe:** utilizado para realizar la traducción

Quienes estén interesados en el uso de los archivos ejecutables pueden solicitarlos al director y/o a la alumna de la tesis. A continuación se incluye mayor detalle de cada uno de ellos para su utilización:

6.2.1 buildFstTesis

La **línea de comando** para ejecutar el programa *buildFstTesis* (desde DOS) es la siguiente:

```
buildFstTesis [Peso Interno] [Peso Final] [Factor] [archivo WFST completo]
```

donde los parámetros especificados son los siguientes:

- **Peso Interno**: Peso asignado para las transiciones de cada uno de los WFST (con excepción de los caracteres literales e iniciales).
- **Peso Final**: Peso asignado para el estado final de cada uno de los WFST con excepción de los caracteres literales e iniciales)
- **Factor**: Valor que se usa para multiplicar los pesos interno y final ingresados como parámetros para usarlos como pesos de los WFST de caracteres literales e iniciales.
- **Archivo WFST completo**: Archivo binario que contiene el traductor completo generado por el sistema que luego es utilizado para realizar las traducciones.

Los **archivos de entrada** utilizados por el programa *buildFstTesis* son los siguientes:

Archivo	Nombre de Archivo	Descripción
Símbolos de Entrada	nat.isyms	Archivo de texto que contiene la lista de símbolos de entrada de los textos a traducir
Símbolos de Salida	nat.osyms	Archivo de texto que contiene la lista de símbolos de salida de los textos traducidos
Abreviaturas Comunes	abreviaturas.txt	Archivo de texto que contiene el listado de abreviaturas comunes
Abreviaturas Por Género	AbrevGenero.txt	Archivo de texto que contiene el listado de las abreviaturas por género
Abreviaturas Singular/Plural	abrevPlural.txt	Archivo de texto que contiene el listado de las abreviaturas singular / plural
Abreviaturas Especiales	abrevEspeciales.txt	Archivo de texto que contiene el listado de abreviaturas especiales

El **archivo de salida** generado por el programa *buildFstTesis* es el siguiente:

Archivo	Nombre de Archivo	Descripción
WFST Completo	Definido por el usuario	Ver Sintaxis

6.2.2 openFstTesisTest

La **línea de comando** para ejecutar el programa *openFstTesisTest* (desde DOS) es la siguiente:

```
openFstTesisTest [Frases a traducir] [N-Mejores Frases traducidas] [Mejores
Frases traducidas] [cantidad de resultados] [WFST completo] [WFST de test]
```

donde los parámetros especificados son los siguientes:

- **Frases a traducir**: Archivo de entrada con las frases a ser traducidas (en formato ANSI), con o sin tags.

  Existen dos (2) casos especiales en la traducción con tags:

  - **Romanos:** se debe intercalar un espacio en blanco entre el Tag de Apertura y la secuencia de letras que representa un número romano y otro espacio en blanco entre el Tag de Cierre para que el mismo sea considerado como tal (ver explicación en sección Generar Romanos en el presente capítulo).

    - **Abreviaturas:** se debe intercalar un espacio en blanco entre Tag de Apertura y la abreviatura para que la misma sea considerada como tal (ver explicación en sección Generar Abreviaturas en el presente capítulo).

- **N-Mejores Frases traducidas:** Archivo de salida (en formato UTF8), con los N-mejores resultados (determinados por el menor peso) de la traducción de cada una de las frases de entrada, donde N se indica como parámetro.
- **Mejores Frases traducidas:** Archivo de salida (en formato UTF8) con la mejor (determinada por el menor peso) traducción de cada una de las frases de entrada, donde N se indica como parámetro.
- **Cantidad de resultados:** Valor que indica la cantidad de resultados para retornar las N-mejores por cada frase.
- **WFST completo:** Archivo binario de entrada que contiene el traductor completo generado por el sistema a utilizar para realizar las traducciones.
- **WFST de test:** Archivo binario de salida que contiene las frases traducidas por el sistema para luego generar los archivos de texto de salida.

Los **archivos de entrada** utilizados por el programa *openFstTesisTest* son los siguientes:

Archivo	**Nombre de Archivo**	**Descripción**
Símbolos de Entrada	nat.isyms	Archivo de texto que contiene la lista de símbolos de entrada de los textos a traducir
Símbolos de Salida	nat.osyms	Archivo de texto que contiene la lista de símbolos de salida de los textos traducidos
WFST Completo	Definido por el usuario	Ver sintaxis
Frases a Traducir	Definido por el usuario	Ver sintaxis

Los **archivos de salida** generados por el programa *openFstTesisTest* son los siguientes:

Archivo	**Nombre de Archivo**	**Descripción**
WFST de Test	Definido por el usuario	Ver sintaxis
N-Mejores Frases traducidas	Definido por el usuario	Ver sintaxis
Mejores Frases traducidas	Definido por el usuario	Ver sintaxis

