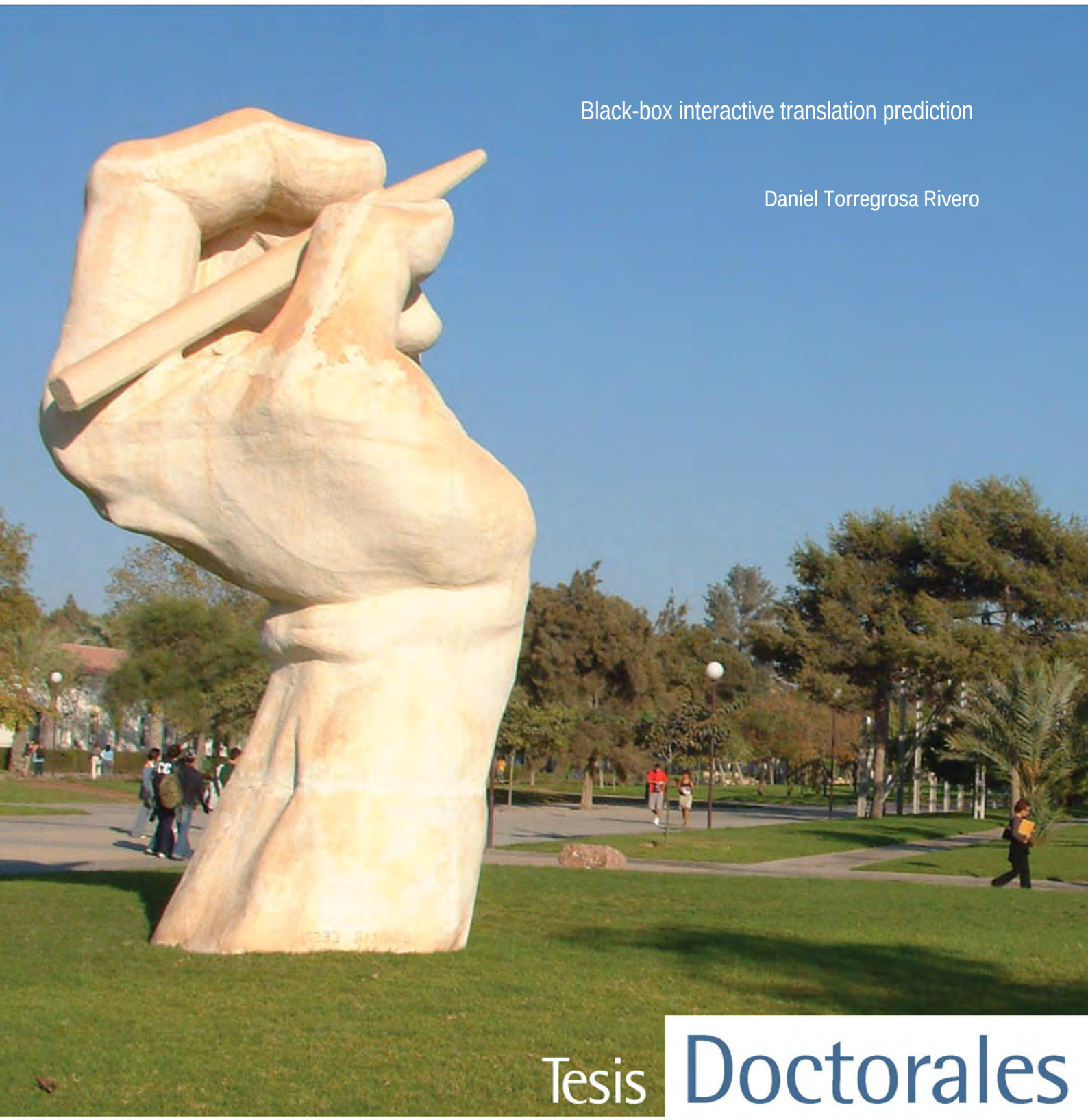
Universitat d'Alacant
Universidad de Alicante

Black-box interactive translation prediction

Daniel Torregrosa Rivero



Tesis Doctorales

www.eltallerdigital.com

UNIVERSIDAD de ALICANTE

Universitat d'Alacant
Universidad de Alicante
Departamento de Lenguages y Sistemas Informáticos
Escuela Politécnica Superior

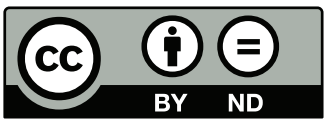
Black-box interactive translation prediction

Daniel Torregrosa Rivero

Tesis presentada para aspirar al grado de
DOCTOR/DOCTORA POR LA UNIVERSIDAD DE ALICANTE
MENCIÓN DE DOCTOR/DOCTORA INTERNACIONAL
DOCTORADO EN INFORMÁTICA

Dirigida por
Dr. Juan Antonio Pérez Ortiz
Dr. Mikel L. Forcada Zubizarreta

El autor de la tesis fue beneficiario de la ayuda ACIF/2014/365 de la Generalitat Valenciana. Parte del trabajo realizado ha sido financianciado por el Ministerio de Economía y Competitividad del Gobierno de España a través del projecto AyuTra (TIN2012-32615), el Gobierno Español a través del proyecto EFFORTUNE (TIN2015-69632-R), y el Gobierno de la República de Kazajistán.



Agradecimientos

En primer lugar, me gustaría agradecer a mis directores de tesis, Juan Antonio Pérez Ortiz y Mikel L. Forcada el esfuerzo que les ha supuesto llevar esta tesis a buen puerto. Ambos han contribuido significativamente al desarrollo de la tesis desde antes de que yo empezara a investigar: Mikel con la idea inicial de la que parte toda la tesis, y Juan Antonio con el primer prototipo mostrando su utilidad. Sin su dirección, la sequía de resultados que se produjo entre 2014 y 2015 hubiera supuesto el abandono por mi parte.

También me gustaría agradecer la parte que ha tomado el resto del grupo de investigación Transducens en mi formación. La ética de trabajo y los ideales que sigue el grupo y todos sus componentes, y que se reflejan en las líneas de investigación que se siguen y proyectos que se desarrollan, fueron factores determinantes que me llevaron a interesarme por la investigación en general, y en las tecnologías de la traducción en particular. Pese a que el desarrollo de mi tesis no permitió colaborar con el resto de miembros del grupo, Felipe Sánchez Martínez y Rafa Carrasco ejercieron una influencia positiva en mi forma de afrontar la investigación. Asimismo, agradecer el buen ambiente de trabajo que hay en el laboratorio del grupo, fomentado por los miembros del grupo que se reunen habitualmente en él; especialmente Miquel Esplà i Gomis y Víctor Sánchez Cartagena, que defendieron sus tesis mientras yo desarrollaba la mia.

Igualmente me gustaría agradecer a la Fondazione Bruno Kessler por darme la oportunidad de trabajar en el grupo HLT, a través del programa de estancias en verano, a los investigadores que, como yo, estaban desarrollando su tesis o disfrutando de una estancia, y, especialmente, a Marco Turchi y Matteo Negri, quienes dirigieron el trabajo que desarrollé durante los tres meses de estancia.

Asimismo, me gustaría agradecer el apoyo recibido por mi familia, especialmente el de mi madre, que vio comenzar mis estudios de doctorado pero no los verá terminar. También agradezco el apoyo de mis amigos, especialmente aquellos que emigraron, y aquellos que decidieron, como yo, completar su formación cursando un programa de doctorado.

Finalmente, me gustaría expresar mi agradecimiento a los autores y grupos de desarrollo de las diversas herramientas liberadas bajo una licencia de software libre o código abierto, ya que gran parte de esta tesis no hubiera sido posible sin sus contribuciones; en especial, a las comunidades de desarrolladores de Apertium, Moses, Thot, y OmegaT.

Apoyo institucional

El autor de la tesis fue beneficiario de la ayuda ACIF/2014/365 de la Generalitat Valenciana. Parte del trabajo realizado ha sido financiado por el Ministerio de Economía y

Competitividad del Gobierno de España a través del proyecto AyuTra (TIN2012-32615), el Ministerio de Economía y Competitividad del Gobierno de España a través del proyecto EFFORTUNE (TIN2015-69632-R), y el Gobierno de la República de Kazajistán.

Daniel Torregrosa Rivero
Alicante, 30 de enero de 2018

Resumen de la tesis

En un mundo globalizado como el actual en el que, además, muchas sociedades son inherentemente multilingües, la traducción e interpretación entre diversas lenguas requiere de un esfuerzo notable debido a su volumen. Por ejemplo, la Unión Europea defiende activamente el plurilingüismo y el derecho de cada ciudadano a comunicarse en el idioma idioma oficial de su estado, presentándolo como un valor para el diálogo, la cohesión y la prosperidad, pese a los costes que genera: la Unión gasta mil millones de euros por año para mantener las políticas de plurilinguismo (Rehm and Uszkoreit, 2013), mantiene la entidad de traducción más grande del mundo (Dirección General de Traducción de la Unión Europea) que emplea, al menos, a 2 000 traductores (DGT annual activity report, 2015).

Podemos encontrar diversas aproximaciones para afrontar la tarea de traducción, y elegir, dependiendo del tipo de tarea y los recursos disponibles, cuál es la más adecuada. Por ejemplo, una aproximación es el uso de un sistema de traducción automática (TA), que es capaz de traducir automáticamente un texto entre idiomas sin necesidad de interacción humana alguna. La TA puede ser usada para hacernos una idea del contenido de un texto escrito en una lengua que no entendemos; a este uso se le denomina *asimilación*. Muchos de los sistemas actuales de TA pueden servir para este cometido, siempre y cuando la traducción preserve el significado del texto original. En cambio, cuando queremos publicar el texto en el idioma al que ha sido traducido, es usualmente necesario repasar y corregir la traducción generada por el sistema; a este uso se le conoce como *diseminación*. En este escenario, incluso los sistemas de TA de mayor calidad[1] pueden producir traducciones que parecen forzadas o que contienen errores, y se hace necesaria la *postedición* de la salida del sistema de TA, un tipo de *traducción asistida por ordenador* (TAO) en el que un traductor humano especializado subsana los errores, imprecisiones, y otros tipos de problemas que aparecen en las traducciones generadas por el sistema de TA. Se ha demostrado[2] que la postedición mejora la productividad de los traductores sin perjudicar la calidad de las traducciones generadas en un entorno comercial.

[1] Se considera que, para un uso concreto, una traducción de calidad es aquella que cumple con su cometido. El concepto de calidad en la traducción es un campo de investigación muy activo en el momento, y su definición con mayor detalle no es el objetivo principal de esta memoria.

[2] Por ejemplo, en los estudios realizados por TAUS (`https://www.taus.net/think-tank/best-practices/postedit-best-practices/machine-translation-post-editing-guidelines`), Memsource (`https://www.memsource.com/blog/2015/08/18/post-editing-in-memsource/`) y Autodesk (`http://langtech.autodesk.com/productivity.html`).

Sin embargo, en muchas ocasiones la calidad de la salida de los sistemas de TA es reducida y el tiempo invertido en posteditar la traducción generada es mayor que el de realizar la traducción desde cero, o puede que no exista un sistema de TA para los idiomas entre los que queremos traducir. En estos escenarios en los que la TA no está disponible, se pueden aplicar otros tipos de TAO. Un tipo usual de TAO son las memorias de traducción, conjuntos de frases ya traducidas previamente; para cada frase que va a ser traducida, el sistema busca si una frase similar ya ha sido previamente traducida por humanos, ofreciendo la traducción correspondiente como sugerencia.

Pero las memorias de traducción solo se pueden aplicar si frases similares han sido previamente traducidas por humanos. En caso de no disponer de trabajos de traducción similares ya realizadas, existe otra forma de TAO, que es en la que se centra esta memoria: la traducción automática interactiva (TAI). En lugar de apoyar al usuario usando frases ya traducidas, un sistema de TAI va generando sugerencias *al vuelo*[3] mediante el uso de TA. La granularidad de las sugerencias puede ir desde una sola palabra hasta una propuesta que complete la traducción de la frase actual.

En el estado de la cuestión, descrito en el Capítulo 1 de la memoria, la TAI se apoya en un sistema de traducción automática basada en corpus, principalmente sistemas de traducción automática estadística (TAE): las sugerencias se obtienen consultando no sólo las traducciones generadas por el sistema de TAE, sino también otros componentes internos usados para generar las traducciones. De esta forma, el sistema de TAI es capaz de acceder directamente a una gran parte de la información contenida en el modelo de TAE. Pero esta forma de acceso hace que el sistema de TAI no sea útil bajo ciertas circunstancias. Por ejemplo, si no hay suficiente corpus paralelo para entrenar un modelo de TAE que genere traducciones con suficiente calidad: generar el corpus paralelo necesario para alcanzar una calidad aceptable puede tener un coste prohibitivo; pero también es posible que exista un sistema de TAE de suficiente calidad para el par de lenguas en el que vamos a trabajar, pero no podamos acceder a los componentes internos, bien porque el software es cerrado y no podemos modificarlo para acceder a los componentes, bien porque es un modelo al que no tenemos acceso directo.

Recientemente, ha aparecido una modalidad que utiliza traducción automática neuronal para generar las sugerencias; en este caso, las modificaciones necesarias para adaptar el modelo de traducción automática neuronal para TAI son mínimas: el modelo de traducción neural predice unidad a unidad[4] dependiendo exclusivamente de la frase origen y del prefijo de la traducción que ya se ha generado; es decir, se puede usar el prefijo escrito por el usuario para obtener la siguiente unidad. Este tipo de sistema sigue estando sujeto a la mismas restricciones relacionadas con la necesidad de disponer de un corpus bilingüe de suficiente tamaño.

[3]Es decir, las sugerencias se generan conforme el usuario va escribiendo la traducción. Las sugerencias dependen directamente de la frase origen y el prefijo de la frase meta tecleado.

[4]Por ejemplo, una unidad, aunque algunos sistemas de traducción neuronales (usando una forma de representación de datos conocida como *byte pair encoding*, codificacion por pares de bytes) trabajan con subsecuencias de letras.

Ejemplo 1

Frase en lengua origen	this	studio	is	spacious
1-gramas	este	estudio	es	espacioso
2-gramas	este estudio			
		estudio es		
			es amplio	
3-gramas	este estudio está			
		estudio es amplio		

Traducción de todos los n-gramas hasta $n = 3$ de la frase en inglés *This studio is spacious* al español, usando Google Translate (en diciembre de 2016). El sistema de TA estadístico hace una selección léxica distinta cuando traduce *spacious* como una palabra suelta (generando la traducción *espacioso*), escogiendo en su lugar el sinónimo *amplio* cuando se traduce con otras palabras. Las frases están en minúsculas para simplificar el ejemplo.

En cambio, la aproximación a la TAI desarrollada en esta memoria de tesis funciona de una forma distinta: puede utilizar cualquier recurso bilingüe capaz de generar una o más traducciones para una secuencia de palabras, pudiendo utilizar no solo sistemas de traducción automática basada en corpus, sino también otros tipos de recursos, como sistemas de traducción basados en reglas, diccionarios, glosarios, etc. Podemos considerar que esta aproximación sigue un esquema de *caja negra*, es decir, que solo necesita consultar la traducción generada por los recursos subyacentes, mientras que las aproximaciones anteriores siguen un esquema de *caja de cristal*, es decir, que no solo consultan las traducción, sino también algunos de los diversos componentes internos necesarios para generarla. Debido a que no se accede a los componentes internos de los recursos bilingües, se utiliza una estrategia distinta para extraer la mayor cantidad de información posible de los mismos: todos los n-gramas (secuencias de n palabras) de la frase hasta cierto valor máximo de n, incluyendo solapamientos, son traducidos usando uno o más recursos bilingües; las traducciones serán utilizadas como sugerencias de TAI. El Ejemplo 1 ilustra la generación de sugerencias.

Esta aproximación tiene una limitación: debido a la forma en la que se generan las sugerencias, no es posible generar un alineamiento entre la frase en lengua origen y la frase que está siendo escrita; es decir, a diferencia de los modelos de caja de cristal, que fuerzan el modelo subyacente para generar la traducción más probable partiendo del prefijo escrito por el traductor, en este modelo de caja negra no es posible saber qué parte de la frase origen ha sido traducida. Utilizar un criterio sencillo, como puede ser ofrecer todas las sugerencias que tengan como prefijo la última palabra escrita por el usuario, es inviable, ya que no se espera que el traductor tolere un número elevado de sugerencias; aunque se ahorre esfuerzo, la TAI no será útil si se tarda más tiempo en leer las sugerencias que en teclear la traducción correspondiente. Por ello, es necesario desarrollar técnicas para elegir qué sugerencias han de ser mostradas al usuario. Dos formas diferentes de ofrecer sugerencias han sido desarrolladas

Ejemplo 2

Frase origen	“this studio is spacious”
Referencia	“este estudio es amplio”
Prefijo	e
Sugerencias	este estudio está **este** estudio es amplio estudio
Prefijo	e**ste_**
Sugerencias	
Prefijo	e*ste* e
Sugerencias	**estudio es amplio** estudio es amplio es
Traducción generada	e**ste** e**studio es amplio**

Ejemplo de una sesión de TAI traduciendo de ingles a español, ofreciendo un máximo de 4 sugerencias, y con las sugerencias del Ejemplo 1. Las sugerencias en negrita son las que fueron más útiles para realizar la traducción; las partes en negrita del prefijo y la traducción generada son las que fueron insertadas usando una sugerencia. En el ejemplo, tras teclear *e*, se ofrecen cuatro sugerencias, y la segunda se utiliza; después, se teclea otra letra (*e*), se ofrecen cuatro sugerencias, y se utiliza la primera, finalizando la traducción. Para facilitar la visualización, se ha sustituido un espacio al final del prefijo por un guión bajo.

en esta tesis: un modelo heurístico y un modelo basado en aprendizaje computacional. El Ejemplo 2 muestra una posible sesión de TAI.

Modelo heurístico

Como una primera aproximación para elegir qué sugerencias se le ofrecen al usuario, se planteó un modelo heurístico basado en observaciones empíricas, que se analiza en el Capítulo 2. Inicialmente, se enunciaron cuatro hipótesis basadas en el comportamiento observado:

- Las sugerencias cortas ahorran menos trabajo, pero es probable que sean útiles; por ejemplo, determinantes o palabras funcionales.

- Las sugerencias largas ahorran más trabajo, pero es menos probable que sean útiles; por ejemplo, una sugerencia puede contener un pequeño error de concordancia y no poder ser usada.

- La traducción en lenguas cercanas es principalmente monótona con desviaciones y la *i*-ésima palabra de la frase en lengua meta es la traducción de la *i*-ésima palabra de la frase en lengua origen, o una cercana.[5]

- Las sugerencias generadas cerca de la parte de la frase que está siendo traducida son, probablemente, más útiles que las que se generan lejos de esta posición; es decir, si somos capaces de predecir correctamente qué parte de la frase origen está siendo traducida, podremos ofrecer sugerencias que son traducción del texto origen.