6.3 Caracteres del Sistema

La siguiente es la lista de caracteres interpretadas por el sistema. La misma se encuentra compuesta por los caracteres que se encuentran presentes en el idioma español.

eps	0	I	73	s	115
(espacio)	32	J	74	t	116
!	33	K	75	u	117
"	34	L	76	v	118
#	35	M	77	w	119
$	36	N	78	x	120
%	37	O	79	y	121
&	38	P	80	z	122
'	39	Q	81	{	123
(	40	R	82	|	124
)	41	S	83	}	125
*	42	T	84	~	126
+	43	U	85	¡	161
,	44	V	86	«	171
-	45	W	87	°	176
.	46	X	88	º	186
/	47	Y	89	»	187
0	48	Z	90	¿	191
1	49	[	91	Á	193
2	50	\	92	Ä	196
3	51	]	93	É	201
4	52	^	94	Ë	203
5	53	_	95	Í	205
6	54	`	96	Ï	207
7	55	a	97	Ñ	209
8	56	b	98	Ó	211
9	57	c	99	Ö	214
:	58	d	100	Ú	218
;	59	e	101	Ü	220
<	60	f	102	á	225
=	61	g	103	ä	228
>	62	h	104	é	233
?	63	i	105	ë	235
@	64	j	106	í	237
A	65	k	107	ï	239
B	66	l	108	ñ	241
C	67	m	109	ó	243
D	68	n	110	ö	246
E	69	o	111	ú	250
F	70	p	112	ü	252
G	71	q	113		
H	72	r	114		

6.4 Símbolos del Sistema

La siguiente es la lista de símbolos interpretados por el sistema y su correspondiente traducción.

!	!	«	«
"	"	»	»
;	;	¿	¿
?	?	:	:
[	[	|	|
]	]	/	/
^	^	\	\
`	`	#	numeral
{	{	%	por ciento
}	}	&	y
'	'	=	igual
(	(	~	tilde
)	)	@	arroba
‚	‚	*	asterisco
-	-	+	más
.	.	$	pesos
_	_	espacio	espacio
¡	¡		

7 Apéndice B – Test

7.1 Casos de Test

Para realizar el test se generó un archivo de texto con cien (100) frases obtenidas de diferentes fuentes de diarios on-line, artículos de la Web y Wikipedia. Se seleccionaron frases de diferentes tópicos, con preferencia con traducciones a realizar incluidas en las mismas, para abordar distintas problemáticas del lenguaje español. A continuación se muestran las primeras frases del archivo:

```
[WIKIPEDIA] El Club Atlético San Lorenzo de Almagro es una institución deportiva
y social centenaria, con sede en la ciudad de Buenos Aires, Argentina, cuya
principal actividad es el fútbol profesional. Es uno de los cinco grandes del
fútbol argentino.

[WIKIPEDIA] Hacia diciembre de 2010, San Lorenzo de Almagro es clasificado por la
IFFHS como el octavo mejor equipo sudamericano de la primer década del siglo XXI.

[WIKIPEDIA] Luego de varios meses, se realizó la primera asamblea extraordinaria,
el 1 de abril de 1908, fecha establecida como el día de su fundación. En ese
momento se decidió elegir el nombre del flamante club.

[WIKIPEDIA] Durante este período se crea el barrio de Boedo mediante la Ordenanza
N° 23698 del 11 de junio de 1968.

[WIKIPEDIA] Posee 162 peñas, de las cuales 124 se encuentran distribuidas a lo
largo y a lo ancho de toda la Argentina, y 28 en el exterior, ubicadas en:
Alemania, Andorra, Australia, Chile, 10 en España (Andalucía, Madrid, Barcelona,
Fuengirola, Oliva, Santander, Tarragona, Tenerife, Valencia y Zaragoza), 5 en
Estados Unidos (Arizona, Hawái, Miami, Nueva York y Texas), Grecia, Inglaterra,
Italia, Israel, Japón, México, Perú, Puerto Rico y Uruguay. En la actualidad
cuenta con 33000 socios aproximadamente.
```