Por ello, se desarrolló la siguiente forma de seleccionar el orden en el que se ofrecen las sugerencias: se ofrecen la sugerencia más corta y la más larga que toman origen en la posición más cercana a la que se está actualmente traduciendo,[6] seguida de la más corta y la más larga de la segunda posición más cercana; cuando no quedan más posiciones origen, se continua con la segunda más corta y la segunda más larga de la posición más cercana, hasta un máximo de M sugerencias. Las sugerencias ofrecidas son siempre compatibles con el prefijo escrito, es decir, las sugerencias tienen como prefijo la palabra[7] que está siendo tecleada al final de la frase meta; no se ofrecen sugerencias si no se ha escrito al menos un caracter de la palabra, es decir, no se ofrecen sugerencias al principio del segmento y tras un espacio.

Asimismo, se desarrolló un modelo de evaluación automática basado en los usados en el estado de la cuestión: el sistema emula un traductor que planea una traducción[8] y teclea exactamente esta traducción; a cada pulsación de tecla, evalúa todas las sugerencias ofrecidas y elige la más larga compatible con la traducción planeada. El sistema no considera como compatibles sugerencias que no completan las palabras, es decir, no aceptará la sugerencia *la* si la palabra que va a escribir es *laringe*. La medida principal usada es la ratio entre las pulsaciones de teclas necesarias para escribir la traducción y la longitud final de la traducción en caracteres (KSR, *keystroke ratio*), considerando que aceptar una sugerencia cuesta una pulsación de tecla, independientemente de la posición en la lista; es decir, para el sistema automático, que no se equivoca al teclear ni replantea la traducción, tendrá un KSR máximo de 1 (valor que ocurre cuando ninguna sugerencia es compatible).

La aproximación heurística a la TAI de caja negra se probó en diversos escenarios. En el primero de ellos, se usó para traducir entre inglés y español y catalán y español usando el sistema de TA basado en reglas Apertium (Forcada et al., 2011), usando la evaluación automática antes descrita. En el Capítulo 2.4.3 se demuestra que se puede ahorrar alrededor del 60 % de las pulsaciones de teclas en el escenario de traducción entre catalán y español, ya que son lenguas muy emparentadas y la calidad de las traducciones generadas por el sistema de TA es alta. En cambio, solo se puede ahorrar alrededor del 20 % de pulsaciones de teclas

[5] Pese a que puede parecer un criterio anti intuitivo, ya que las lenguas suelen tener longitudes de frase distintas, se han observado resultados positivos para pares de lenguas distantes como inglés y árabe o inglés y chino simplificado en el Capítulo 3.

[6] Asumiendo que la *i*-ésima palabra de la frase en lengua meta se corresponde con la *i*-ésima palabra de la frase en lengua origen.

[7] Probablemente, un prefijo de la palabra, ya que las sugerencias a mostrar se evalúan a cada pulsación de tecla.

[8] El sistema automático recibe una traducción referencia.

para el escenario de traducción entre inglés y español; la calidad del sistema de TA para este par de lenguas y el dominio usado es baja, y pocas de las sugerencias pudieron ser usadas para completar las traducciones. Los resultados de estos experimentos han sido publicados (Pérez-Ortiz et al., 2014).

En el segundo escenario, se usó TAI para traducir entre inglés y español e inglés y checo, usando el sistema de TA estadístico Moses (Koehn et al., 2007) y el sistema de evaluación automática descrito previamente. Se usaron tres sistemas distintos para cada par de lenguas: uno entrenado con corpus de fuera del dominio, otro entrenado con corpus de fuera del dominio pero afinado usando datos de dentro del dominio, y otro entrenado completamente con datos de dentro del dominio. Independientemente del modelo de traducción, se ahorraron alrededor del 40 % de pulsaciones de teclas para la traducción entre inglés y español, y alrededor del 30 % de pulsaciones de teclas para la traducción entre inglés y checo. Los resultados de estos experimentos han sido publicados (Torregrosa et al., 2014).

Finalmente, se hizo una prueba con traductores no profesionales que tradujeron de catalán a español usando el sistema de TA basado en reglas Apertium. En esta prueba, los traductores tenían que traducir cinco frases sin asistencia y cinco con asistencia. Los traductores se ahorraron una media del 50 % de pulsaciones de teclas al recibir asistencia.[9] Al utilizar el sistema de evaluación automática previamente descrito usando como referencia las traducciones que generaron los usuarios, se mostró que se hubieran podido ahorrar hasta el 80 % de pulsaciones si los usuarios no hubieran cometido ningún error y hubieran aceptado todas las sugerencias compatibles. Al finalizar la prueba, los usuarios debían responder un pequeño formulario; todos ellos expresaron que la herramienta era cómoda de usar y les había ayudado a traducir más rápido y con menos esfuerzo. Los resultados de estos experimentos han sido publicados (Pérez-Ortiz et al., 2014).

Modelo basado en aprendizaje computacional

Pese a que el modelo heurístico obtiene resultados positivos, permitiendo ahorrar esfuerzo a los traductores, tiene un problema: no está basado en ningún principio sólido, sino en observaciones empíricas, y, aunque los patrones observados permitieron definir una heurística con buenos resultados, es difícil extraer patrones más complejos que mejoren el rendimiento del sistema. Por ese motivo, se ha desarrollado una segunda aproximación usa técnicas de aprendizaje computacional, en concreto, una red neuronal no recurrente. Esta aproximación no solo obtiene mejores resultados que la heurística, sino que, además, está basada en principios más rigurosos, ya que extrae automáticamente los patrones presentes en un conjunto de características (también llamadas *features*). Esta aproximación se analiza en el Capítulo 3.

[9] Comparado con el KSR sin asistencia, que, en todos los casos, es superior a 1, ya que los humanos cometen errores al teclear o replantean partes de la traducción, lo que acarrea pulsar más teclas de las que finalmente tendrá la traducción.

Las características son propiedades que se pueden medir y representan una faceta de un evento o instancia;[10] las características se usan para predecir la clase (en un proceso llamado clasificación) o un valor en un espacio continuo (en un proceso llamado regresión) de un evento o instancia. Se han definido 30 características que representan cada sugerencia:

- Un conjunto de 25 características que contienen información sobre la posición de la sugerencia en la frase origen, la posición en la que ha sido usada en la frase meta, la longitud del segmento en lengua origen y de su traducción, y la relación de éstos con la distribución de valores para las sugerencias que son aceptadas durante el entrenamiento.

- Una característica que codifica la letra inicial de la sugerencia: en los experimentos previos se detectó que determinadas letras o combinaciones de letras son más comunes que otras al inicio de palabra: por ejemplo, en español, la combinación de letras *de* es muy común a principio de palabra, ya que la preposición *de* aparece frecuentemente; pero hay muy pocas palabras que empiecen con la letra *x*. De esta forma, la red puede aprender a ser más permisiva con ciertos prefijos, y más estricta con otros.

- Una característica que codifica si la palabra anterior a la que está siendo escrita ha sido insertada al aceptar una sugerencia (independientemente de la longitud de la sugerencia) o no. Es posible que un usuario que ha aceptado una sugerencia acepte otra inmediatamente después.

- Una característica que representa la relación del intervalo de la frase origen de la sugerencia actual (p procede del intervalo $[b_p, e_p]$ de la frase origen) con la última sugerencia aceptada por el usuario (p'), con cinco casos diferentes: si p termina antes de empezar p' ($e_p + 1 < b'_p$), si es contigua e inmediatamente anterior ($e_p + 1 = b'_p$), si solapa ($b_p \in [b_{p'}, e_{p'}] \vee e_p \in [b_{p'}, e_{p'}]$), si es contigua e inmediatamente posterior ($b_p = e'_p + 1$), o si p empieza después de terminar p' ($b_p > e'_p + 1$). No es recomendable ofrecer sugerencias que solapen, ya que eso significa que estamos traduciendo dos veces la misma parte de la frase origen.

- Dos características que utilizan el modelo de alineamiento ligero descrito por Esplà-Gomis y Forcada (2012): el modelo de alineamiento ligero utiliza las traducciones de los n-gramas, al igual que el método de TAI de caja negra; explotando los segmentos ya traducidos, podemos estimar el alineamiento entre la frase origen y la frase meta. La primera característica codifica cómo de probable es que la sugerencia provenga de la parte de la oración origen que está siendo traducida (según el modelo de alineamiento); la segunda característica codifica una penalización para sugerencias que provengan de partes que han sido probablemente ya traducidas (según el modelo de alineamiento).

Durante el entrenamiento, las sugerencias que han sido usadas tienen una salida deseada de 1, y las sugerencias que no han sido usadas tienen un valor de 0, pero al usar la red neuronal en el proceso de evaluación, la salida se encuentra en el intervalo $(0, 1)$: este valor puede ser

[10] Por ejemplo, en este caso, cada sugerencia es un evento.

interpretado como la probabilidad de una sugerencia de ser usada. Esta probabilidad se usa para ordenar las sugerencias, ofreciendo las M más prometedoras.

La aproximación a la TAI de caja negra usando el modelo basado en aprendizaje computacional se usó en cuatro escenarios distintos. En el primero, se usó para traducir entre inglés y español usando el sistema de TA estadístico Moses y el sistema de evaluación automática. Se entrenaron os modelos de red neuronal distinto: uno que utiliza solo dos características que contienen la misma información en la que se basa el modelo heurístico (es decir, una característica con la distancia entre la palabra que se está tecleando actualmente y la posición de origen de la sugerencia, y otro con la longitud de la sugerencia), y un segundo modelo usando todas las características descritas. Ambos modelos mejoran los resultados obtenidos por la heurística: la mejora entre la heurística y el modelo neural usando las dos características muestra que, pese a que la heurística obtenía buenos resultados, estos se pueden mejorar los resultados procesando mejor la información; el modelo que usa todas las características obtiene mejores resultados que la heurística y el otro modelo, ahorrando hasta un 40 % de pulsaciones de teclas. Los resultados de estos experimentos han sido publicados (Torregrosa et al., 2016).

El segundo escenario utiliza TAI de caja negra y TAI de caja de cristal para traducir entre tres pares de lenguas, inglés y español, inglés y árabe, e inglés y chino simplificado, usando el sistema de evaluación automática. Como sistema de TAI de caja de cristal, se usa Thot (Ortiz-Martínez and Casacuberta, 2014), que es capaz de operar como un sistema de TA estadística y TAI usando el mismo modelo; para operar en igualdad de condiciones, las sugerencias del modelo de TAI de caja negra se tradujeron usando el modo de TA estadística de Thot y los mismos modelos. En la comparación entre el modo de TAI de Thot y la aproximación de TAI de caja negra, el sistema de TAI de caja negra obtiene mejores resultados en cinco de seis escenarios; Thot solo obtiene mejores resultados traduciendo de inglés a español, mientras que el sistema de TAI de caja negra obtiene mejores resultados traduciendo de español a inglés, y traduciendo entre inglés y árabe e inglés y chino en cualquier dirección. El ahorro de pulsaciones de teclas fue alrededor del 35 % para traducción entre inglés y árabe, 40 % entre inglés y español, y 20 % entre inglés y chino simplificado. Los resultados de estos experimentos han sido publicados (Torregrosa et al., 2017).

El tercer escenario utiliza los mismos modelos y sistemas que el anterior, pero evaluando los sistemas con traductores no profesionales que traducen de inglés a español. Los traductores tradujeron cinco frases sin asistencia, cinco frases con asistencia de TAI de caja negra y cinco frases con asistencia de TAI de caja de cristal. Los traductores ahorraron pulsaciones de teclas en ambas condiciones, ahorrando un 10 % de pulsaciones con la aproximación de caja negra y un 15 % con la aproximación de caja de cristal, pero solo ahorraron tiempo al traducir con la aproximación de caja negra, sistema con el que tradujeron un 4 % más rápido; en la prueba con TAI de caja de cristal, tradujeron un 12 % más lento. Al terminar la prueba, los usuarios debían responder un pequeño cuestionario; los traductores opinaron que ambas aproximaciones eran cómodas y les permitieron mejorar la calidad y velocidad de sus traducciones, pese a que los datos muestran que perdieron tiempo con una aproximación. Los resultados de estos experimentos han sido publicados (Torregrosa et al., 2017).

Finalmente, el cuarto escenario utiliza TAI de caja negra para traducir de inglés a español, usando el sistema de TA estadística Moses. El sistema fue evaluado por traductores profesionales, en una prueba de tres horas remunerada. Los traductores afrontaron tres pruebas de 45 minutos: la primera, sin asistencia alguna, la segunda, con asistencia de TAI de caja negra, y la tercera con asistencia de TAI de caja negra con un umbral que hacía que sólo alrededor del 25 % de sugerencias se mostraran (comparado con la prueba sin umbral).[11] En la segunda prueba, los traductores ahorraron hasta un 40 % de pulsaciones de teclas, y la mitad de ellos aumentaron su velocidad de traducción hasta en un 30 %; en cambio, los traductores obtuvieron peores resultados en la tercera prueba, ya que no se encontraron cómodos con la cantidad reducida de sugerencias. Al finalizar la prueba, los usuarios debían responder un pequeño cuestionario; la mayoría de ellos expresaron que la herramienta era cómoda de usar, y les permitió traducir más rápido, pese a que muchos opinaron que la herramienta les hacía la tarea de traducción más difícil, o que les hacía más difícil mantener el nivel de calidad. En cambio, todos los traductores menos uno opinaron que la aproximación con umbral era peor que la aproximación sin umbral, mostrándose frustrados con las sugerencias ofrecidas; a los traductores no se les explicó el tipo de asistencia que se les iba a ofrecer en cada prueba, y muchos expresaron que, si hubieran conocido de antemano que la tercera prueba iba a ofrecer una menor cantidad de sugerencias, pero que las sugerencias eran más prometedoras, hubieran afrontado la tarea de forma distinta. Solo un traductor expresó que la prueba con umbral era mejor, ya que la aproximación sin umbral ofrecía demasiadas sugerencias para su gusto.

Conclusiones y trabajo futuro

En esta memoria se describe el desarrollo de un sistema de traducción automática interactiva (TAI) que, a diferencia del resto de los existentes en el estado de la cuestión, que necesitan entrenar un modelo de traducción automática basada en corpus para funcionar, es capaz de utilizar cualquier tipo de recurso bilingüe para generar las sugerencias. Para ordenar las sugerencias que se ofrecen, se han desarrollado dos aproximaciones, una heurística basada en observaciones y una basada en aprendizaje computacional.

Ambas aproximaciones han sido evaluadas usando medios de evaluación automática, y tres rondas de evaluación humana: la primera para el modelo heurístico, y la segunda y tercera con el modelo neuronal; las evaluaciones primera y segunda fueron realizadas con traductores no profesionales, mientras que la tercera se realizó con traductores profesionales que recibieron una remuneración por participar en la prueba. La evaluación automática muestra que se puede ahorrar entre el 20 % y el 50 % de pulsaciones de teclas; los traductores ahorraron alrededor del 15 % de pulsaciones, pero podrían haber ahorrado hasta un 80 % si hubieran usado todas las sugerencias compatibles con las traducciones que generaron finalmente. La mayoría de

[11] Cómo se ha comentado previamente, la red neuronal da una puntuación en el intervalo $(0, 1)$ a cada sugerencia, que puede ser interpretada como la probabilidad de ser usada. Esta información permite definir un umbral de tal forma que solo aquellas sugerencias con alta probabilidad de ser usadas sean ofrecidas al usuario; si no se usa la estrategia del umbral, se ofrece un máximo de sugerencias fijo, siendo posible ofrecer sugerencias cuya probabilidad de ser usadas sea baja, aunque sean las más probables del conjunto.

usuarios se mostró satisfecho con la aproximación; de hecho, expresaron que les ayudó a traducir, y que su uso es cómodo.

El estado actual de la investigación permite vislumbrar diversas nuevas líneas de investigación, que se analizan en el Capítulo 4. Por ejemplo, se puede probar la aproximación de TAI de caja negra con otros tipos de recursos bilingües, como diccionarios, memorias de traducción, glosarios, traducción automática neuronal, etc. También se puede probar con combinaciones de recursos bilingües, ya sea traduciendo con todos los recursos y creando una nueva característica que codifique el recurso o tipo de recurso que ha generado la sugerencia, dejando al modelo de aprendizaje computacional que decida qué modelos son más útiles en qué condiciones; en caso que generar estas traducciones sea un proceso caro (ya sea computacionalmente o porque los recursos bilingües usados son pagados como un servicio), se puede desarrollar un modelo que permita prever qué sistemas de traducción funcionan mejor para cada segmento, reduciendo la carga necesaria para generar las sugerencias.

Una segunda vía de investigación viene de cómo se generan las sugerencias: actualmente, todos los n-gramas se traducen; es posible que otros tipos de subdivisiones mejoren el rendimiento del sistema y/o reduzcan la cantidad de traducciones realizadas. Por ejemplo, se pueden atender a criterios sintácticos o semánticos para dividir la frase en unidades, evitando traducir n-gramas que crucen la barrera entre unidades. También se pueden atender a modelos basados en observación empírica, como el modelo planteado por Carl et al. (2011), en el que se analiza la tarea de traducción y se subdivide en diversas unidades: el traductor se fija en una unidad, plantea la traducción, escribe la traducción, y pasa a la siguiente unidad; el tamaño de estas unidades depende del perfil del usuario, ya que se observó que traductores con mayor experiencia usaban unidades más grandes, y eran capaces de solapar la escritura de una unidad con la lectura de la siguiente.

Una tercera vía de investigación viene de cómo se ordenan las sugerencias en el modelo de aprendizaje computacional: actualmente, cada sugerencia se trata como un evento independiente, pese a que parte de las características dependen de la interacción entre sugerencias. En lugar de puntuar cada sugerencia por separado, se pueden usar modelos que aprenden a ordenar listas, como el descrito por Hang (2011); el uso de aprender a ordenar listas en lugar de tratar las sugerencias como eventos independientes puede llevar a mejorar el rendimiento del sistema de TAI de caja negra en dos formas distintas: no solo es posible que ordene las sugerencias mejor al tener más información,[12] sino que, además, puede llevar a una estrategia donde se diversifique el tipo de sugerencias ofrecidas.[13]

Una cuarta vía de investigación es la adaptación a los hábitos del usuario. Una posible forma de conseguir este objetivo es creando una lista de sugerencias confiables, es decir, aquellas que fueron usadas (o pudieron ser usadas) en traducciones anteriores. El sistema

[12]La información combinada de todas las sugerencias, en lugar de las características que reflejan interacción entre sugerencias.

[13]El modelo actual da puntuaciones parecidas a sugerencias parecidas; es posible que todas las sugerencias ofrecidas sean muy parecidas (posiblemente, que alguna sea prefijo de otra) y que, al no ser compatible una (por tener un prefijo incompatible), no lo sea ninguna. Es posible que el modelo que aprende a ordenar listas evite este escenario.

operaría como una memoria de traducción para sugerencias: al inicio, el traductor no obtendría ninguna sugerencia; conforme se traducen frases, aquellas sugerencias que se podrían haber usado en la traducción se añaden a la lista de sugerencias confiables, de tal forma que si las mismas sugerencias se generan en frases posteriores, se ofrecerán al usuario. De esta forma, no solo se ofrecen menos sugerencias al usuario, sino que además, solo se ofrecen aquellas consideradas de alta calidad (es decir, que se podrían haber usado para generar una traducción anterior) y que tienen un estilo similar al que usa el traductor.

Una quinta vía de investigación es la reducción de la repetición en las sugerencias que se ofrecen. En ocasiones, en la lista de sugerencias aparecen sugerencias que están contenidas en otras (por ejemplo, *la casa* y *la casa roja*). Al combinar ambas sugerencias en una se puede optar por no ofrecer más, reduciendo el tamaño de la lista de sugerencias, o introducir una nueva sugerencia que contenga texto distinto, aumentando la probabilidad de que al menos una sugerencia de la lista sea compatible con la traducción que se está escribiendo. Se pueden usar diversos modelos para combinar las sugerencias, como, por ejemplo, uno menos estricto que solo combina sugerencias si la sugerencia menos probable está contenida en la más probable, y uno más estricto que combina todas las sugerencias independientemente de su probabilidad.

Una sexta vía de investigación es el tratamiento de sugerencias que se han usado y rechazado; actualmente, el modelo de TAI no reacciona cuando una sugerencia se usa o se ignora. Se puede desarrollar un sistema que penalize aquellas sugerencias que se rechazan habitualmente para molestar menos al usuario; asimismo, se puede desarrollar un sistema que elimine aquellas sugerencias que han sido usadas, o sugerencias que solapen con las que han sido usadas.[14]

Finalmente, se puede mejorar el modelo de evaluación automática: el modelo actual opera con diferentes restricciones que pueden ser consideradas excesivas. Por ejemplo, se puede usar un modelo capaz de elegir sugerencias que no son compatibles con la referencia con la que cuenta el sistema de evaluación automática, pero pueden ser posteditadas para encajar con la referencia. También se puede diseñar un modelo que utilice una referencia con alternativas, es decir, con cierta capacidad de elegir palabras sinónimas o usar un registro o estilo distinto. Finalmente, se puede mejorar el sistema actual usado para contabilizar las pulsaciones de teclas, ya que no todas las acciones modeladas que "cuestan" una pulsación de tecla cuestan lo mismo.

Todo el software utilizado en esta memoria usa o ha sido liberado bajo una licencia de código libre y abierto. Forecat[15] es la implementación de ITP de caja negra desarrollada en paralelo con esta memoria, e implementa las diversas técnicas y aproximaciones descritas en ella; Forecat ha sido usado para medir automáticamente el rendimiento de todos los modelos descritos en la memoria, así como para realizar la primera de las pruebas con traductores,

[14] Una de las características del modelo de aprendizaje computacional ya penaliza sugerencias que solapan con las que han sido usadas, pero se pueden tomar otro tipo de medidas, como por ejemplo, eliminarlas.

[15] `https://github.com/transducens/forecat`

vía un navegador web. Forecat-OmegaT[16] es un complemento para la herramienta de TAO OmegaT[17] que introduce la funcionalidad de TAI, y ha sido usado para realizar dos de las tres pruebas con humanos.

[16] https://github.com/transducens/forecat-omegat
[17] http://www.omegat.org/

Preface

As the world becomes more globalized, multilingual communities proliferate, and minority languages get recognized by the society, the effort dedicated to translation and interpretation increases. For example, the European Union eagerly defends multilingualism as a main value for dialogue, cohesion, and prosperity: member states have the right to communicate and express in their own official languages. Currently, there are 24 official languages[18] in Europe, and more than 60 different languages are spoken. About half of the population in Europe claims to understand English, but only 10% are willing or able to consume information, applications and services in this language, what creates a segmented market. The European Union spends around 1 billion euros per year in order to keep the multilingualism policies (Rehm and Uszkoreit, 2013), and owns the largest translation entity in the world, the Directorate-General for Translation of the European Union, that hires almost 2 000 linguists (DGT annual activity report, 2015). Initiatives like eTranslation,[19] a machine translation service provided by the European Commission that translates from an into any official language, further promote multilingualism in Europe.

> In order to support cross-border exchanges between users, consumers, countries and regions [...], robust and high-quality cross- and multilingual language technologies need to be developed urgently. [...] Language technology is a key enabler for sustainable, cost-effective and socially beneficial solutions to overcome language barriers. (Rehm and Uszkoreit (2013), p. 7–8)

Computer assisted translation (CAT) technologies are used to improve the performance of human translators in order to keep up with the current translation demands. There are several different ways in which a CAT tool can assist a translator; namely, two of the most widespread approaches in the industry are post-editing and translation memories: in the former, a human translator mends style and inaccuracies of the output of a machine translation (MT) system;[20] in the later, past, already performed translations of similar sentences are used to speed up the translation process.

[18] With Croatian as the latest recognized language in 2013.

[19] `https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eTranslation`

[20] A piece of software capable of translating a sentence from one language to a different one without any human interaction.

This dissertation focuses on a different CAT modality, interactive translation prediction (ITP), where a computer assists a human translator by offering a prediction of the next words of the translation that is being carried out, with the objective of saving effort to the user. The translator may then accept the suggestion, or disregard it and keep typing, what forces the system to provide different suggestions. This process constitutes a dialogue between the translator and the computer, where both parties collaborate to speed up the translation process while keeping the translation devoid of errors.

Multiple approaches to ITP have been developed, but all of them share a common feature: they depend on a tightly coupled corpus-based machine translation system[21] that not only produces automatic translations, but also provides additional information needed to generate the ITP suggestions. These MT systems need extensive parallel corpora to be trained, as, if they are trained with a small amount of data, the translation quality may be too low to generate useful suggestions, and generating enough resources to train an useful system may be an expensive task.

In this dissertation a black-box approach to ITP is proposed, one that can use any bilingual resource capable of providing one or more translations of source language subsegments without needing any additional information. Not relying on a specific resource allows to include new sources of bilingual information: translators can use any resource they have access to, such as dictionaries, rule-based machine translation or paid translation-as-a-service system. This proposal splits the segment to be translated in all the possible overlapping subsegments up to a given length, then translates them by means of any available bilingual resource. A neural network model trained over some thousand sentences is used to rank and select which suggestions are shown to the user; if no parallel corpora is available at all, a simple heuristic can be used instead.

A free/open-source black-box ITP implementation, `Forecat`,[22] was developed to prove the concepts and techniques that were devised. Forecat can be used in three different configurations:

- As a web page, capable of both working in a full server-side mode, or in a client-side mode that offloads most of the server work.
- As a collection of web services providing the same functionality as the server-side mode of the web page. A web component[23] using this services is also available.
- Integrated into OmegaT,[24] a free/open-source translation memory application intended for professional translators.

[21] Corpus-based MT systems are systems that learn how to translate from already available bilingual corpora with little to no human interaction. Notable corpus-based MT approaches are statistical machine translation (SMT) and neural machine translation (NMT).

[22] `https://github.com/transducens/forecat`

[23] Web components (`https://www.w3.org/standards/techs/components`) is a recently developed W3C standard that lets developers to encapsulate HTML elements into reusable components.

[24] `https://omegat.org`

This thesis has been possible thanks to the ideas and constant supervision of Prof. Juan Antonio Pérez Ortiz and Full Prof. Mikel L. Forcada from the Departament de Llenguatges i Sistemes Informàtics at Universitat d'Alacant.

Structure of this dissertation

This dissertation is structured in 4 chapters and 3 appendices:

Chapter 1 introduces the most important concepts needed to understand translation tasks and computer-assisted translation, explores the history and state-of-the-art of interactive translation prediction, and contextualizes the motivation behind the development of the black-box techniques.

Chapter 2 describes the basics of black-box interactive translation prediction, the automatic evaluation procedure and metrics used to measure the performance, a language-model-based monolingual text predictor used as a baseline, and the simple heuristic used to rank and select the suggestions as a proof of the black-box ITP approach concept, along with preliminary human evaluation.

Chapter 3 presents a more principled approach to rank and select the suggestions based on a neural network that improves the performance of the heuristic approach, along with human evaluation.

Chapter 4 summarizes the main contributions of the work and outlines future research topics.

Publications

A large part of this dissertation has been published in peer-reviewed conference proceedings:

Torregrosa, D., Pérez-Ortiz, J. A., and Forcada, M. L. (2017). Comparative human and automatic evaluation of glass-box and black-box approaches to interactive translation prediction. *The Prague Bulletin of Mathematical Linguistics*, 108(1):97–108 **[Chapter 3]**

Torregrosa, D., Forcada, M. L., and Pérez-Ortiz, J. A. (2016). Ranking suggestions for black-box interactive translation prediction systems with multilayer perceptrons. In

Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA), volume 1: MT researchers track, pages 65–78 **[Chapter 3]**

Torregrosa, D., Forcada, M. L., and Pérez-Ortiz, J. A. (2014). An open-source web-based tool for resource-agnostic interactive translation prediction. *The Prague Bulletin of Mathematical Linguistics*, 102(1):69–80 **[Chapter 2]**

Pérez-Ortiz, J. A., Torregrosa, D., and Forcada, M. L. (2014). Black-box integration of heterogeneous bilingual resources into an interactive translation system. *EACL 2014 Workshop on Humans and Computer-assisted Translation*, pages 57–65 **[Chapter 2]**

Additionally, oral presentations were accepted in workshops:

Torregrosa, D., Forcada, M. L., and Pérez-Ortiz, J. A. (2016). Comparison of two different interactive translation prediction approaches integrated in a computer-assisted translation workbench. In *Workshop on Interacting with Machine Translation (iMT)*. **[Chapter 3]**

Torregrosa, D., Pérez-Ortiz, J. A. (2013). Computer assisted translation using machine translation of source sentence segments. In *WoTI 2013*, pages 2–24 **[Chapter 2]**

Contents

Chapter 1

Introduction

There are multiple approaches that can be taken when confronted with a translation task: the choice depends on the context of the document, the task, and the purpose and the public that will read the translation. These approaches can be located in a spectrum where one end represents an entirely human-conducted translation and the other represents a totally automatic, machine-conducted translation provided by some kind of machine translation (MT) system. Some notable approaches in this spectrum are:

- **Human translation (HT)**: a human translator (who understands both the source and target languages) reads a document in a language and, by using all the knowledge available, writes a document in a different language that conveys the same information.
- **Translation memories (TM)**: an automatic system (also named translation memory manager, TMM) searches in a database of previously performed translations (the translation memory) for sentences that are highly similar or equal to the one that is going to be translated, and offers the best match or matches. A human translator then reads the suggestions, and either accepts one of them, translating or mending the parts of the translation corresponding to mismatches (if any), or provides a new translation if the match was not accurate enough.
- **Interactive translation prediction (ITP)**: a human translator and a bilingual resource collaborate to translate the sentences; while the user types the target sentence, the ITP system predicts and suggests the continuation of the translation the user is going to type next using the information contained in the bilingual resource. This approach is the main focus of this dissertation.
- **Post-editing (PE)**: an automatic MT system provides a translation, that is then checked and mended by a human translator by adding, removing or reordering segments of the proposed translation.
- **Machine translation (MT)**: the translation is carried out by a MT system that automatically produces a target sentence given a source sentence; this is useful in some

constrained language scenarios, where the output of a MT system is considered good enough to serve for the same purpose as the original text, or to gist the meaning of a document written in a foreign, unknown language.

Each one of these different approaches have different advantages and drawbacks. Therefore, the decision of choosing one of them depends on many factors, such as the availability of resources, languages involved, domain of the texts, scope of the task, budget, etc. For example,

- Translation memories can be useful when translating multiple versions of the same document, as large sections of the document will probably remain unchanged. Translating again the same sentences is a waste of resources.
- Machine translation can be useful to gist information from a source in a unknown language for a very reduced cost (or even for free); for example, when combined with optical character recognition, it can be used to translate a sign in a foreign unknown language, even if it uses a different alphabet or script.
- Post-editing can be useful when an available bilingual resource (such as a MT system) produces translations that need little revising from a human to reach an adequate quality. Conversely, if the output of the bilingual resource is garbled or unintelligible, the effort needed to fix it will most probably be greater than simply translating from scratch.

In most cases where human translators and computers collaborate to generate the translation, a computer assisted translation (CAT) tool is used. CAT tools usually integrate one or more functionalities such as:

- Spelling and grammar checkers, that are integrated into the workflow to help the translator keep all the tools available in the same interface.
- Dictionaries, terminology databases, glossaries and bilingual concordancers, which help the translator to write consistent translations in the context of a given domain or task.
- Machine translation, used either as an inspiration or aid for the human translator or as the basis for post-editing.
- A translation memory manager, as previously explained.
- Other project-management related tools that allow translator groups or translating agencies to split a large translation task into different, smaller subtasks that can be performed by a single translator. The tools also provide the same resources (such as glossaries and translation memories) to all the translators involved; by using the same resources, the translations are kept homogeneous and consistent so they can be assembled to form the original task in a seamless way.

In this chapter, the basic concepts needed to define and contextualize the motivation of the research developed in the framework of this dissertation will be introduced: first, the

different kinds of machine translation systems that currently exist will be further explored in Section 1.1; then, translation memories and post-editing will be described in Section 1.2 and Section 1.3 respectively; finally, the main topic of the dissertation, interactive translation prediction (ITP), will be introduced, and the state of the art of interactive translation prediction will be explored in Section 1.4 and the history of ITP human evaluations will be summarized in Section B

1.1. Machine translation

Machine translation (MT) is a process where a piece of software is used by a information system to translate a text in a source language into a target language without the need of interacting with a human. MT can be used for two different scenarios:

- The first scenario is called **assimilation**: the MT system is used by a human to effortlessly translate text in an unknown language into a known one. The translation is then used by the human as is to gist the meaning of a previously unreadable text, even if the output has improper syntax, is ungrammatical, contains untranslated words or has other kinds of mistakes and inaccuracies that do not make the information stop being useful for its purpose.

- The second scenario is called **dissemination**: in this scenario, not only the translated information has to be useful, but also needs to adhere to higher language standards.[1] The text may be revised to not only mend mistakes, but also reduce the ambiguity, improve the accuracy and enhance the style; although in some scenarios this revision may not be necessary, such as MT operating on tightly constrained domains or very close languages.

MT systems can be classified depending on how the system acquires the linguistic information and how the information is used.

1.1.1. Rule-based machine translation

Rule-based MT (RBMT) is a knowledge-based system where humans (such as linguists, translators, or people with a high proficiency in a language) create declarative knowledge such as rules and dictionaries that encode the knowledge needed to perform the translation task. The information is encoded in the form of rules, e.g. in English, the adjective usually appears just before the noun, but, in Spanish, the adjective usually appears after; the expert would create a rule that reorders adjective+noun into noun+adjective. RBMT systems can be

[1] This standards are not absolute and depend on the particular task: for example, translated medical texts should pass a stricter revision than instruction leaflets of home appliances.

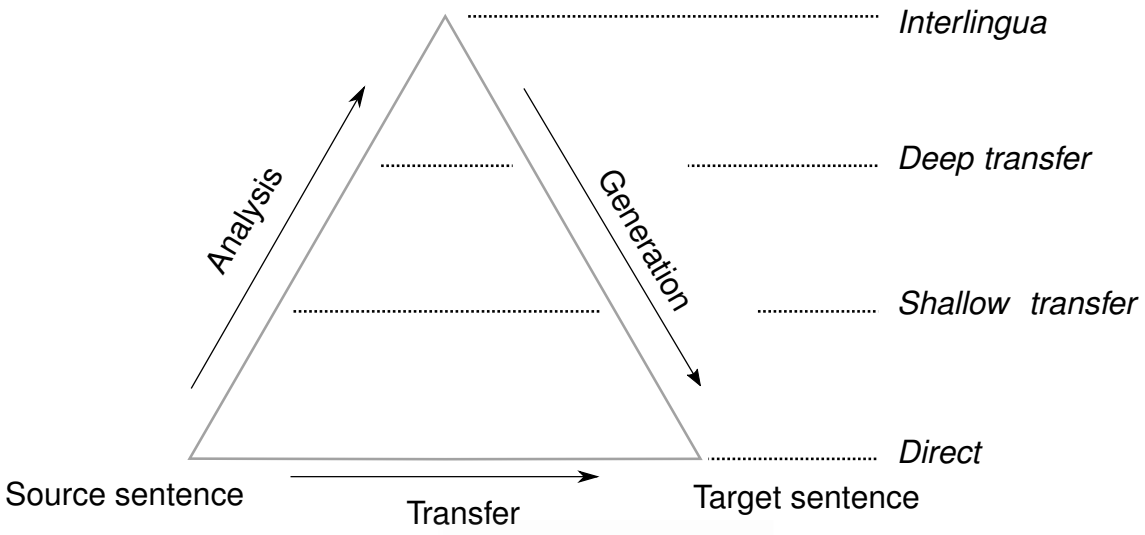


Figure 1.1: Vauquois triangle. Labels in italics represent some classes, but the triangle defines a continuous space, and systems can work at any level, making the task of classifying them very ambiguous.

classified in different categories using the Vauquois (1968) triangle (shown in Figure 1.1), that splits the translation problem into three different tasks (analysis, transfer, and generation):[2]

- On the bottom of the triangle there is a case where no analysis or transfer is used, the direct translation. This approach is only fit for simple translation tasks, such as translating single tokens,[3] as it is incapable of reordering or adding tokens when needed.

- The complexity of the transfer strategies increase with their height in the triangle. This strategies are characterized by using an analysis system that transforms the source sentence into some intermediate representation, a transfer system that converts the source-language intermediate representation to target-language intermediate representation, and a generation system that generates the final target-language sentence using the target-language intermediate representation. For example, a system can use morphological analysis to transform each source sentence into the lemma o canonical form of the word and categorized with morphological information; then, the lexical transfer (such as a bilingual dictionary that operates over the lemmas or canonical forms) translates the words into the target language; finally, the system uses morphological generation to

[2]The Vauquois triangle can also be used to classify other kinds of MT systems, but it is mainly used to classify RBMT systems, where selecting a different level for analysis, transfer or generation can drastically modify the nature of the declarative knowledge encoded in the system.

[3]Usually, a single word constitutes a token, but there are many exceptions. For example, the English possessive ('s) is usually regarded as a separate token; some languages, such as Chinese or Japanese, do not clearly separate words using spaces; and some agglutinative and polysynthetic languages, such as Japanese or the Turkic language family, allow multiple morphemes to be joined in the same unit or have morphemes that cannot exist as isolated words.

transform the lemma or canonical form into the correct form using the morphological information. Additional rules may be added to reorder words or fix the agreement of the morphological information before the generation[4] or to deal with ambiguity.[5] Also, additional modules may be added, like syntactical and semantic analysis and generation. Depending on the strategies used, RBMT systems may be classified as shallow transfer (such as Apertium (Forcada et al., 2011)) or deep transfer (such as TectoMT (Žabokrtský et al., 2008)), although there are no strict definitions for shallow and deep.

- At the top of the triangle there is an special case that only uses analysis and generation, named interlingua. The source sentence is analyzed into a language-independent representation, then generated in the target language. This approach is very attractive as, once a language has an analyzer, it can be translated into any other language with a generator; similarly, when a language has a generator, it can be used to translate from any other language with an analyzer. The main disadvantage is that creating an intermediate representation is a task that becomes more and more complex as more languages and domains are represented. An example of a system using this approach for a set of limited domains is grammatical framework (Ranta, 2004).