Por otra parte se generó un archivo de referencia con las traducciones realizadas por un humano, de cómo leería en voz alta las mismas frases, para permitir la comparación con las traducciones obtenidas por el sistema. Las siguientes frases son las traducciones de referencias generadas para las primeras frases antes mencionadas.

```
[WIKIPEDIA] El Club Atlético San Lorenzo de Almagro es una institución deportiva
y social centenaria, con sede en la ciudad de Buenos Aires, Argentina, cuya
principal actividad es el fútbol profesional. Es uno de los cinco grandes del
fútbol argentino.

[WIKIPEDIA] Hacia diciembre de dos mil diez, San Lorenzo de Almagro es
clasificado por la I EFE EFE HACHE ESE como el octavo mejor equipo sudamericano
de la primer década del siglo veintiuno.

[WIKIPEDIA] Luego de varios meses, se realizó la primera asamblea extraordinaria,
el uno de abril de mil novecientos ocho, fecha establecida como el día de su
fundación. En ese momento se decidió elegir el nombre del flamante club.

[WIKIPEDIA] Durante este período se crea el barrio de Boedo mediante la Ordenanza
número veintitres mil seiscientos noventa y ocho del once de junio de mil
novecientos sesenta y ocho.

[WIKIPEDIA] Posee ciento sesenta y dos peñas, de las cuales ciento veinticuatro
se encuentran distribuidas a lo largo y a lo ancho de toda la Argentina, y
veintiocho en el exterior, ubicadas en: Alemania, Andorra, Australia, Chile, diez
en España (Andalucía, Madrid, Barcelona, Fuengirola, Oliva, Santander, Tarragona,
Tenerife, Valencia y Zaragoza), cinco en Estados Unidos (Arizona, Hawái, Miami,
Nueva York y Texas), Grecia, Inglaterra, Italia, Israel, Japón, México, Perú,
Puerto Rico y Uruguay. En la actualidad cuenta con treinta y tres mil socios
aproximadamente.
```

Luego al ejecutar el programa de traducción (openFstTesisTest.exe) se obtuvo el resultado de las traducciones realizadas por el sistema, como se muestra a continuación para las primeras frases.

```
[WIKIPEDIA] El Club Atlético San Lorenzo de Almagro es una institución deportiva
y social centenaria, con sede en la ciudad de Buenos Aires, Argentina, cuya
principal actividad es el fútbol profesional. Es uno de los cinco grandes del
fútbol argentino.

[WIKIPEDIA] Hacia diciembre de dos mil diez , San Lorenzo de Almagro es
clasificado por la IFFHS como el octavo mejor equipo sudamericano de la primer
década del siglo veintiuno .

[WIKIPEDIA] Luego de varios meses, se realizó la primera asamblea extraordinaria,
el uno  de abril de mil novecientos ocho , fecha establecida como el día de su
fundación. En ese momento se decidió elegir el nombre del flamante club.

[WIKIPEDIA] Durante este período se crea el barrio de Boedo mediante la Ordenanza
número veintitres mil seiscientos noventa y ocho  del once  de junio de mil
novecientos sesenta y ocho .
```

Asimismo, inicialmente se realizaron tests de unidades particulares para validar las distintas ocurrencias de cada uno de los WFSTs específicos (naturales, reales, ordinales, romanos, fechas, horas, abreviaturas, etc.). A continuación se incluye una muestra de algunos de ellos:

0	1000000°	IX	U2
16	9887901°	X	UB40
123	1887701°	LI	etc.
4567	988.770°	XC	Bs.As.
98765	1.012	CCC	min.
988790	1.010	DCCXI	06/02/2002
1320461	1.0102	CMII	13/09/03
15340308	1.99	MM	16:30
100000	1.50	MI	4:10
999999999	+2,012	MMMCMXCIX	08:05
1°	0,25	dale	23:59:59
16°	+4,0102	ciclón	06:07:08
123°	-5,99	DALE	00:00:00
4567°	+6,50	CICLON	16:00
98765°	II	dc	4510-1100
988790°	VIII	AFJP	$2.99

7.2 Test de Selección de Pesos

Para realizar el test de selección de pesos se ejecutó un script implementado en Python que invoca los programas que construye el WFSTs con los pesos especificados en el script (buildFstTesis.exe), el que realiza las traducciones de texto de entrada (openFstTesisTest.exe) y el que establece las diferencias entre los resultados obtenidos y las referencias generadas por humanos (Diff.exe). Los detalles del proceso del desarrollo del test se encuentran descriptos en 4.4.5 Selección de pesos. En la Tabla 4.1 se describen los programas utilizados para el test.

Programa	**Descripción**
script.py	Ejecuta en secuencia los programas que se describen a continuación guardando los archivos resultantes de la ejecución para la combinación de pesos determinada.

buildFstTesis.exe	Construye el WFST completo (con la unión de los WFST parciales, p.ej. naturales, romanos, abreviaturas, etc.)
openFstTesisTest.exe	Realiza la traducción de un archivo de entrada con cien (100) frases de ejemplo y retorna un archivo con el resultado de la traducción usando el WFST construido en el paso anterior.
Diff.exe	Establece las diferencias archivo con el resultado de la traducción realizada en el paso anterior y un archivo de referencias con la traducción realizada por un humano. Esto permite comparar la efectividad de los resultados retornados por el sistema de la tesis. El objetivo es minimizar dichas diferencias.

Tabla 7.1 – Programas de Test de Selección de Pesos

7.2.1 Resultado seleccionado:

El resultado de combinación de pesos seleccionado para realizar la evaluación es la siguiente, porque tiene la menor cantidad de diferencias entre las frases de referencias (traducidas por humanos) y las frases traducidas por el sistema y que además las retorna en el menor tiempo posible con respecto a las otras pruebas corridas en PC con configuraciones similares.