The main advantages of RBMT systems are:

- Unlike corpus-based MT systems, no corpora are needed to create a system. This allows for the creation of MT engines for under-resourced language pairs with little to no parallel corpora available.

- The process is deterministic and the knowledge used to perform a translation can be traced and modified to provide the expected output; every error can, in principle, be fixed with a rule, no matter how unusual it is.

The main shortcoming of RBMT systems is that they need lots of human work to be created, and that the workload of maintaining a system grows with the scope of the task, as it needs to deal with more ambiguity and interactions between words. This kind of system is useful for tightly constrained domains, for tasks that require precise control over the output and for language pairs that lack publicly available parallel corpus. Notable RBMT systems are Systran (Toma, 1977), Lucy[6] or Apertium (Forcada et al., 2011).

[4]For example, when translating adjective+noun from English to Spanish, two modifications should be performed: one that inverts the order of the words (as, in Spanish, the noun usually precedes the adjective), and a second one that decides the gender of the words, as English nouns and adjectives are genderless, but Spanish ones have different surface forms depending on the gender.

[5]Some words may have more than one possible interpretation due to homography; for example, in English, the word “sow” can be a noun (female pig), or a verb (to plant seed). There are different strategies to deal with this: some systems may choose to disambiguate as soon as possible and use an unique representation for the word, and others may delay the decision and contemplate the different possibilities for multiple steps in the process.

[6]`http://www.lucysoftware.com`

Example 1.1

To illustrate the different components of a statistical machine translation (SMT) system, the Spanish sentence

La mesa gruesa marrón

and its English translation

The thick tan table

will be used.

1.1.2. Statistical machine translation

Statistical MT (SMT) is a kind of corpus-based MT system in which machine learning techniques are used to translate. SMT systems do not encode the knowledge of an expert;[7] rather, a set of language-and-domain independent features are defined and used to analyze vast amounts of parallel bilingual corpora (a text or collection of texts, each one translated into one or more languages). The cost for training an SMT system is rather low, as it is mostly fully automatic and needs little to no human input, but it is only possible to do so if bilingual corpora exists. For some language pairs, corpora is readily available: for example, some public organizations generate documentation in multiple languages that can be used to train a SMT system, such as the United Nations (Ziemski et al., 2016) (English to Spanish, French, Russian, Simplified Chinese and Arabic) or the European Parliament (Koehn, 2005) (English to all the other 23 official languages in the European Union); but texts of adequate size of other domains or languages may not be readily available. Some notable statistical MT systems are Moses (Koehn et al., 2007)[8] or Yandex.Translate.[9]

There are many different types of SMT approaches, such as phrase-based, syntax-based or hierarchical. Phrase-based SMT is the basis for many classical and state-of-the-art interactive machine translation (ITP) approaches (which will be addressed in Section 1.4). For this reason, how phrase-based SMT works is briefly introduced, aided by the Example 1.1.

Formally, given $S = s_1 s_2 ... s_{|S|}$ as a sentence in the source language, the SMT system tries to find $T_{\text{best}} = t_1, t_2 ... t_{|T|}$, an equivalent sentence in the target language that maximizes:

$$T_{\text{best}} = \underset{T}{\operatorname{argmax}}\, p(T|S)$$

[7] There is no need for translators when training a new language pair, as the SMT system does not need to be changed, and the process is automatic. Still, the bilingual corpora needed to train the system was produced by translators, and translation knowledge was needed when developing the SMT system features.

[8] Moses is not only a SMT system, but a platform that includes all the different tools needed to train your own SMT system.

[9] Yandex.Translate follows a hybrid SMT/neural machine translation model since September 2017, and it is available at `https://translate.yandex.com/`

Example 1.2

A possible translation model table for the word *gruesa* when translating Spanish to English.

s	$p(s\|t)$
thick	0.4
gross	0.4
broad	0.15
heavy	0.05

where $p(T|S)$ is the probability of T being a translation of S. As these probabilities are unknown, they are approximated using the maximum likelihood estimation: $p(T|S) \simeq c(S,T)/c(S)$, where $c(S)$ is the number of times S as been seen during the training and $c(T,S)$ the number of times T has been seen as a translation of S. This model would operate similarly to a database: a sentence S_{new} that has never been seen before, cannot be translated ($\forall T p(T|S_{\text{new}}) = 0$); similarly, it cannot generate new translations.

Therefore, this model is mostly useless as a general-purpose MT engine. To address this problem, the process is broken up in smaller steps, then combine them to obtain the initial model, in a process called generative modeling.

1.1.2.1. Word-level statistical machine translation

The first approach used was the noisy-channel model, a classic model originated during the early development of the information theory (Shannon, 1948), that combines a translation model $p(S|T)$ and a language model $p(T)$:

$$T_{\text{best}} = \underset{T}{\operatorname{argmax}}\, p(T|S) = \underset{T}{\operatorname{argmax}}\, \frac{p(S|T)p(T)}{p(S)} = \underset{T}{\operatorname{argmax}}\, p(S|T)p(T).$$

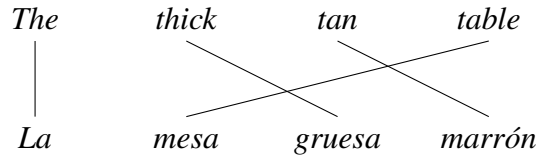
As $p(S)$ is constant for the purpose of finding T_{best}, it can be removed from the equation.

A translation model is a function that, given one source language word[10] s, returns a probability distribution with the probability of each word in the target language. This probability is estimated using the maximum likelihood estimation, that simply counts how many times the target word t is aligned with the source word s over the amount of times t has been seen in the corpora. Let $c(t)$ be the amount of times the word t has been seen in the corpora and $c(t,s)$ the amount of times that s cooccurs alongside t in a sentence pair in the corpora,[11] the probability of translating s to t is approximated as:

[10] Translation models can operate at many levels; in this example, it will operate with words.

[11] That is, S and T are aligned in the corpora and $s \in S$ and $t \in T$.

Example 1.3



A possible alignment between the English sentence *The thick tan table* and the Spanish sentence *La mesa gruesa marrón*

$$p(s|t) \approx \frac{c(t,s)}{c(t)}.$$

See Example 1.2 for an example of a translation model table.

The translation model can also be used as an *alignment model* by aligning each t with the s that maximizes $p(s|t)$. This is known as the IBM Model 1 (Brown et al., 1993), that can map multiple source words with the same target word. But Model 1 has obvious flaws: it does not reflect target words that should be left unaligned[12] and has trouble if the same word appears more than once in the same sentence. There are more complex IBM models (2 to 5) (Brown et al., 1993) that incrementally solve different problems and therefore improve the accuracy of the alignments. See Example 1.3 for an alignment example.

A language model is a probability distribution over sequences of words.[13] It is used to assess the probability that a the sentence t belongs to the language or domain that is represented in the language model. The most common used language model is the n-gram language model, that approximates the probability of a word t_i using the previous n words as context. The symbols start and end are usually included to denote the start and end of a sentence. For example, a 3-gram language model would be modeled as

$$p(t) = p(t_1|\text{start})p(t_2|\text{start}\,t_1)(\prod_{n=1}^{|T|} p(t_n|t_{n-1}t_{n-2}))p(\text{end}|t_{|T|-1}t_{|T|})$$

When combined with the translation model, the language model has two effects: it guides the translation so it is more fluent in the target language and it helps to reduce the ambiguity of some words. In this example, it is equally probable to translate the homograph Spanish

[12] That is, the word has to be added to the target-language sentence and has no equivalent in the source-language sentence.

[13] As with the translation model, it may operate at different levels; in this example, the language model will operate over words, but other models are possible, e.g. a language model based on letter sequences.

Example 1.4

A possible translation options table for the Spanish sentence “La mesa gruesa marrón”, using a maximum phrase size of 2 and 2 options per phrase.

Source sentence	La	mesa	gruesa	marrón
Translation options of length 1	the	table	thick	brown
	her	board	heavy	tan
Translation options of length 2	the table		thick tan	
	the board		thick brown	
		thick table		
		heavy table		

word *gruesa* into the English adjective *thick* or the noun *gross*[14] (see Example 1.2). But the English language model will give a higher probability to *thick table* (adjective+noun) than to *gross table* (noun+noun).

So far, this SMT model only works with words, and the alignment model only can map one source word to one target word; the system is unable to translate one source word into multiple target words; e.g. the model would be unable to translate the German compound word *Sprachgefühl*, that is translated to *‘feeling for language’*. To address this problem, the concept of multi-word units was proposed; this units were named *phrases*. By using phrases, the IBM models can map many source words into many target words. To extract phrases, the model seeks for groups of words in source language whose words are exclusively aligned to a group of words in target language and vice-versa. Therefore, the translation model that operates on words is replaced with the phrase model that operates on phrases.

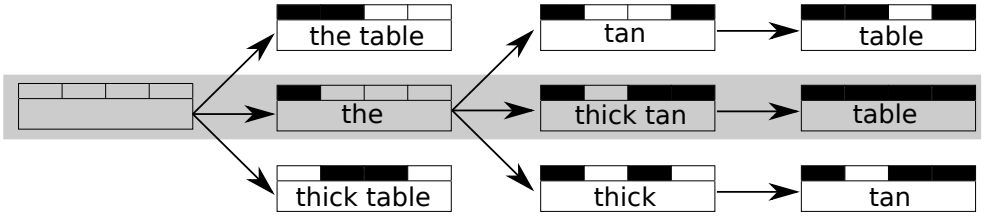
This simple scheme that operates with a phrase model and a language model «achieves generally better translation quality than the word-based statistical IBM Models» (Koehn (2010b), Chapter 5.3).[15] More advanced systems that add more models to the equation exist; a common approach is the log-linear model, that weighs each model with a parameter that usually gets learned during the tuning process. For example, Moses (Koehn et al., 2007) uses the IBM Model 2 distortion function for improving the reordering, the inverse translation probabilities ($p(s|t)$), lexicalized weighting for dealing with rare phrase pairs, word penalties that penalize translations that are too long or too short, and phrase penalties that controls if the model prefers to split a sentence in fewer longer phrases or more shorter ones.

Finally, all these models are combined to find the best-scoring translation in a process known as *decoding*. The decoding process splits the source sentence in all the phrases known by the phrase model and obtains their translations, also known as the *translation options* (see Example 1.4 for the translation options available for this example). The process is initialized with the empty hypothesis, a blank string that translates no words from the source

[14] A group of 144 elements.

[15] The system this quote refers to also adds a simple distortion model: an exponentially decaying cost function that makes long range movements more expensive than shorter movements (Koehn, 2010b, Chapter 5.1.2).

Example 1.5



Decoding process. Each node represents the phrase being added and the words already covered (on top of each node); the actual translation for a given path is the ordered concatenation of the phrases of all the nodes of the path, starting with the empty hypothesis. Only a limited number of nodes are expanded, as the complete tree would expand each node with every phrase that does not cover words that have already been covered in the path. Usually, the expansion algorithm uses techniques to discard less promising branches. The highlighted path is the most likely translation according to the model.

sentence. Then, the hypothesis gets expanded by picking a translation option that does not cover source words already covered in the hypothesis. This process is recursively repeated until all the words are covered; the most probable complete translation hypothesis is selected as the translation for the source sentence. See Example 1.5 for a possible decoding trace for this example. This process is computationally expensive, and fully exploring all the possible hypothesis is unfeasible; different techniques such as beam searching, pruning, stack decoding or limiting the reordering distance are used to greatly reduce the amount of hypothesis without damaging the quality of the final translation.

1.1.3. Neural machine translation

Neural machine translation (NMT) is a kind of corpus-based machine translation that his appearing in the recent literature (Bahdanau et al., 2015; Sutskever et al., 2014), although pioneering work describing the possibility of using recurrent neural networks to translate was published in the past (Neco and Forcada, 1997; Pérez-Ortiz et al., 2001). Rather than defining what kind of features the system should use to translate, it uses a deep-learning model in which a self-learned representation of the words is directly used to translate.

A neural network is a machine learning approach inspired in biological neural networks. The main component of the neural network is the neuron (or unit) that receive an arbitrary amount of inputs and generate an output value using an activation function. The units are interconnected and organized in layers:

- The input layer is the first layer. The units in this layer have no inputs, and its outputs are a representation of an event.

- The output layer is the last layer. The units in this layer have no outputs, and, after the network stabilizes, the internal state of the units contains a prediction (for example, the predicted class of the event) for the given input.
- One or more hidden layers. The units in the hidden layers have both inputs and outputs, and can be interconnected in many ways. If the network connections can be represented as an acyclic directed graph, it is considered a feedforward neural network; if there are loops, then it is considered as a feedback or recurrent neural network.

The input of each unit is weighted; that is, each unit treats the output of connected neurons differently. The weights are learned automatically in the training process.

The NMT approach uses a special kind of unit capable of storing information for an arbitrary interval, such as the long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997) or the gated recurrent units (GRU) (Cho et al., 2014). These units follow a complex recurrent architecture, where a central unit stores a state, and different gates define the behaviour of the unit.

There are multiple NMT approaches, but the most common one uses two recurrent neural networks, one acting as a encoder and another as a decoder; the encoder receives an arbitrary length sentence, one token at a time,[16] and builds an intermediate representation of the sentence. If the neural network uses an attention mechanism (Bahdanau et al., 2015), then the intermediate state after each token is received is saved; otherwise, only the final state is used. Then, the decoder uses the final state (or the collection of states, guided by the attention mechanism) to generate the output sentence, one token at a time, until the end-of-segment token is the most likely one. All the different models (the encoder, the decoder, and the attention model if it is being used) are jointly trained.

1.2. Translation memories

Translation memory managers (TMM) provide a kind of CAT that focuses on reusing already performed translations to speed up the translation of new sentences. Translation memory managers use translation memories (TM), databases composed of several translation units, that is, text of variable granularity,[17] attached to a translation performed in the past by a human translator.

The TMM uses a fuzzy-match algorithm to retrieve translation units that have a highly similar or identical source segment. The matches are offered to the translators, indicating the source-side tokens that do not match with the current source segment, along with the translation performed in the past. TMM can be usually configured to only show translation units over a specific similarity threshold, as only very similar units are useful. For example,

[16] In some models, the encoder receives the full token sequence, then the same sequence but reversed, one token at a time.

[17] Such as phrases, whole sentences, sentence-like segments (titles, items in a list), etc.

Trados Studio[18] has a default threshold of 70%, meaning that the TMM only shows translation units whose source segment has, at least, 70% of the words present in the current source segment. Some systems offer additional confidence measures, such as matching not only the segment being worked in, but also the adjacent ones, or even full paragraphs or documents. Also, the TMM can show additional information to the user, such as not only showing the mismatching words in the source-language part of the matching translation unit, but also in the target-language part (Espla-Gomis et al., 2015).

Translation memories help to accelerate the process of translation tasks similar to the ones that have been performed in the past (such as different versions of the same document, or texts in the same domain), and to keep the consistency of the translations, which is specially useful if there are multiple translators working in the same document, or if different versions of the same document are being translated.

1.3. Post-editing

Post-editing is a type of CAT in which a human mends the errors or inadequacies or improves the style of the output of an MT system.[19] While it may appear that the set of skills needed in post-editing and translation task overlap, they are not exactly the same. Post-editing studies report significant increases in performance and it is heavily advised by the translation industry; the amount of tasks performed using post-editing increases yearly.[20]

1.4. Interactive translation prediction

Interactive translation prediction (ITP), also known as interactive machine translation,[21] is a CAT modality where a human translator and a program collaborate to create a translation. The main difference between ITP and the rest of CAT modalities is that there is an active collaboration between human and program: as the human types the translation, the ITP system offers context-sensitive suggestions, that the translator can either accept or ignore; by contrast,

[18] http://producthelp.sdl.com/sdl%20trados%20studio/client_en/Edit_View/TMs/EVWorkingwithTMsAbout_Translation_Memory_Matches.htm

[19] The main difference between TM and post-editing is that, in TM, the initial point is the translation (performed by a human translator) of a similar sentence, and the errors to mend come from the differences in the source sentence; meanwhile, in post-editing, the initial point is the output of a MT system.

[20] See the studies by TAUS (https://www.taus.net/think-tank/best-practices/postedit-best-practices/machine-translation-post-editing-guidelines), Memsource (https://www.memsource.com/blog/2015/08/18/post-editing-in-memsource/) and Autodesk (http://langtech.autodesk.com/productivity.html).

[21] While the later denomination appears in most of the classic literature, the former term is more common nowadays, as it better represents how this CAT approach works. For simplifying this section, the terminology ITP will be used to refer to interactive machine translation, even when the referenced paper uses the interactive machine translation denomination.

Example 1.6

S	*La mesa gruesa marrón*
T	*The thick tan table*
Prefix	t
Offered suggestions	**the** the thick table table thick tan table
Prefix	**the_**
Offered suggestions	
Prefix	t*he* t
Offered suggestions	table **thick tan table** the the thick table
Final translation	**the thick tan table**

Example of an ITP session translating from Spanish to English, with the maximum number of suggestions 4. The suggestions in bold are the ones selected because they were the most useful for advancing in the translation; the bolded parts of the prefix have been inserted by accepting a suggestion. In the second prefix, a trailing space has been replaced with an underscore for visibility; no underscore has been typed or inserted via suggestion in this example.

both post-editing and translation memories offer a fixed suggestion that does not react to human interaction. The process can be seen as a refinement of monolingual text prediction technology similar to current mobile keyboards autocompletion technologies; rather than offering the most probable word according to a monolingual model, the suggestions are informed by the source sentence. See Example 1.6 for a ITP session.

MIND (Kay and Martins, 1970) was the system that coined the ITP term, but the approach was very different from current ITP: the system asks the user different questions about the source sentence that helps to disambiguate the sense of the words, then generates a fully automatic translation of it. The main aim of this system was to use monolingual source-language speakers to improve the quality of the automatic translations without requiring the interaction of a professional, bilingual translator; the user helps the machine by answering questions that are easily answered by humans, but hard to encode in software. Several systems followed this approach, until Foster et al. (1997) argued that

> In our opinion, these problems would be greatly alleviated if the focus of interaction were shifted from the meaning of the source text to the form of the target text. This would relieve the translators of the burden of having to provide explicit

> source text analyses, and give them direct control over the final translation without having to resort to postediting. It would also make possible a very simple and natural style of interaction consisting of manipulations of the actual words and characters in the target text. In such a system, a translation would emerge from a series of alternating contributions by human and machine, with the translator's inputs serving as progressively informative constraints for the MT component, which would normally respond to each of them with a fresh proposal for all or part of the target text.
>
> This approach, which might be called target-text mediated (TTM) IMT [interactive machine translation], encompasses a number of interesting possibilities. (Foster et al. (1997), p. 177)

ITP[22] systems have two main characteristics:

- The final translation is created by the interaction between human and machine; user input constrains the suggestions offered by the machine. Foster et al. (1997) argues that this is a reformulation of the main idea of a previous project, Transtalk (Brousseau et al., 1995), a translation dictation system that improved the accuracy of the target-text speech recognition by enriching it with the output of an SMT system that translates the source text.
- When the target sentence gets modified by the user by inputting one character, the machine gets the cue to update and provide suggestions.[23]

Early interactive translation prediction: TransType (Foster et al., 1997; Langlais et al., 2000) was the first tool following the target-text mediated ITP paradigm. TransType was developed at the Université de Montréal, and the main objectives of the project were

> - When the translator has a particular text in mind, the system's proposals can speed up the process of typing it
> - When the translator is searching for an adequate formulation, the proposals can serve as suggestions
> - When the translator makes a mistake, the system can provide a warning, either implicitly by failing to complete the mistake or explicitly by attaching an annotation for later review
>
> (Quote retrieved from `http://rali.iro.umontreal.ca/rali/?q=en/TransType` 1st May 2017)

[22] As MIND-like ITP fell out of use, target-text mediated ITP was shortened to just ITP.