- **p = 1**
- **pf = 10**
- **factor = 5000**

donde:

- p: Peso Interno = de los estados internos de los WFSTs
- pf: Peso Final = del estado final de cada WFSTs
- factor: Factor que permite distanciar la solución trivial (caracteres literales / iniciales) de las particulares (naturales, reales, romanos, etc.).
    - Solución Trivial: XX = XX
    - Particular (Romano natural): XX = veinte

Ver detalle de los resultados obtenidos en la Tabla 7.2.

P	Pf	Factor	CantDiff	Segundos	Minutos
1	1	5000	98	824,03	13,73
1	1	7500	98	796,18	13,27
1	1	10000	98	800,32	13,34
1	5	5000	95	617,58	10,29
1	5	7500	95	621,71	10,36
1	5	10000	95	586,75	9,78
1	10	5000	95	564,82	9,41

P	Pf	Factor	CantDiff	Segundos	Minutos
1	10	7500	95	577,69	9,63
1	10	10000	95	568,14	9,47
5	1	5000	105	373,27	6,22
5	1	7500	105	373,79	6,23
5	1	10000	105	374,38	6,24
5	5	5000	98	575,28	9,59
5	5	7500	98	607,08	10,12

P	Pf	Factor	CantDiff	Segundos	Minutos
5	5	10000	98	600,53	10,01
5	10	5000	95	600,07	10,00
5	10	7500	95	600,27	10,00
5	10	10000	95	601,29	10,02
10	1	5000	105	374,16	6,24
10	1	7500	105	375,14	6,25
10	1	10000	105	372,46	6,21
10	5	5000	105	376,05	6,27
10	5	7500	105	376,05	6,27
10	5	10000	105	372,17	6,20
10	10	5000	98	574,45	9,57
10	10	7500	98	563,48	9,39
10	10	10000	98	594,64	9,91
1	5	1000	230	619,01	10,32
1	5	2500	135	612,19	10,20
1	7,5	1000	230	594,83	9,91
1	7,5	2500	135	594,93	9,92
1	7,5	5000	95	564,54	9,41
1	10	1000	230	566,01	9,43
1	10	2500	135	573	9,55
3	5	1000	160	563,76	9,40
3	5	2500	130	563,21	9,39
3	5	5000	95	572,83	9,55
3	7,5	1000	221	569,58	9,49
3	7,5	2500	130	574,46	9,57
3	7,5	5000	95	605,188	10,09
3	10	1000	224	596,4	9,94
3	10	2500	130	606,8	10,11
3	10	5000	95	597,27	9,95
5	5	1000	160	600,77	10,01
5	5	2500	95	596,39	9,94
5	7,5	1000	160	596,73	9,95
5	7,5	2500	95	564,87	9,41
5	7,5	5000	98	569,4	9,49
5	10	1000	218	575,83	9,60
5	10	2500	130	565,48	9,42
5	10	12500	249	617,83	10,30
5	10	15000	95	574,631	9,58
5	10	17500	95	577,64	9,63
5	10	20000	110	568,87	9,48
5	10	40000	1132	604,23	10,07
5	12,5	10000	98	563,32	9,39
5	15	10000	98	563,83	9,40
7,5	10	10000	98	597,4	9,96
7,5	12,5	10000	98	563,62	9,39
7,5	15	10000	98	595,72	9,93
8	10	10000	98	569,19	9,49
8	12,5	10000	98	571,24	9,52
8	15	10000	98	595,8	9,93
9	10	10000	98	573,68	9,56
9	12,5	10000	98	563,72	9,40
9	15	10000	98	595,61	9,93
5	100	10000	2896	618,75	10,31

Tabla 7.2 – Resultados de Test de Selección de Pesos

7.2.2 Determinización

Se observó que se manifiesta un error de OpenFST cuando se intenta determinizar un WFST donde el peso interno de los WFST (p) es mayor al peso final (pf). Es decir que, al determinizar un WFST con p > pf el algoritmo de construcción del WFST completo, no termina.

Con lo cual solo para ejecutar los tests en estos casos se removió la determinación de los WFST para poder establecer las diferencias en dichos casos. Luego, cuando se eliminó la determinización se evitó el error que se produce cuando los pesos internos son mayores a los finales.

Cabe destacar que dada la combinación de pesos seleccionada (p = 1, pf = 10, factor = 5000) la determinación no fue removida para realizar la evaluación.

7.3 Herramienta de Comparación de archivos (Diff)

Se desarrolló una herramienta que permite cuantificar las diferencias entre los archivos para comparar los resultados obtenidos por el sistema para realizar los test y la evaluación. Esta

herramienta toma como entrada dos archivos de texto a comparar y como resultado devuelve un archivo de texto con las diferencias marcadas y con la cantidad total de diferencia encontradas. La misma se encuentra desarrollada en C++ standard y está basada en la biblioteca *diff* [Diff Code], proyecto de Google que provee algoritmos para realizar operaciones para archivos de texto plano.