[23] ITP using different input methods such as text-to-speech or electronic pen (a device that captures the handwriting or strokes performed by a human, then transforms them into digital input) has been researched in the CASMACAT project (Ortiz-Martínez et al., 2012).

when translating between English and French. TransType faced and partially solved the first problems of ITP:

- how to generate and rank the suggestions in order to show them in a timely manner, without slowing down the translator
- and how to efficiently interact with translators.

Some of the techniques used in current ITP were planned, but not implemented in this tool, such as suggestions for segments with more than one word. Foster et al. (1997) argue that an SMT-based approach is the most viable for this purpose:

- First, an evaluator estimates the probability of different translations for the source sentence, using:
  - an interpolated trigram language model,
  - a hidden markov model (HMM),
  - an alignment model that takes into account the target sentence length,
  - a translation invariant manager, that captures tokens that are not to be translated, such as proper nouns, numbers or codes,
  - and an anti-cache model, a small window of recent target-text tokens that are forcibly assigned a very low probability if they appear multiple times.
- Then, a generator matches the current prefix and the different translations generated by the evaluator in order to show the best suggestions.

To address the real-time requirements of this process (as the suggestions need to be offered in a short timespan in order to not slow down the translator), the vocabulary is divided into two sets: one with the most usual words (named *active component* in their work), and a second one with the rest of the vocabulary (named *passive component*). The most probable words from the active component are offered to the user; only if there are not enough words in the active part to fill the list, it gets expanded with the matching words in the passive component. The anti-cache model penalizes repeating the same suggestions in a small window: this often allows errors at the end of the suggestion (e.g. incorrect morphological form) to be fixed when the user types the next characters of the word. In some tests, hard-coded multi-word suggestions named “briskels” were used.

The TransType interface was split into two areas: the left one contained the source-language segments, and the right one their translations. As users typed, suggestions were offered in the form of drop-down menus; users may select one of the suggestions with a special command, or just ignore them by keep on typing; another command is used to validate the current segment and proceed to the next one. The TransType interface can be seen in Figure 1.2.

Several studies have been done on this tool. They assessed the performance of the tool via automatic evaluation, showing that the translations could be completed using only a

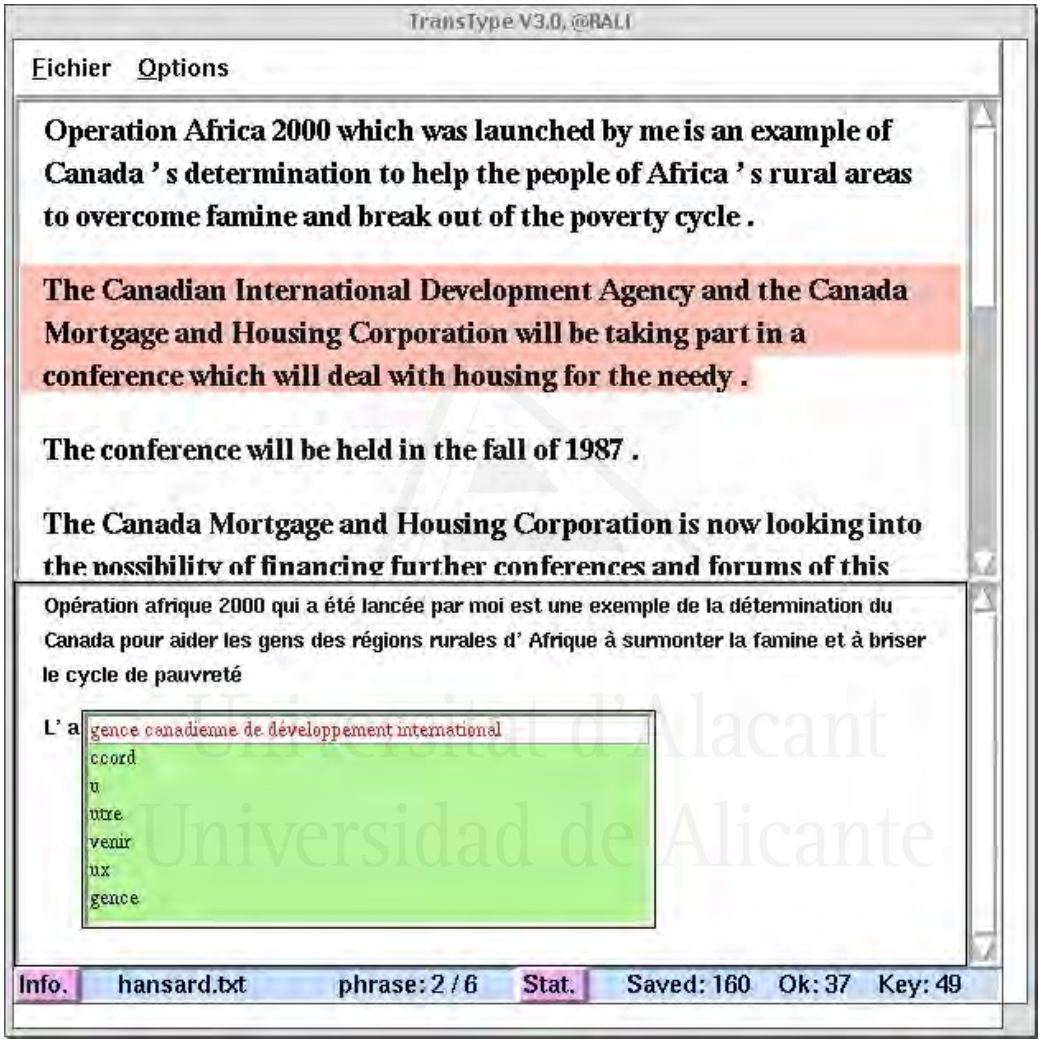


Figure 1.2: TransType interface. The top half of the screen contains the source segments, and the target is typed in the bottom half. The completions are given in a drop-down menu at the insertion point. The phrase (named "briskel" in the paper) *[a]gence canadienne de développement international* also appears as a suggestion. The figure was taken from the paper by Langlais et al. (2002).

third of keystrokes, compared to unassisted translation. In human evaluation, using a real translation task scenario, translations took more time (up to a 17%) but less work (up to 50% less keystrokes) to type, compared to translating without suggestions. During the evaluation, translators with different levels of acceptance of the system were used. The users felt they worked faster, even if data proved the opposite. The users also thought suggestions were useful, although using them facilitated a more literal, monotonic, word by word translation.

Using the feedback of the first user study (Langlais et al., 2000), an updated TransType prototype implemented some upgrades, such as using lexicons that let the user add multi-word translations, and the interface was slightly improved.

TransType had a second stage, called TransType2 (Macklovitch, 2006), in which new features were implemented: the translation model was improved to use phrases, new languages were added to the tool, and the interface was overhauled using the feedback obtained during the second TransType user study (Langlais et al., 2002). Three new translation engines were developed for this project:

- RALI (Foster, 2000), a maximum-entropy minimum-divergence MT model developed in Canada for translating between English and French which proposes multiple suggestions for the next words.
- ITI (Cubel et al., 2003), developed in Spain, is a finite-state model that offers a single completion of the rest of the sentence, and can translate between English and French or English and Spanish.
- RWTH (Och et al., 2003), developed in Germany, is a statistical MT system that offers completions of the rest of the sentence and can translate between English and French, English and Spanish or English and German.

Also, users could select different domains for the MT system: technical manuals (trained with the Xerox corpus), European Community official documents or official reports of the debates of the House of Commons of Canada.[24] The interface also was modernized, adding new configuration options, such as the ability to change the assignment of some hotkeys. TransType 2 interface can be seen in Figure 1.2.

Evaluations (Macklovitch, 2006) were extended in time, so users could learn to use the system. Over 18 months, 5 rounds of user trials were conducted; the first two were mainly preparatory, and the later three were performed under approximately real working conditions. The third test let the users to choose between showing only one full sentence translation, or multiple shorter suggestions, with the testers preferring the former option. The tests showed that using the tool increased the productivity by $15 - 20\%$ (measured in words per minute) after the learning period. The main concern of the evaluators was that the system was unable to learn from their input: translators were used to the TM workflow, where new translations get added to the TM, and they can be offered in following translations; but the underlying MT

[24] These published reports are also known as Hansard: the collection of documents is usually referred as the Hansard corpus.

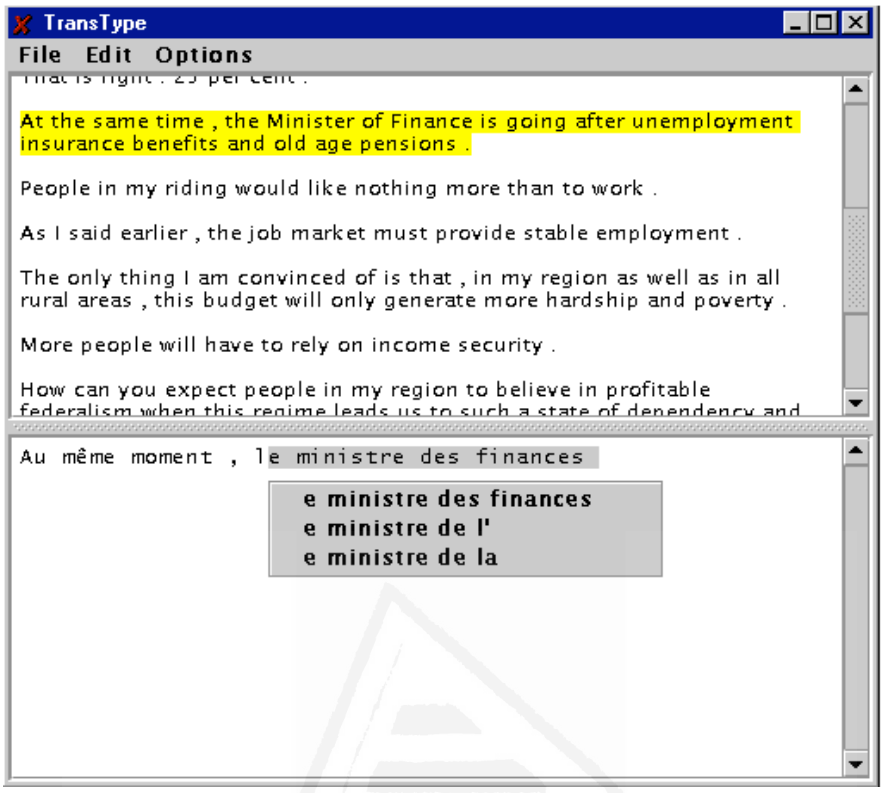


Figure 1.3: TransType2 interface. The top half of the screen contains the source segments, and the target is typed in the bottom half. The completions are given in a drop-down menu at the insertion point. The figure was taken from the paper by Langlais and Lapalme (2002).

engine of TransType 2 did not update with user actions, therefore forcing the translators to fix similar errors multiple times. The TransType2 project ended in 2005.

Interactive translation prediction based on phrase-based statistical machine translation: As MT engines advance and computer power increases, more complex models can be used for ITP. Caitra (Koehn, 2009b) is a web-based ITP tool that uses a modified version of the SMT engine Moses (Koehn et al., 2007) to generate and rank the suggestions.[25] Caitra is not only able to suggest words and phrases to continue the translation, but also shows a translation options table with word and phrase translations and their probabilities (similar, in concept, to the one in the Example 1.4), and it is able to operate in post-editing mode, where the alignment between the original SMT proposal and the post-edited sentence is shown, and show which parts of the source sentence are already translated in the final translation. Caitra also logged all the activity of the user so it can be analyzed later; Caitra used to be hosted by the University of Edinburgh and available for free,[26] therefore, great amounts of testing

[25]The modifications exposed some of the internal components of Moses, such as the decoder graph or the phrase table, in order to use them to generate the suggestions.

[26]As 1st May 2017, the tool is not available.

Figure 1.4: Part of Caitra interface. The top half of the image corresponds to the ITP interface: the source sentence appears over a textbox where the user can type the translation. The most probable continuation appears under the textbox, to the right of the "enter" button. The bottom half shows the translation options table, with the words and phrases that are most likely to be translation of the covered source subsegments; e.g., "the magnificent" is a highly likely translation for *"le magnifique"*. The user can directly click on the words and phrases to insert them into the translation. The figure was taken from the paper by Koehn (2009b).

data were available to accurately evaluate the system. Caitra was compared to post-editing, achieving a similar performance (Koehn and Haddow, 2009).

Caitra also recovered the initial idea of MIND: using monolingual users to improve the translation quality; but, rather than using a human that understands the source language to answer questions about the source sentence, it uses a human that understands the target language to edit the output of the SMT engine using the information shown in the translation options table. The study (Koehn, 2010a) shows that monolingual editors are able to often produce correct translations, reaching the performance of professional bilingual translators if the monolingual translator has good language skills and an excellent understanding of the domain. Caitra interface can be seen in Figure 1.4

Thot (Ortiz-Martínez and Casacuberta, 2014) is a log-linear phrase-based SMT model that achieves a performance close to the one of Moses. Thot was the only ITP system in active development during the development of this dissertation (between the years 2013 and 2017). Therefore, Thot will be used as a state-of-the-art ITP system to compare with.

The main differences between Thot and Moses are:

- Both Thot and Moses implement online learning as a way to quickly incorporate new data into already trained models, but the granularity is different: Moses online learning techniques are suited for processing large blocks of data, but Thot techniques are suited to update the model on a sentence-by-sentence basis; user feedback is added to the

model as soon as possible, in order to minimize the number of times the user has to correct the same mistake twice in the same session.

- Thot can operate as an ITP engine if a prefix is provided:[27] the best path according to the word graph[28] constrained by the typed prefix is offered as a suggestion; an error correction algorithm is used to align the given prefix if part of it is not present in the word graph, either because the given translation is not possible with the current model.[29]

Both Caitra/Moses and Thot were integrated into the CAT workbench CASMACAT (González-Rubio et al., 2012), that offers ITP along common CAT functions such as post-editing, quality estimation and TM. Thot also offers sentence-by-sentence online learning that adapts the underlying statistical model to the style of the user. CASMACAT also tested other approaches to translation, such as using electronic pens for post-editing (Alabau and Casacuberta, 2012). CASMACAT ITP interface can be seen in Figure 1.5.

Neural interactive translation prediction As explained earlier, a new kind of corpus-based SMT that uses deep-learning techniques to translate has recently appeared in the bibliography, neural machine translation. The decoder produces the n-th word of the target sentence T using the intermediate state, the attention model, and the already translated prefix of T; therefore, using an arbitrary prefix for T (for example, the prefix typed by the user) is rather trivial. Currently, the main drawback of using NMT are that the models are computationally expensive to use, requiring specialized, fairly expensive hardware; a common approach to reduce the computational cost is to reduce the size of the vocabularies in order to provide suggestions in real time. The process of training the models is also extremely expensive, requiring several weeks of computation using the specialized hardware.

Black-box ITP The previous ITP approaches follow a strategy that will be referred to as glass-box:[30] they use an embedded SMT engine to generate the suggestions that are offered to the user as the translation is being carried out. The SMT engine is not only queried for translations, but also different parts of the internals that are part of the translation process are

[27] Moses was also able to continue a translation using the `-continue-partial-translation` option, but the functionality got removed in May 2013 `http://www.statmt.org/moses/?n=Advanced.Obsolete#ntoc4`.

[28] A word graph is a structure that is very similar to the representation of the decoder algorithm found in the Example 1.5; word and phrase translations are ordered in an acyclic directed graph, and the words or phrase translations are weighted with their translation probability (according to the log-linear model).

[29] For example, if there is an unknown word in the translation, or if the translation had a very low probability and it was pruned during the expansion of the word graph.

[30] Making a parallelism with the black/glass boxes found in other disciplines, namely, electronics. A black box is a closed, opaque system with well defined inputs and outputs: what happens inside cannot be perceived, only the outputs can be observed. Conversely, a glass box is also a closed system with inputs and outputs, but it is transparent: the inner components and how they evolve as the inputs are being processed can be observed and studied along with the output value.

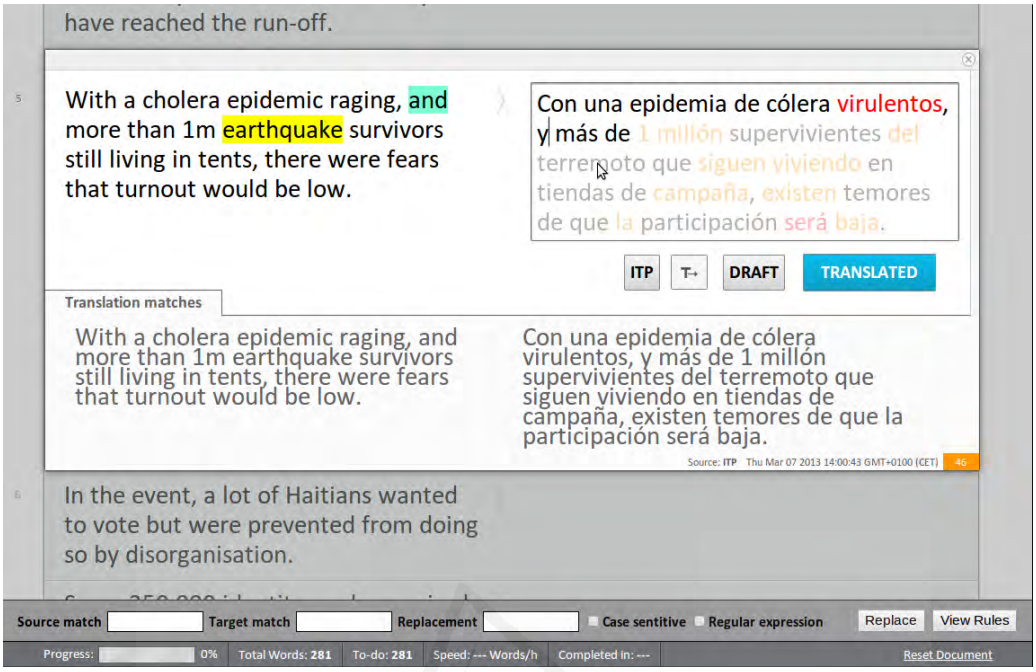


Figure 1.5: CASMACAT interface. The top half of the highlighted part of the image contains the source sentence (to the left) and the textbox where the user types the translation (to the right). The text to the right of the caret is the translation completion suggestion. The bottom half of the highlighted area shows a a fuzzy match extracted from a translation memory. Outside of the highlighted area the rest of segments to translate can be seen, along with some statistics. The figure was taken from the paper by Alabau et al. (2013).

further queried for obtaining more information, such as translation alternatives or different scores and probabilities. This way, as much of the information contained in the SMT model as possibles is extracted and used to generate the ITP suggestions. Thus, the glass-box ITP model is tightly coupled to the SMT engine, and cannot work with any other kind of bilingual source (unless it can provide the same information as the embedded one), and inherits the SMT reliance on vast amounts of parallel corpora. As previously explained, neural machine translation only needs to be slightly modified to provide ITP; yet, it keeps the reliance on vast amounts of parallel corpora, and it cannot be used for ITP if the NMT decoder cannot be modified to take an arbitrary prefix.

To address this limitation, an alternative approach was envisaged: to use MT (and eventually any kind of bilingual resource) as a black box to provide ITP suggestions, that is, just by translating segments from source to target language. The source segment is split into all the possible subsegments up to a given length (L), then the subsegments are translated by using any bilingual resource capable of providing one[31] or more translations. This approach is based in two central ideas:

[31]The bilingual resource may also fail to provide a translation for some subsegment, e.g., a dictionary will most likely be able to only translate single words in the canonical form (e.g. infinitive for verbs) and some common phrases, but will provide no translation for most subsegments. This scenario is not desirable, as the

- MT will most certainly not have enough quality to translate full sentences flawlessly, but small segments with limited context can still be used to assemble a good translation; thus, the user can be assisted by translating small segments of the original sentence and offering them as the translation is carried out.
- As the internal information contained in the bilingual resource cannot be accessed, all the overlapping n-grams (or subsegments) of the sentence are translated using the bilingual resource; in this way, more information can be extracted from the bilingual resource by translating the same word under different contexts (the surrounding words in the subsegment).

As the ITP system does not have the responsibility of providing bilingual data, it does not have to rely in one particular kind of resource; this permits the integration of any kind of bilingual resource the user has access to, such as:

- Bilingual resources that do not provide any kind of additional information needed to assess how good the suggestions are, such as dictionaries or RBMT; this is specially interesting for language pairs that do not have enough available corpus resources to train a SMT system that is good enough to be helpful to the user, but have other types of bilingual resources available.
- SMT systems that contain the information needed, but are distributed as proprietary, closed-source software. Therefore, even when there is a SMT model available to be used for ITP, the software cannot be modified to permit the access to the internal information needed for ITP. This case stands for NMT: if the decoder cannot be modified to take an arbitrary prefix, it cannot be used for ITP.
- Translation-as-a-service products (a special case of the previous category), where the translation engine is not accessible or even known, and one can only query it through a web service.

There are other differences between glass and black-box ITP:

- Suggestion availability:
  - In glass-box ITP, the system has to generate a new translation continuation for every keystroke; this means that, for reading the suggestion, the user has to stop typing[32] and wait for the translation to be performed, hampering the performance in high-latency scenarios.
  - In black-box ITP, the translation can be performed as soon as possible, as it does not depend on the prefix: the impact of a high latency bilingual resource may be minimized by pretranslating the segments and keeping them in a cache before the beginning of the translation work. Even if the work is to be started immediately and a slow bilingual resource is used, the subsegment translation task is performed

system will not be able to provide suggestions for the translation of a word or subsegment, but it is not a critical failure: the user will still get suggestions coming from the translation of other subsegments.

[32] Otherwise, the system would get new prefixes and would be forced to generate new suggestions

subsegment by subsegment, so the user can start working immediately and more and more suggestions will be available as time goes on.

- Translation work:
  - In glass-box ITP, an unknown amount of translation work is being performed, as every time the prefix changes new suggestions may have to be produced, which requires decoding. This can have a high computational cost.
  - In black-box ITP, a fixed amount of words are being translated, roughly equal to the length of the source segment times the maximum subsegment length.[33] As some subsegments will appear multiple times in the corpus,[34] these translations can be reused to further reduce the translation cost.
- Computational cost:
  - In glass-box ITP, the SMT engine is queried with every change in the prefix. This may lead to a significant computational load. For PBMT-based ITP, cost also increases with the length of the sentence being translated (Knowles and Koehn, 2016) and with the length of the mismatched prefix[35] (Koehn, 2016). In Chapter 3, the black-box approach is compared to a glass-box ITP implementation that showed the same trend.
  - In black-box ITP, the translation process is offloaded to the bilingual resource; once the segments are translated, the cost of selecting which suggestions are to be shown is fairly low: in the black-box experiments, no approach[36] took more than 100 ms to show the suggestions to the user, with the median close to 10 ms, not including translation time.[37] As the black-box approach does not keep track of the position, the system incurs in no penalty if suggestions are not accepted or if the prefix has unknown words.

Forecat To test the novel approach presented in this dissertation, Forecat, a free and open-source ITP system, was implemented. Forecat has similar functionality to the one used by other approaches: the source text is shown along the translation, and suggestions are offered as the user types the translation and restrained by the typed prefix; suggestions can be accepted using the cursor movement keys, a hotkey, or the mouse. Forecat also includes an additional

[33] As previously explained, the bilingual resources may not provide good enough translations for long segments, but the translations for the short segments may be useful. Values for the maximum size of the subsegment will be explored in the next Chapter.

[34] Such as the subsegment "the", that is expected to appear often in most English texts.

[35] That is, the length of the sequence of words that the underlying SMT model cannot generate.

[36] Neither the heuristic used in Chapter 2 nor the machine learning approach used in Chapter 3; the neural network used in Chapter 3 uses less than 10 MB of RAM and it is capable of scoring 10^5 suggestions per second.

[37] The translation time varies with the kind of bilingual resources used and the computer performing the task; for example, Apertium is able to translate several thousand words per second, the Moses models used in this dissertation could translate hundreds of words per second, and an NMT system operating on a CPU can only translate dozens of words per second.

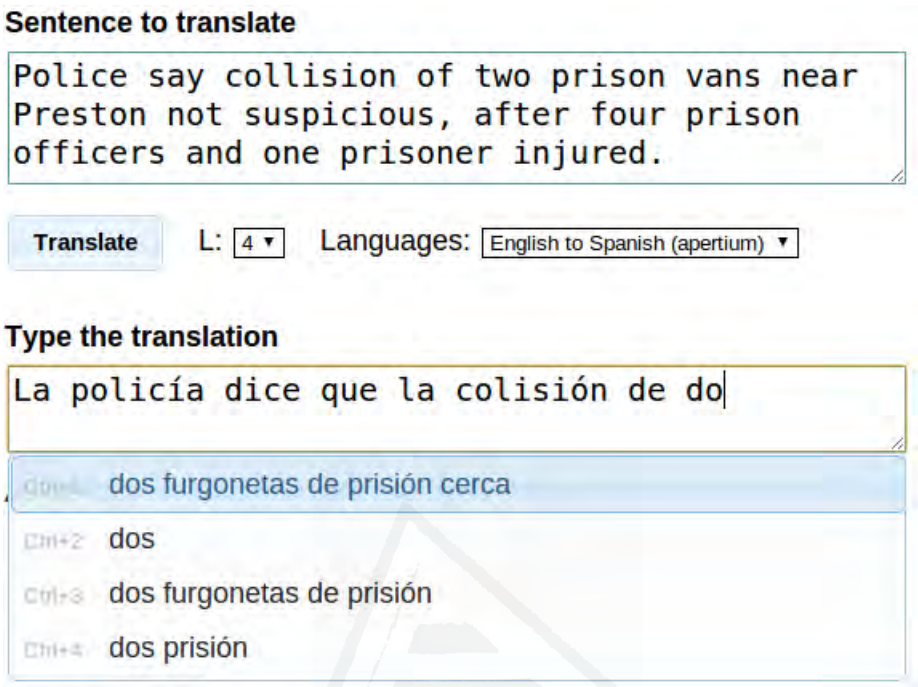


Figure 1.6: Forecat web interface. The top textbox has the source sentence, and the bottom one is where the user types the translation. The translate button is used to generate the subsegment translations. The value of L can be selected in this interface, but the value of M is fixed to 4. The dropdown list lets the user select the source and target languages, and the bilingual resource that will be used to translate. The suggestions appear as a dropdown list under the target sentence.

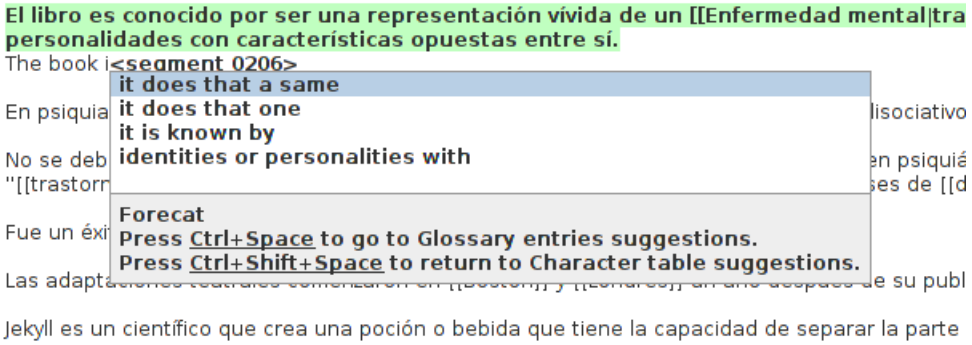

El libro es conocido por ser una representación vívida de un [[Enfermedad mental|tra
personalidades con características opuestas entre sí.
The book i<segment 0206>
it does that a same
it does that one
it is known by
identities or personalities with
Forecat
Press Ctrl+Space to go to Glossary entries suggestions.
Press Ctrl+Shift+Space to return to Character table suggestions.
Jekyll es un científico que crea una poción o bebida que tiene la capacidad de separar la parte

Figure 1.7: Forecat OmegaT interface. The suggestions appear as part of the autocomplete component of OmegaT, that also includes other features such as glossaries and character tables.

	Translations	Languages	Interface
TransType	Ad-hoc (Updated IBM models)	English–French	Desktop
TransType2	Ad-hoc (RALI, ITI, RWTH)	English–French, English–Spanish, English–German	Desktop
Caitra	Phrase-based SMT (Moses)	Any language pair with enough available corpora	Web interface
Thot	Phrase-based SMT	Any language pair with enough available corpora	Web interface
Forecat	Any accessible bilingual resource	Any language pair with an available bilingual resource	Web interface, web services, OmegaT plugin

Table 1.1: Comparison of the different approaches.

feature that lets the user use a different key to select the prefix of a suggestion word by word by using a key, rather than using the whole suggestion. Forecat can also be used via three different interfaces: as a webpage, as web services (that let any webpage to integrate the ITP functionality), and as a plugin for the TM workbench OmegaT.[38] The web interface of Forecat can be seen in Figure 1.6, and the OmegaT plugin interface can be seen in Figure 1.7. A table comparing the major features of each described system can be seen in Table 1.1.

[38] http://www.omegat.org/

Chapter 2

A heuristic approach to black-box ITP

Different ITP approaches have been described in the previous chapter, but most of them share a common trait: they follow a glass-box strategy, and rely on the inner workings of a tightly coupled SMT system; therefore, they inherit the main corpus-based MT limitation, namely, the reliance on extensive parallel corpora. A novel method has been developed as part of this dissertation, which, unlike the previously described approaches, follows a black-box strategy: suggestions are obtained by translating all the contiguous overlapping n-grams (or subsegments) up to a given value of n by means of any available bilingual resource capable of delivering one or more translations into the target language. These bilingual resources may be MT systems, but also translation memories, dictionaries, catalogues of bilingual phrases, or any combination of them. The translated subsegments are then offered as the translator carries on with the translation task.

In this chapter, the initial black-box ITP idea will be described in depth in Section 2.1. Later, the automatic evaluation procedure and the metrics used to measure the performance of the proposed strategies will be described in Section 2.2, and a language model based monolingual predictor that will be used as a baseline will be described in Section 2.3. Afterwards, the initial heuristic proposed to rank and select which suggestions should be proposed will be detailed along with the justification of the strategy used, and the results of the automatic evaluation in Section 2.4 and the results of a preliminary user study in Section 2.5. Finally, the shortcomings of the heuristic will be described in Section 2.6.

2.1. Starting point

As explained in the previous chapter, the objective is to build an ITP system without the main limitation of glass-box ITP: the reliance on the tightly coupled corpus-based MT system. Instead, the aim is to create a system that is able to provide suggestions using any bilingual resource.

Example 2.1

Source sentence	this	studio	is	spacious
Subsegments of length 1	este	estudio	es	espacioso
Subsegments of length 2	este estudio			
		estudio es		
			es amplio	
Subsegments of length 3	este estudio está			
		estudio es amplio		

Translation of all the possible subsegments up to length $L = 3$ of the English sentence *This studio is spacious* to Spanish. In this example, the subsegments are translated using only one resource that provides one translation per subsegment. The resourced used to translate this sentences (Google Translate in December 2016) made a different lexical selection when translating *spacious* as a single word (translated as *espacioso*), choosing the synonym *amplio* when translating *is spacious* and *studio is spacious*. The sentence and suggestions have been lower-cased to simplify the example.

The first step of the process is to extract all the contiguous overlapping n-grams (or subsegments), for every $n \leq L$. The actual value of L depends on multiple factors, such as:

- The kind of resources being used. For example, a dictionary may only be able to translate single words and some short multi-word units, while a MT system will not have any trouble translating longer segments.
- The source and target languages. Obtaining adequate translations for long segments between closely related languages (such as Spanish and Catalan) is usually easier than translating between less related languages (such as Arabic and English).
- The domain of the translation task. Constrained domains with limited vocabularies are easier to translate, such as instruction leaflets, and others are harder, such as literature.

These segments are then translated by using any bilingual resource capable of providing one or more translations, as shown in Example 2.1. This creates a suggestion pool, where each suggestion is comprised of a target language subsegment that will be offered to the user, and the start and end of the source language of each subsegment.[1]

Then, as the user types the translation, the ITP system will offer suggestions to the user. On each keystroke, the system will offer between zero and M suggestions; the user can then accept any suggestion deemed fit to continue the translation, or ignore them and keep on typing. The main criterion when selecting the suggestion is that only those suggestions that have as prefix the last partially written word of the translation can be offered to the user.

[1] If a source subsegment is repeated, or different source subsegments are translated into the same target subsegment, the suggestion will have multiple start-end pairs.

Example 2.2

S	"this studio is spacious"
T	"este estudio es amplio"
W_1^1	e
P_O^S	este estudio está **este** estudio es amplio estudio
W_2^0 P_O^S	e**ste_**
W_2^1	e*ste* e
P_O^S	**estudio es amplio** estudio es amplio es
W_4^6	e**ste** e**studio es amplio**

Example of an ITP session translating from English to Spanish, with the maximum number of suggestions $M = 4$ and maximum subsegment length $L = 3$. The bold suggestions are the ones selected because they were the most useful for advancing in the translation; the bolded parts of W_n^mhave been inserted using a suggestion. In this example, the user types the first letter of the translation (e), and gets four suggestions; the second one is selected. Then, the user types another letter (e) and gets another four suggestions; the first one is selected, and the translation is finished. In the second prefix (W_2^0), a trailing space has been replaced with an underscore for visibility; no underscore has been typed or inserted via suggestion in this example.

Still, only few suggestions can be shown to the user; if too many suggestions are shown, the user will lose too much time reading them and will start to ignore them. For this mean, a technique to rank and select the M most promising suggestions is needed. Two strategies have been devised: a simple heuristic approach that will be presented in Section 2.4, and a machine-learning-based approach that will be described in Chapter 3. Example 2.2 shows a full ITP session.

2.1.1. Formal definition

To formalize this process,

- $|x|$: the number of words in the sequence of words x
- $|x|_c$: the number of characters in the sequence of words x
- $S = S_1 ... S_{|S|}$: the sentence in source language; S_i is the i-th token in S

- $T = T_1...T_{|T|}$: the sentence in target language; T_i is the i-th token in T.
- The currently typed prefix of T; $n-1$ tokens and the first m characters of the n-th token of T have already been written:

$$W_n^m = \begin{cases} W_{n-1} w_n^m & \text{if } n > 1 \\ w_n^m & \text{otherwise} \end{cases}$$

- $w_n^m = T_n[1:m]$: the first m characters of the token T_n.
- P^S : the suggestion set, all the subsegments of S up to a maximum length L, translated by means of any bilingual resource
- $p \in P^S,\ p = (s_p, t_p, b_p, e_p)$: a suggestion composed of
  - $s_p = S_{b_p}...S_{e_p}$: the source-language segment of the suggestion
  - b_p: the starting position of s_p in S
  - e_p: the ending position of s_p in S
  - t_p: the translation of s_p, that is, the text that may be offered to the user as a suggestion
- $P_C^S(W_n^m) = \{p \in P^S \mid w_n^m \in \text{Prefix}(t_p)\}$: the compatible suggestion subset, that is, the suggestions that are compatible with the currently typed prefix; $\text{Prefix}(t_p)$ is the set of prefixes of t_p. As previously discussed, no suggestions are compatible with an empty prefix: $P_C^S(W_n^0) = \emptyset$.
- $g(p, W_n^m) \in [0,1]$: The goodness of a suggestion for the current prefix, an estimate of how likely is that this suggestion will be selected by the user.
- The offered suggestions list, an ordered list with the suggestions from $P_C^S(W_n^m)$ that contains up to M suggestions sorted by their goodness ($g(p, W_n^m)$) score:

$$\begin{aligned} P_O^S(W_n^m, M) = &[p_1...p_i \mid i \leq M,\ p_j, p_k \in P_C^S(W_n^m), \\ &\forall\, j,k\ 1 \leq j < k \leq M,\ g(p_j, W_n^m) \geq g(p_k, W_n^m)] \end{aligned}$$

2.2. Evaluation of the approach

2.2.1. Automatic evaluation algorithm

Extensive human evaluation is both slow and expensive; therefore, to evaluate the performance of the developed approaches, an automatic evaluation procedure was used. This

automatic evaluation procedure is similar to the one used in the bibliography, that was initially defined in the TransType project (Foster et al., 1997; Langlais et al., 2000). The procedure emulates the behaviour of a translator who reads the source sentence and mentally builds the translation of the sentence;[2] then, proceeds to exactly "type" this translation. For every keystroke, the system offers up to M suggestions: the emulated user reads every one of them and either chooses the longest suggestion that exactly matches the translation being typed or ignores the suggestions and "types" the next character, until the translation is completed. Only suggestions with full-word translations will be accepted: if the word of T currently being translated, w_n^m, is *thesaurus*, a suggestion *the* will not be accepted. The output of this automatic procedure is a sequence of keystrokes and accepted suggestions.

A slight difference between the black-box approach and the glass-box approaches in the state of the art is that the black-box approach does not offer suggestions with an empty prefix or after typing space ($w_n^0 =$""), as all the suggestions in P^S would be in the compatible suggestions set P_C^S.