La **línea de comando** para ejecutar el programa *Diff* (desde DOS) es la siguiente:

```
diff [archivo #1] [archivo #2] [archivo salida] [archivo #diferencias]
[formato de archivo]
```

donde los parámetros especificados son los siguientes:

- **Archivo #1**: Ubicación y nombre del primer archivo de texto para realizar la comparación.
- **Archivo #2**: Ubicación y nombre del segundo archivo de texto para realizar la comparación.
- **Archivo Salida**: Ubicación y nombre del archivo de salida que contiene el resultado de la comparación de los archivos #1 y #2, donde las diferencias se encuentran marcadas.
- **Archivo #diferencias**: Ubicación y nombre del archivo de salida que contiene la cantidad de diferencias encontradas entre de los archivos #1 y #2. Este archivo es utilizado para el test de selección de pesos ya que se permite ver rápidamente el número de las diferencias entre los resultados de la traducción del sistema y las referencias humanas.
- **Formato de archivos**: Codificación de caracteres de los archivos #1 y #2.
  - –u: Formato UTF8
  - –a: Formato ANSI

7.4 Detalle de diferencias encontradas en la evaluación

Durante la evaluación se encontraron diferencias entre las traducciones realizadas por el sistema y las referencias. Existen casos que no pudieron ser resueltos con tags debido a su naturaleza, o bien, porque el traductor no contemplaba la traducción esperada según las referencias. A continuación, se analizan algunos casos representativos, junto a soluciones propuestas para los mismos (para que trabajos futuros las tengan en cuenta).

7.4.1 Abreviaturas

Entrada	Referencias	Traducción realizada	Solución propuesta
R. P.	ruta provincial	reverendo P	Preprocesar la entrada para eliminar espacios en blanco y delimitar que se trata de una única abreviatura (R.P.).
<ABREV> R. P.</ABREV>	ruta provincial	reverendo padre	Análogo a la anterior.

Entrada	Referencias	Traducción realizada	Solución propuesta
<ABREV> a. C.</ABREV>	Antes de Cristo	área compañía	Análogo a la anterior
etc. (fin de frase)	etcétera.	etcétera	Se podría definir un tag que indique que la abreviatura coincide con un fin de frase para indicar al traductor que el punto debe incluirse también.
a.C. (fin de frase)	Antes de Cristo.	antes de Cristo	Análogo a la anterior
Radiofonía	Radiofonía	radiodifusión fonía	Se realizó la expansión de la palabra *Radio* por *radiofonía* debido a que se encuentra en la lista de abreviaturas provista por la RAE. En este caso, se debería delimitar el alcance de la expansión.
CD	ce de	cuatrocientos	Análisis de contexto ya que el traductor lo toma como un número romano. Se resuelve con tags.
C	Ce	Centésimo	Análisis de contexto ya que el traductor lo toma como abreviatura en lugar de sigla. Se resuelve con tags.
/ton	Por tonelada	/ton	Extender la lista de abreviaturas y tratar el símbolo / con un tag especial para que lo expanda como *por*.
disp.	Disposición	disp.	Extender la lista de abreviaturas
M2	Metros cuadrados	mdos	Extender la lista de abreviaturas (para singular y plural)
FFAA	Fuerzas Armadas	FFAA	En este caso la abreviatura no fue expandida debido a que no tenía el punto entre la F y la A. Para ello se debería extender el listado de abreviaturas para contemplar todos los casos con punto y sin punto para que se realicen las traducciones correspondientes para este tipo de abreviaturas (Ej: FFCC, EEUU, RRPP).

7.4.2 Fechas

Entrada	Referencias	Traducción realizada	Solución propuesta
1 de mayo	primero de mayo	Uno de mayo	Análisis de contexto y luego definir que el *uno* es *primero*. Se podría contemplar con tags especiales.

7.4.3 Grados

Entrada	Referencias	Traducción realizada	Solución propuesta
4.7° C	cuatro grados Celcius con siete décimas	cuatro punto siete grados centésimo	Extender el WFST de Grados y el alcance de su respectivo tag, para contemplar los grados Celsius y Fahrenheit y la correspondiente especificación de los decimales.

7.4.4 Horas

Entrada	Referencias	Traducción realizada	Solución propuesta
de 10 a 13 y 14:30 a 19:30	de diez a trece y *de* catorce treinta a diecinueve treinta	diez a trece y catorce horas y treinta minutos a diecinueve horas y treinta minutos	Análisis de contexto y luego definir para períodos la traducción de *y* = *y de*. Se podría contemplar con tags especiales.
22.30	veintidos y treinta	veintidos punto treinta	Análisis de contexto ya que el traductor lo toma como un número real. Se resuelve con tags.

7.4.5 Importes

Entrada	Referencias	Traducción realizada	Solución propuesta
3,82 pesos	tres pesos con ochenta y dos centavos	tres coma ochenta y dos pesos	Contemplar los decimales como centavos. Agregar la palabra *centavos* en la traducción.
USD 200	doscientos dólares	USD doscientos	Contemplar el símbolo *USD* para importes. Podría usarse un archivo de configuración.

Entrada	Referencias	Traducción realizada	Solución propuesta
$50 millones	cincuenta *millones de pesos*	cincuenta *pesos millones*	La traducción realizada por el sistema aplica para el símbolo *$* con *el número*, entonces la palabra *millones* no la traduce, ni cambia el orden como en las referencias. Se podría considerar incluir este caso en el WFST de importes.
$1000000	un millón *de* pesos	un millón pesos	Este caso se debería tratar especialmente ya que cuando se trata *de* un millón se agrega la palabra de antes de la palabra pesos. Se debería actualizar el WFST de importes para contemplar este caso.

7.4.6 Naturales

Entrada	Referencias	Traducción realizada	Solución propuesta
(1872-1970)	(mil ochocientos setenta y dos-mil novecientos setenta)	(uno ocho siete dos uno nueve siete cero)	Análisis de contexto, ya que el traductor lo toma como teléfono en lugar de tomarlo como período. Se resuelve con tags, pero se podría contemplar con tags especiales que indiquen periodos para también traducir el "-".
2011/2012	dos mil once / dos mil doce	doscientos uno un medio cero doce	Análisis de contexto, ya que el traductor lo toma *1/2* como *un medio* en lugar del último y primer digito de los números antes y después de la barra. Se resuelve con tags.
26,1 puntos	veintiseis coma un puntos	veintiseis coma uno puntos	Incluir en el WFST de naturales el tratamiento del caso (*un*), ya que no esta contemplado. Luego realizar el análisis de contexto para indicarle que sería *un* en lugar de *uno* según la palabra que le sigue.

Entrada	Referencias	Traducción realizada	Solución propuesta
450 personas	cuatrocientas cincuenta personas	cuatrocientos cincuenta personas	Incluir en el WFST de naturales el tratamiento del caso (**cientas,* donde * indica la expansión del número correspondiente), ya que no esta contemplado. Luego realizar el análisis de contexto para indicarle que sería *cuatrocientas* en lugar de *cuatrocientos* según la palabra que le sigue.

7.4.7 Reales

Entrada	Referencias	Traducción realizada	Solución propuesta
1.5 km.	un kilómetro y medio	uno punto cinco kilómetros	Si bien la traducción realizada no sería incorrecta, se puede tratar este caso como fracción. Extender las traducciones posibles de fracciones como este caso. Luego usar un tag de fracción para que se realice la traducción correcta según el contexto.
0,414	cero coma cuatrocientos catorce	cero coma cuatro uno cuatro	Si bien la traducción realizada no sería incorrecta, se podría tratar este caso indicando que se debería traducir como centenas (como las decenas).

7.4.8 Símbolos

Entrada	Referencias	Traducción realizada	Solución propuesta
Años '60	años sesenta	años 'sesenta	En este caso no haría falta corregir esto debido a que las siguientes fases del proceso del sistema TTS no considerarían este símbolo para asociar un sonido.
4x4	cuatro por cuatro	Cuatro xcuatro	Definir la *x* como abreviatura especial (*x* = *por*). Luego indicar con el tag de abreviatura de acuerdo al contexto cuando deba realizar esta traducción.

Entrada	Referencias	Traducción realizada	Solución propuesta
D'Alambert	D'Alambert	DAlambert	El carácter que se encuentra a continuación de la *D* no fue reconocido por el traductor. Se podrían contemplar los caracteres especiales en un archivo de configuración que sea extensible para permitir al traductor el reconocimiento de los mismos.
Knoedler & Company	Knoedler & Company	Knoedler y Company	El carácter especial *&* debería ser traducido como *y* o bien como *and*, y las referencias deberían indicar una de estas dos ocurrencias. Se podría indicar con un tag cual sería la expansión correcta para este carácter. También se podría incluir en el archivo de abreviaturas especiales.
Antonio R. Deane	Antonio erre Deane	Antonio erre . Deane	El punto que sigue a la inicial del segundo nombre no se debería traducir. En este caso se podría definir con un tag para que se traduzca el punto como blanco según el contexto.
Hashtag #twitter	hashtag #twitter	hashtag numeral twitter	Este es un caso donde en las referencias el símbolo no fue traducido, y la traducción del sistema es correcta. Actualmente el # es un símbolo muy usado desde que Twitter tomó protagonismo, pero el lector en este caso omite decirlo en voz alta. Si bien, la traducción del sistema es correcta, se podría incluir un tag para decidir si se debe traducir o no.
Resoluciones 54/175	resoluciones cincuenta y cuatro *barra* ciento setenta y cinco	resoluciones cincuenta y cuatro / ciento setenta y cinco	Definir las traducciones posibles de este símbolo según el contexto. Si debe traducir o no. Luego se debería desambiguar con un tag especial.