The translation a human translator is going to type cannot be known beforehand, but the automatic evaluation system gets a translation reference provided; therefore, the viable suggestions, those that can be used to advance in the translation being currently typed, can be known. The longest suggestion in the intersection between the viable suggestion set and the offered suggestion set is the winning suggestion: choosing the winning suggestion is the (locally) optimal action to minimize the effort.

To formalize these sets,

- The viable suggestions set, the suggestions that can be used to advance in the translation being currently typed:

$$P_V^S(W_n^m) = \{p \in P_C^S(W_n^m) \mid \forall\, k\,, 1 \leq k \leq |t_p|,\ t_p[k] = T_{n+k}\}:$$

- $P_{OV}^S(W_n^m, M) = P_O^S(W_n^m, M) \cap P_V^S(W_n^m)$: the intersection between the viable suggestions and the offered suggestions.

- The winning suggestion at each position:

$$\phi^S(W_n^m, M) = \begin{cases} \varnothing & \text{if } P_{OV}^S(W_n^m, M) = \varnothing \\ \operatorname{argmax}_{p \in P_{OV}^S(W_n^m, M)}(|p|) & \text{otherwise} \end{cases}$$

The output of the automatic evaluation is a sequence of actions that is comprised of keystrokes (typing one character when no viable suggestions are offered) and suggestions that belong to the winning suggestions set; yet, not all the combinations of winning suggestions are valid: for example, in Example 2.2, the winning suggestion of W_2^1 and W_2^2 would be the

[2]When performing the automatic evaluation of an approach, a translation reference is provided to the system; the system "types" exactly this reference. This is different from the behaviour of a human translator, who may plan an initial translation, but then modify it for different reasons; for example, a suggestion with a translation alternative that fits better the sentence but the translator did not think about.

same, *estudio es amplio*, but, if the suggestion at W_2^1 is chosen, then no suggestions will be offered for W_2^2, as accepting the suggestion advances the translation directly to W_4^6. Those suggestions that are part of the sequence of actions generated by the automatic evaluation are in $\psi^S(M)$.

2.2.2. Metrics

Similarly as in the paper by (Langlais et al., 2000), the keystroke ratio (KSR), the ratio between the number of keystrokes and the length in characters of the translation,[3] is used to measure the performance of this system. The following actions can be used to type the translation:

- Typing a character has a cost of 1 keystroke.
- Accepting a suggestion has a cost of 1 keystroke, no matter its rank on the suggestion list. The suggestion can be accepted either by
  - Using `Alt`+p, being p the position of the suggestion in the list, with a cost of 1. It is assumed that the automatic evaluation system uses this option to choose the winning suggestion among the offered suggestions.
  - Using the mouse, with a cost of 1.[4]
  - Using the arrow keys and the enter key, with a cost of 1 per arrow key pressed plus 1 for the enter key.

The KSR measures how much mechanical effort is saved, but it does not measure the cognitive effort needed to read the suggestions. It also does not measure other side effects, like the user spending less time checking the parts of the final translation that were inserted by accepting a suggestion, or whether the translation improved in quality when the user was offered a suggestion with a better translation that the planned one; but these effects are hard to estimate during the automatic evaluation. The KSR takes a value in $\mathbb{R}^+$; a KSR of 0 is impossible, as at least one keystroke is needed to accept a suggestion. If KSR ≥ 1, it means the user made some mistakes while typing the translation; still, the user can make mistakes and obtain KSR < 1 if suggestions are used. During the automatic evaluation, the maximum value for the KSR will be 1, as the system will make no mistakes while "typing" the translation.

In order to represent the cognitive effort needed to read the suggestions, a second metric is used: the accepted suggestion ratio (ASR), the quotient of the number of times where

[3] Either the provided reference to the automatic evaluation system or the translation that the user typed.

[4] Using the mouse is costly: the cost of moving your hands from the keyboard to the mouse (0.40 *s*), pointing with the mouse (0.80 *s* to 1.50 *s* with an average of 1.10 *s*), performing the click (0.20 *s*) and moving your hand back to the keyboard (0.40 *s*) is 2.10 *s* on average. A good typist would have typed 3 words in that span of time; an average one, 1.4 words (Card et al., 1980). But most of these actions can be performed along the process of reading and evaluating the suggestions. No empirical evidence about the actual time cost of using the mouse compared to using the keyboard exists, therefore a cost of 1 will be used.

one of the shown suggestions was used divided by the amount of times in which at least one suggestion is shown. This can be seen as a user confidence measure: if the ASR is close to zero, it means that too many incompatible suggestions are being shown, and the user will eventually stop looking at them; conversely, if the ASR close to one, it means the suggestions are often useful and the user confidence will increase. The ASR can take values in $[0, 1]$: a value of 0 means no suggestion was used; a value of 1 means that the user selected a suggestion every time the system offered a suggestion list.

In Example 2.2, the final KSR is $5/22 = 0.23$ and the final ASR is $2/2 = 1$ (a suggestion was used every time suggestions were offered).

2.3. A monolingual predictor baseline

Monolingual text prediction is commonplace nowadays. Most modern mobile phones include an on-screen tactile keyboard to input text, and, due to the increased difficulty in typing (as they lack feedback when typing and the size of the keys is small), usually include two functionalities that help users to type: an automatic orthographic corrector and a text predictor that offers the most likely word that will be typed.

The second feature offers a kind of assistance that is identical to ITP, but does not use bilingual information at all; instead, the predictions are based on a language model (previously explained in Section 1.1.2.1): the system iterates over a dictionary with common words, computes how likely is each word to follow the currently typed segment, and offers an n-best list with the most likely words to the user. For example, if the user has typed *Kill two birds with one*, the word *stone* will most likely be proposed next. Then, the user can choose one of the predictions, or just ignore them and keep typing.

On the one hand, monolingual text prediction based on language models has one great advantage: it is possible to train a language model of almost any given language, as monolingual corpora are readily available; the language model can be trained with any collection of texts, such as books, webpages, etc.[5] On the other hand, it is expected that monolingual text prediction obtains worse results than ITP, as ITP is using bilingual information to generate the suggestions.

For these reasons, a monolingual text predictor based on a language model was used as a baseline for ITP. The same automatic evaluation procedure explained in Section 2.2.1 and the same metrics described in Section 2.2.2 (KSR and ASR) were used. The monolingual text predictor generated only the one-word suggestion that was the most likely continuation for the currently typed prefix, constrained by the last partially typed word, that is, if the user typed *Kill two birds with one st*, the model could have suggested any word with *st* as a prefix, such as *stone* or *street*.[6] The model always suggests one word, even with a blank prefix. Only

[5] Some monolignual text predictions systems also update the model according to what the user types.

[6] This restriction is similar to the one used to initially filter the suggestions of the black-box ITP model.

the top 1 000, 000 tokens of each language model will be used to generate predictions: this is done both to reduce the cost of the process, and to exclude less frequent tokens that were seen too few times (usually just once) to be relevant.

KenLM (Heafield, 2011) was used to train and query the language models.

2.4. A simple heuristic to rank the suggestions

2.4.1. Initial hypothesis

When the user types W_n^m, the system offers the suggestion list $P_O^S(W_n^m, M)$: in the example shown in Example 2.1, with $w_n^1 = e$ all the elements in P^S will be in P_C^S; however, users are not expected to tolerate long lists of suggestions: a good criterion for ordering the suggestions ($g(p, W_n^m)$) is needed in order to keep a low value of M and not to leave many viable suggestions out. Therefore, a strategy for ranking and selecting which suggestions are shown to the user is needed.

As a basis for defining a strategy, four hypothesis were stated:

- Shorter suggestions are more likely to be viable but save little effort when used.
- Longer suggestions are seldom viable, but save lots of effort when used.
- Translation is mainly monotonous and the i-th word of T is likely to be a translation of the i-th word of S (or a close one).[7]
- Suggestions that take origin near the part of the sentence being translated are more viable than those further away; that is, those suggestions whose b_p is closer to the word currently being translated k.

2.4.2. Corpora and resources

The experiments performed in this section used the first 15 000 sentences of the Corpus *DGT-TM*[8] for English–Spanish evaluation, and 15 000 sentences from the Corpus *El Periódico de Catalunya* for Catalan–Spanish evaluation. The rest of each corpus was used to train a language model for the monolingual text predictor; 1 912 074 sentences for English–Spanish and 799 954 sentences for Catalan–Spanish.

[7] Even when this hypothesis may look counter-intuitive, the heuristic derived from it has proved to work for distant language pairs such as English–Arabic and English–Simplified Chinese.

[8] See Appendix I for a description of each corpus.

As a bilingual resource, the RBMT platform Apertium[9] (Forcada et al., 2011) was used, as Apertium has models for all four translation directions.

All the experiments in the section will use these corpora and resources, unless explicitly noted. All the differences between the values shown in the figures in this section are statistically significant, calculated using bootstrap resampling (Koehn, 2004) and 1000 iterations with $p = 0.05$, unless explicitly noted.

2.4.3. Proof of the hypotheses

In order to support the first and second hypotheses, different values of l, the subsegment length,[10] were tested. Results show that, for Catalan–Spanish translation (Figure 2.1), the KSR improved as the value of l increased up to $l = 3$, when compared to the previous value of l; for $l = 4$ and $l = 5$, the KSR is worse, but still beats the KSR for $l = 1$. The Apertium model for Catalan–Spanish is very accurate, as the languages are tightly related and the Apertium community has put a lot of effort into improving the model. The results support both hypotheses: segments with a length of 1 are accepted very often (more than half the occasions where at least one compatible suggestion was in $P^S_O(W^m_n, M)$), and segments with a length of 4 are seldom accepted ($P^S_O(W^m_n, M)$ only had a compatible suggestion 10% of times where it was not empty); however the segments of length 4 contributed to reduce the effort more than the segments of length 1.

The results for the same experiments for English–Spanish (Figure 2.2) show that Apertium produced worse translations than for Spanish–Catalan; in this case, $l = 1$ is the best value for both KSR and ASR, and both values get worse as l increases, and only $l = 1$ is statistically significantly different than the rest of values, that have a KSR close to 1. Even when the resource is not as good, some keystrokes may be saved; in this scenario, RBMT performs worse for ITP than in Spanish–Catalan.

A different experiment using $L = 4$ and unlimited M was also performed in order to support the third and fourth hypotheses:[11] every time there is a winning suggestion (that is, $p' \in \phi^S(W^m_n, M)$), the distance between where the suggestion originates ($b_{p'}$) and where is used (n) is computed and compared to the distances of the rest of suggestions in $P^S_C(W^m_n)$. As different suggestions can share the same origin position b_p, the metric used is the rank of the distance between where p is used and $b_{p'}$ in the set of distances; this means that, if $b_{p'} - n = 2$ and there are suggestions with $b_{p''} - n = 1$ and $b_{p'''} - n = -1$, the distance rank of p' is 3. Negative distances, that is, the suggestions that are used to the right of its origin position, take preference over positive distances.

[9] The Revision 79719 (25-06-2017) of the Apertium repository at `http://svn.code.sf.net/p/apertium/svn/trunk/` was used for the engine and linguistic data in these experiments.

[10] Not to be confused with the maximum subsegment length L, that uses subsegments with lengths within the $[1...L]$ range.

[11] The optimal value of L will be explored in the following Section 2.4.5; a value of $L = 4$ was deemed as ideal.

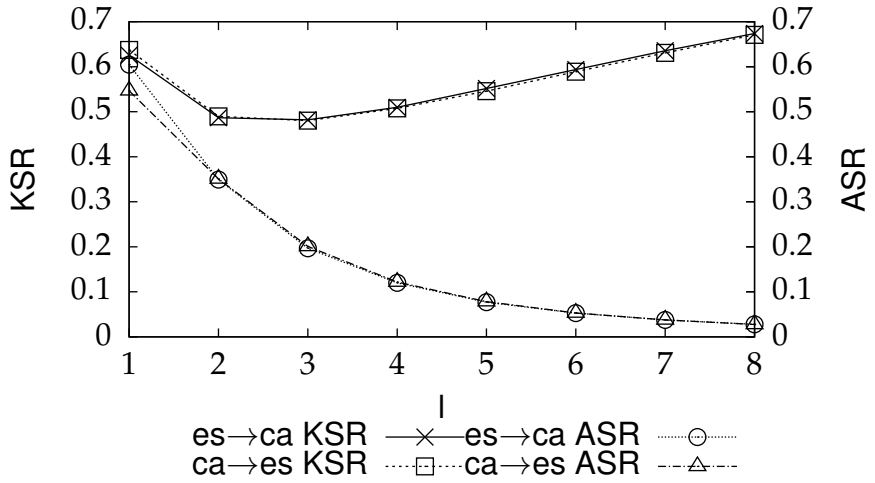


Figure 2.1: KSR and ASR for the different values of l for Catalan–Spanish translation using Apertium. The KSR for $l = 1$ is worse than the one for $l \in [2, 3, 4, 5]$, and improves up to $l = 3$.

The corpus *United Nations* [12] was used to train a Thot SMT model in this evaluation. The Thot SMT model will be used as the only bilingual resource.[13] The results can be seen in Figure 2.3; the vast majority of winning suggestions belong to the first distance rank, supporting the hypothesis that the closest suggestions have a higher probability of being viable. Figure 2.4 shows the actual distance between where the suggestion originates ($b_{p'}$) and where is used (n) rather than the distance rank; most of the winning and viable suggestions take origin in close positions, supporting the third and fourth hypothesis.

2.4.4. Description of the heuristic

A naïve heuristic based on the previously stated hypothesis (now supported by the data obtained during the experiments) was devised. It prioritizes those suggestions that come from nearby positions, and tries to offer both long suggestions (that save lots of keystrokes but are seldom viable) and short suggestions (that save a smaller amount of keystrokes but are viable more often).

When the user translates the k-th word of the source sentence, up to M suggestions are offered following this order:

[12] Corpus *DGT-TM* and *El Periódico* are not directly comparable, and Corpus *El Periódico* is Catalan–Spanish, languages that can be almost directly translated in a word-by-word basis. Using more distant pairs like English–Arabic (en–ar) and English–Simplified Chinese (en–zh) helps to better perceive the impact of this phenomena.

[13] Apertium offers no English–Simplified Chinese or English–Arabic; refer to Chapter 3 for a detailed description of the corpora used and the setup for Thot.

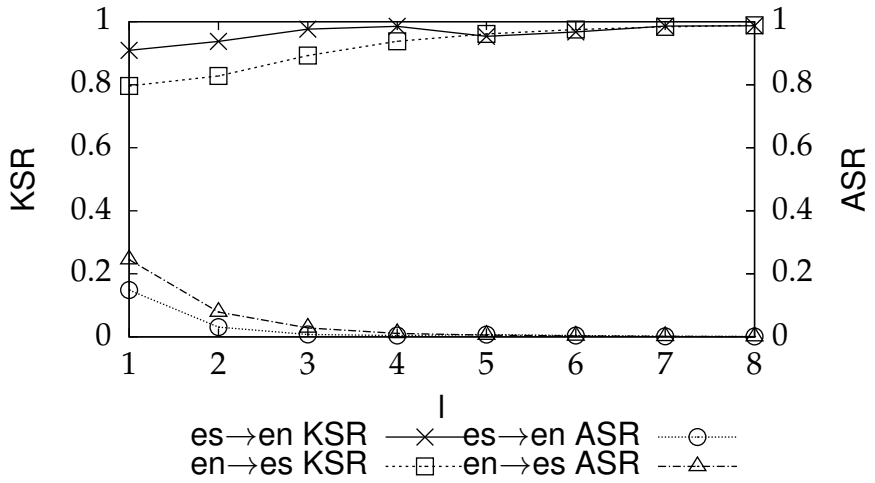


Figure 2.2: KSR and ASR for the different values of l for English–Spanish translation.

- The shortest and longest suggestions originated on the closest position[14] in $P_C^S(W_n^m)$ are offered.
- The shortest and longest suggestions originated on the next closest position are offered until there are no more different origin positions.
- The second shortest and second longest suggestions originated on the closest position are offered, and so on; that is, position takes precedence over length when ordering the suggestions.

An example of which suggestions would the heuristic show with $M = 4$ can be seen in the Example 2.3.

This naïve, preliminary approach achieved a good performance, but a more principled strategy based on machine learning was later devised, a machine learning based approach which is described in Chapter 3.

2.4.5. Optimizing the values of L and M

Using the evaluation method previously described, different values for L and M were tested. First, different values for the maximum subsegment length L were tested without limiting the maximum number of suggestions offered ($M = \infty$): the results can be seen in Figure 2.5 and 2.6; a fixed value for L ($L = 4$) was deemed as ideal for the rest of experiments, as the performance did not improve much past that point, and generating longer suggestions has a cost: to generate the suggestions, $L|S| - L(L-1)/2$ words have to be translated; most

[14] If there are suggestions in 2 different closest positions, the one to the right of k takes precedence over the one to the left.

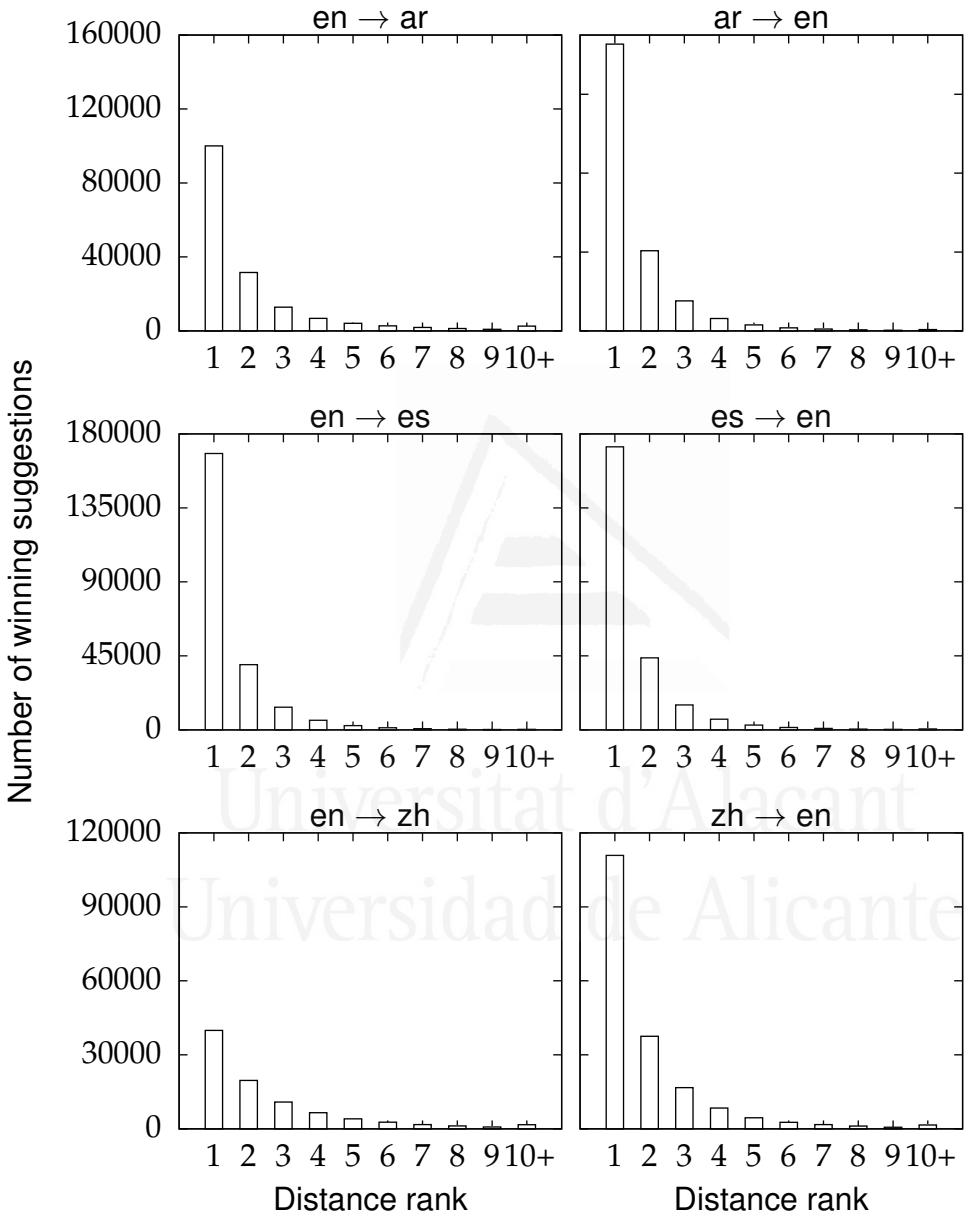


Figure 2.3: Distribution of winning suggestions in the different distance ranks. Most of the winning suggestions are in the closest distance rank, and the number of winning suggestions greatly decreases as the distance rank increases.