8 Apéndice C – Glosario

Término / Abreviatura	**Descripción**
Abreviatura	[GRA 10] Letra o conjunto de letras usadas para representar de forma breve una palabra o una frase. • cap., pag., Cia., km, srta.
Acrónimo	[RAE Acr] 1. m. Tipo de sigla que se pronuncia como una palabra; p. ej., o(bjeto) v(olador) n(o) i(dentificado). 2. m. Vocablo formado por la unión de elementos de dos o más palabras, constituido por el principio de la primera y el final de la última, p. ej., ofi(cina infor)mática, o, frecuentemente, por otras combinaciones, p. ej., so(und) n(avigation) a(nd) r(anging), Ban(co) es(pañol) (de) (crédi)to. [GRA 10] Palabra formada por las letras iniciales de una expresión compuesta, pero que suele ajustarse a las reglas fonológicas de la lengua. • sida, radar, ovni, láser.
FST	*Finite-State Transducers* (traductores de estados finitos)
OpenFST	Librería desarrollada en C++ para la construcción, combinación, optimización y búsqueda en los WFSTs
PLN	*Procesamiento del Lenguaje Natural* (área de investigación del procesamiento del lenguaje)
Sigla	[RAE Sig] 1. f. Palabra formada por el conjunto de letras iniciales de una expresión compleja; p. ej., O(rganización de) N(aciones) U(nidas), o(bjeto) v(olante) n(o) i(dentificado), Í(ndice de) P(recios al) C(onsumo). 2. f. Cada una de las letras de una sigla (palabra formada por letras iniciales). P. ej., N, O y U son siglas en ONU. 3. f. Cualquier signo que sirve para ahorrar letras o espacio en la escritura. [GRA 10]

Término / Abreviatura	**Descripción**
	Palabra formada por las letras iniciales de una expresión compuesta. • ONU, IVA, DGI, IBM, AFIP.
TTS	*Text To Speach* (traducción del texto a su equivalente oral)
WFST	*Weighted Finite-State Transducers* (traductores de estados finitos con peso)
Wordnets	[WIK WordNet] Base de datos léxica del Idioma Inglés. Agrupa palabras en inglés en conjuntos de sinónimos llamados synsets, proporcionando definiciones cortas y generales, y almacena las relaciones semánticas entre los conjuntos de sinónimos.

9 Apéndice D - Bibliografía

[ALL 07] **"OpenFst: A General and Efficient Weighted Finite-State Transducer Library"**. Extended Abstract of an Invited Talk. Autores: *Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri.* (2007)

[BON 00] **"Actividades en el Área de Conversión de Texto a Habla en el Centro TALP"**. Centro de Investigación TALP Universitat Politècnica de Catalunya, UPC. Autores: *A. Bonafonte, A. Febrer, A. Moreno, J.A. Rodríguez Fonollosa, A. Sesma* (Nov-2000)

[COS 09] **"Scale-invariant transition probabilities in free word association trajectories".**. – Journal of Frontiers in Integrative Neurosciences. Vol 3 - Autores: *Martín Elias Costa, Flavia Bonomo, Mariano Sigman* (2009)

[Diff Code] **"google-diff-match-patch".** Diff, Match and Patch libraries for Plain Text. *Google project*.
http://code.google.com/p/google-diff-match-patch/

[EAG 99] **"The EAGLES 7-step recipe".** EAGLES Evaluarion Workgroup.
http://www.issco.unige.ch/en/research/projects/eagles/ewg99/7steps.html

[EDO 00] **"OpenFst For Windows Getting Started"**
http://www.edobashira.com/2009/03/openfst-for-windows-getting-started.html

[EST 10] **"Evaluación de sistemas de Procesamiento de Lenguaje Natural"**. Escuela de Ciencias Informáticas (ECI) 2010. Departamento de computación. Facultad de Ciencias Exactas y Naturales. Universidad de Buenos Aires (UBA) –Autor: *Paula Estrella,* FaMAF, UNC (2010)
http://www.famaf.unc.edu.ar/~pestrella/elic-2010/elic2010-estrella.pdf

[FEMTI] **"Framework for the Evaluation of Machine Translation"** (FEMTI). International Standards for Language Engineering (ISLE)
http://www.issco.unige.ch/en/research/projects/isle/femti/framed-glossary.html

[FES 01] **"The Festival Speech Synthesis System"**

http://www.cstr.ed.ac.uk/projects/festival/

[FreeLing] **"FreeLing Home Page"**. An Open Source Suite of Language Analyzers.

http://nlp.lsi.upc.edu/freeling/

[FST 00] **"OpenFST Library"**. http://www.openfst.org/

[FST 01] **"OpenFst: An Open-Source, Weighted Finite-State Transducer Library and its Applications to Speech and Language"**. Introduction. Autores: *Cyril Allauzen, Martin Jansche, Michael Riley* (2009)

[FST 02] **"OpenFst: An Open-Source, Weighted Finite-State Transducer Library and its Applications to Speech and Language"**. Part I. Theory and Algorithms. - Autores: *Cyril Allauzen, Martin Jansche, Michael Riley* (2009)

[FST 03] **"OpenFST Library – Available Operations"**

http://www.openfst.org/twiki/bin/view/FST/FstQuickTour#AvailableOperations

[FST 04] **"OpenFst Download"**

http://www.openfst.org/twiki/bin/view/FST/FstDownload

http://www.openfst.org/twiki/bin/view/Contrib/OpenFstWin

[FST 05] **"OpenFst Weights"**

http://www.openfst.org/twiki/bin/view/FST/FstQuickTour#FST_Weights

[GRA 00] **"Graphviz - Graph Visualization Software"**. http://www.graphviz.org/.

[GRA 10] "**Front-End de Sistemas TTS**". Introduccion a las Tecnologias del Habla 2o cuatrimestre 2010. Departamento de computación. Facultad de Ciencias Exactas y Naturales. Universidad de Buenos Aires (UBA). Autor: *Agustin Gravano* (2011)

[GOD 12] "**Procesamiento de Consultas, Feedback de Relevancia y Evaluación**". Análisis y Recuperación de Información. Autor: *Dra. Daniela Godoy* (2012)

http://www.exa.unicen.edu.ar/catedras/ayrdatos/slides/procesamiento-6p.pdf

[JIM 99] **"Adaptación y mejora de un sistema de preprocesamiento y categorización gramatical".** Pag.49-104 – Proyecto de Fin de Carrera - Universidad Politécnica de Madrid. Autora*: Azucena Jiménez Pozo* (1-12-1999)

[JUR 09] **"Speech and Language Processing"**. Autores*: - Daniel Jurafsky and James H.Martin*. 2nd edition. Prentice-Hall (2009)

[MAN 01] **"How Text-to-Speech Works".** Autor: *Aadil Mansoor* (2001)
http://project.uet.itgo.com/textto1.htm

[MAN 09] **"Chapter 8: Evaluation in information retrieval"**. An Introduction to Information Retrieval. Autores: *Christopher D. Manning*, *Prabhakar Raghavan* y *Hinrich Schütze* (2009)

[MIL 01] **"Evaluation of Machine Translation Output for an Unknown Source Language: Report of an ISLE-Based Investigation".** Autores: *Keith J. Miller* (The MITRE Corporation,

Washington D.C., USA), *Donna M. Gates* (Carnegie Mellon University, Pittsburgh, USA), *Nancy Underwood* (CST, Denmark) and *Josemina Magdalen* (The Hebrew University of Jerusalem). MT Summit VIII, Santiago di Compostela, Spain (September 2001)

[MON 03] **"Estrategias para la mejora de la naturalidad y la incorporación de variedad emocional a la conversión texto a voz en castellano".** Tesis Doctoral. – Universidad Politécnica de Madrid. Autor: *Juan Manuel Montero Martínez* (2003)

[RAE Abr] **"Lista de abreviaturas"**. Real Academia Española (RAE).
http://buscon.rae.es/dpdI/apendices/apendice2.html

[RAE Acr] **"Acrónimo".** Real Academia Española (RAE).
http://lema.rae.es/drae/?val=acrónimo

[RAE Ord] **"Ordinales"**. Real Academia Española (RAE).
http://lema.rae.es/dpd/?key=ordinales

[RAE Sig] **"Sigla".** Real Academia Española (RAE).
http://lema.rae.es/drae/?val=sigla

[SPR 08] **"Linguistic Processing for Speech Synthesis"** Autor: *R.Sproat* (2008)

[VIT Ord] **"Números Ordinales".** Artículo Vitutor.
http://www.vitutor.net/1/a_0.html

[WIK 01] **"Sintesis del habla".** Artículo Wikipedia.
http://es.wikipedia.org/wiki/Sintetización_del_habla

[WIK Acr] **"Acrónimo".** Artículo Wikipedia.
http://es.wikipedia.org/wiki/Acrónimo

[WIK ASC] **"ASCII".** Artículo Wikipedia.
http://en.wikipedia.org/wiki/ASCII

[WIK Ord] **"Número Ordinal".** Artículo Wikipedia.
http://es.wikipedia.org/wiki/Número_ordinal

[WIK WordNet] **"WordNet".** Artículo Wikipedia.
http://es.wikipedia.org/wiki/WordNet