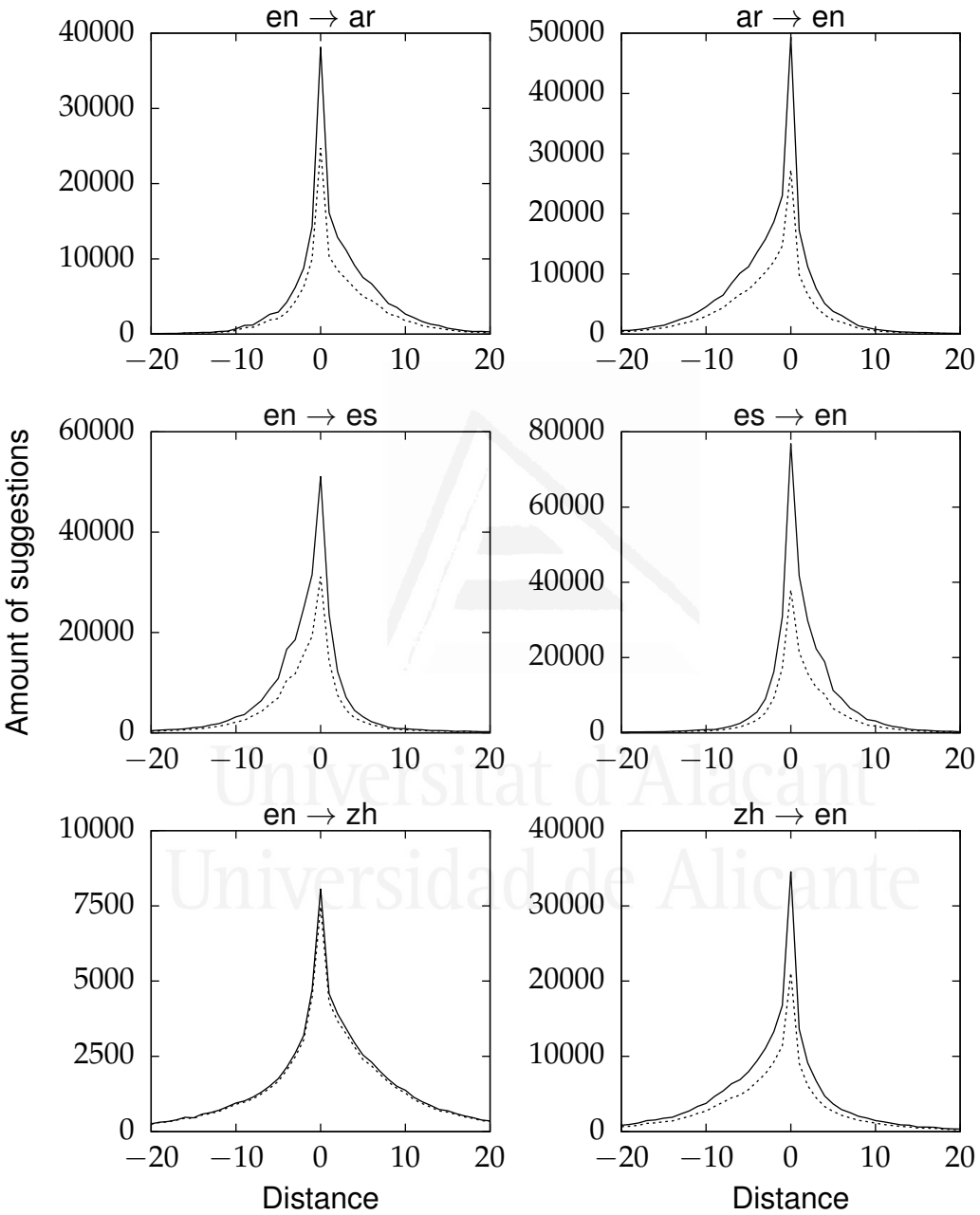


Figure 2.4: Distribution of winning (dashed line) and usable (solid line) suggestions as a function of distance. Most of the winning and viable suggestions come from close positions, and their number greatly decrease as the distance increases. The graphs only show distances (measured in words) in the $(-20, 20)$ range; a small amount of suggestions fall outside that range.

Example 2.3

This	studio	is	spacious
este	(**2nd**) estudio	(**4th**) es	espacioso
este estudio			
	estudio es		
		(**3rd**) es amplio	
este estudio está			
	(**1st**) estudio es amplio		

Suggestions offered for W_2^1 =*Este e* with $M = 4$, following the indicated order (from 1st to 4th).

translation as a service providers charge by the word, and further increasing the value of L would linearly increase the monetary cost of using the black-box approach.

Once the value for L was fixed, a similar batch of automatic experiments with different values of M was performed: the results can be seen in Figure 2.7 and Figure 2.8. Even with such a naïve heuristic, the performance achieved proves the viability of the black-box approach, saving at least 20% of the keystrokes when showing up to 4 suggestions. As with L, the performance does not improve much past $M = 4$, and more suggestions would imply a bigger cognitive load on the user.

The performance of the monolingual predictor described in baseline described in Section 2.3 can be seen in Figure 2.5 and 2.6. The English–Spanish monolingual predictor performs better than the black-box ITP approach, but, as already discussed, the performance of the bilingual resource used for this translation task is low; conversely, the Catalan–Spanish monolingual predictor performs worse than the black-box ITP approach; yet, the monolingual predictor can lead to saving around 30% keystrokes.

2.4.6. Greedy and optimal evaluation

The automatic evaluation procedure described is a greedy, left-to-right longest match strategy, meaning that it takes decisions locally (the longest suggestion in $\phi^S(W_n^m, M)$) and obtains a suboptimal answer. This is no different than how a human user would interact with the system, as the human is probably unable to predict the suggestions that will be offered in the future and can only locally decide which suggestion is the best to continue the translation. Nonetheless, it is necessary to measure how much performance is lost as a consequence of using the greedy strategy. For this mean, a strategy that is able to choose the optimal sequence of actions to minimize the KSR of the translation process by exhaustively exploring the solution space was used.

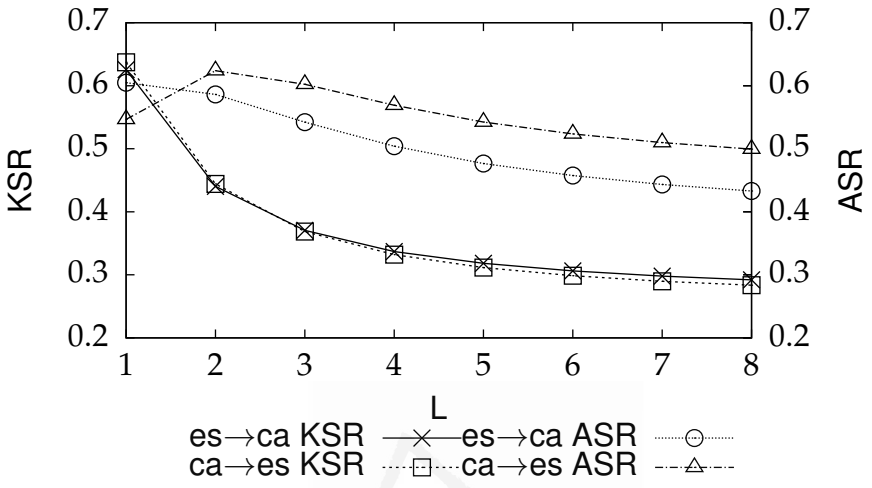


Figure 2.5: KSR and ASR for the different values of L for Catalan–Spanish.

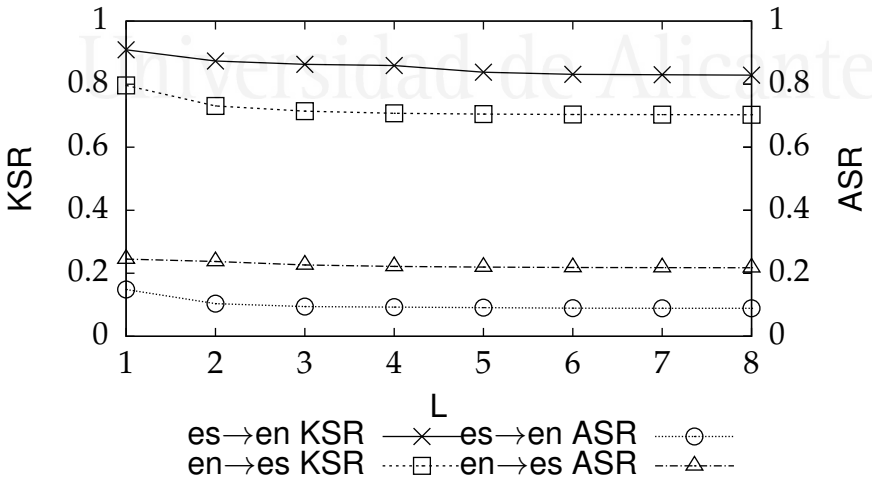


Figure 2.6: KSR and ASR for the different values of L for English–Spanish.

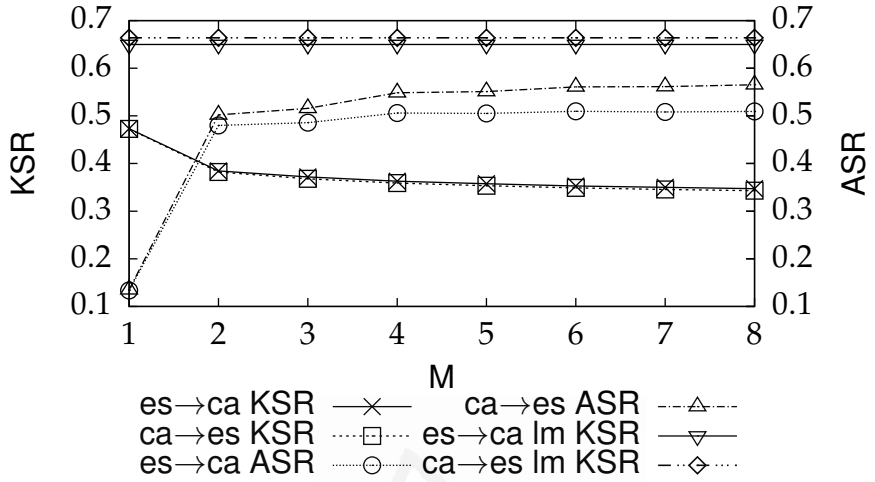


Figure 2.7: KSR and ASR for the different values of M for Catalan–Spanish. The monolingual predictor (tagged as lm KSR) always offers 1 suggestion.

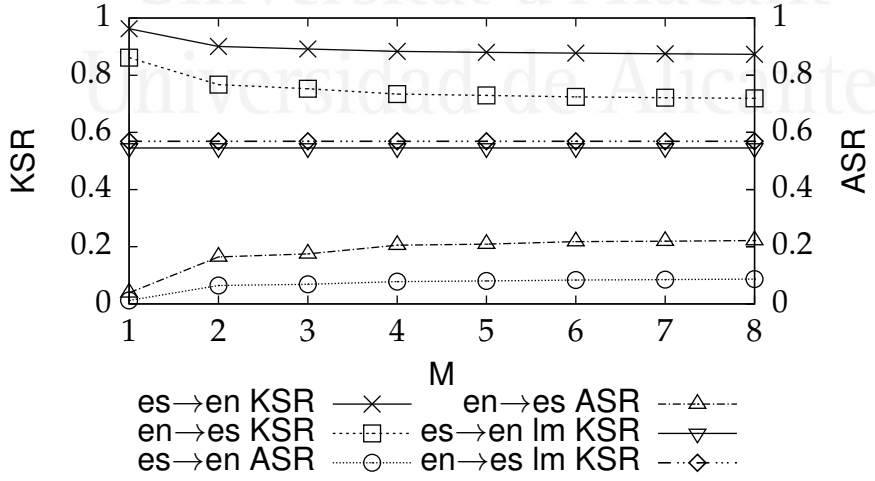


Figure 2.8: KSR and ASR for the different values of M for English–Spanish. The monolingual predictor (tagged as lm KSR) always offers 1 suggestion.

L	1	2	3	4	5	6	7
Greedy	0,809	0,745	0,727	0,720	0,717	0,715	0,715
Optimal	0,795	0,737	0,720	0,714	0,711	0,710	0,709

Table 2.1: KSR values of the greedy and optimal approaches, for the different values of L and with $M = \infty$, for English to Spanish translation.

S:	Optical-fibre cables for data transmission
Greedy:	
T:	Cables de fibra óptica para transmisión de datos
Optimal:	

Figure 2.9: An example of a sentence in which the greedy strategy gets a lower score than the optimal strategy, using $L = 4$, translating from English to Spanish. The brackets indicate that the text was generated by accepting a suggestion, and text not covered by brackets was typed. Apertium translates *for data transmission* as *para transmisión de datos*, but *data transmission* as *transmisión de dato*.

Rather than choosing to select a suggestion based only on the local optima, this optimal strategy analyses all the possible scenarios as they branch out depending on each taken action, finding the global optimal solution that minimizes the KSR. Results can be seen in Table 2.1.

The difference in KSR between the greedy and optimal evaluation comes from slight differences in how segments are translated. A synthetic scenario is illustrated in Example 2.4, where the greedy approach fails to achieve the optimal answer. Figure 2.9 contains the discussion of a real case that happened during the evaluation using Apertium.

The optimal approach has a much higher computational cost but the improvement in performance is rather small (around 1%) and, in most cases, non statistically significant; also, it makes the evaluation unrealistic, as human translators would have to correctly predict future suggestions to attain the same performance. Therefore, the greedy strategy will be used to automatically evaluate the different configurations.

2.4.7. In-domain and out-of-domain evaluation

Up to this point, the performance of the black-box ITP approach has only been tested with one particular bilingual resource, the RBMT system Apertium. To prove that the approach is feasible for any bilingual resource, and not only performs well under these particular conditions, it has to be tested using different bilingual resources. In this evaluation, the black-box ITP approach will be tested using two different SMT models, one trained with out-of-domain data and another with in-domain data. Moses (Koehn et al., 2007) will be used for training and querying the SMT model.

This experiment is inspired in the work by Haddow and Koehn (2012), where the influence of in-domain and out-of-domain corpora when training SMT systems is evaluated. The task

Example 2.4

Consider the following sentence in English and its translation in Spanish

S = 'the big red house'
T ='*la casa roja grande*'

and the suggestions (for this example, only these suggestions will be used)

Source sentence	the	big	red	house
Suggestions	el	grande rojo		la casa
	el grande		casa roja	

In this example, the word *casa* has morphological information that both *red* and *the* need to be translated into their correct surface forms in Spanish, *roja* and *la*; otherwise, the bilingual resource will default to the most probable surface form, that is, masculine singular, and would generate *rojo* and *el*, that are not viable. So, there are two possible action sequences:

- The user chooses *la casa*: the rest of the translation will have to be typed, as no suggestion will match.
- The user types *la*: the suggestion *casa roja* will be offered, and can be accepted to save keys during the translation.

Hence, ignoring the first suggestion leads to saving more keystrokes overall than using it.

will consist on English–Spanish and English–Czech translation, as those language pairs are, respectively, the ones that obtain the best and the worst BLEU (Papineni et al., 2002) scores in that work. The corpora used is the same as in that paper, and the size of each one is detailed in Table 2.2.

The test set is comprised of 15 000 sentences from Corpus *News Commentary*. Three different models were trained:

- An out-of-domain model, trained and tuned[15] with Corpus *Europarl*.
- An in-domain model, trained and tuned with Corpus *News Commentary*.

[15]The size of the tuning set is 2 000 sentences. The rest of the corpus was used as training set.

Languages	Corpus *Europarl*	Corpus *News Commentary*
English–Spanish	1 912 074	155 760
English–Czech	623 913	122 720

Table 2.2: Number of segments for each corpus and language pair.

BLEU	*Europarl*	*News Commentary*	*Mixed*
English→Czech	14.83	15.25	15.15
Czech→English	23.81	25.8	25.91
English→Spanish	34.89	35.22	35.01
Spanish→English	35.02	36.01	35.97

Table 2.3: BLEU score of each trained model.

- A mixed model, trained with Corpus *Europarl*[16] but tuned with Corpus *News Commentary*, referred to as *Mixed*.

The phrase-based SMT systems have been trained in the standard way, more exactly as described by Haddow and Koehn (2012). The corpora has been tokenized, truecased and cleaned using the standard Moses tools. The language model was built using KenLM Heafield (2011), and has an order of 5. Finally, the models were trained and tuned using the standard Moses tools. Finally, the translation phrase tables were compressed using the method proposed by Junczys-Dowmunt (2012). Only the best translation proposed by Moses for each segment has been considered in these experiments. The BLEU scores of each model are similar to the ones reported in the paper by Haddow and Koehn (2012); see Table 2.3 for the scores. Additionally, a language model trained with the in-domain training corpus will be used as a monolingual language predictor, as described in 2.3, that will be referred as *Monolingual*.

The KSR values for the automatic evaluation (using $M = L = 4$, as described in Section 2.4.5) are shown in Table 2.4. As expected, English–Spanish performed better (savings in keystrokes up to 48%) than English–Czech (with savings in keystrokes up to 31%) because it is generally easier to translate between English and Spanish, and the available corpora were larger in this case as well. Though statistically significant, the differences between the different in-domain and out-of-domain systems are relatively small: the best improvement comes from using in-domain data in the English→Spanish task, where an additional 8% keystrokes can be saved.

Nonetheless, using the black-box ITP approach in any of these scenarios leads to saving, at least, 30% keystrokes in the automatic evaluation. These results support the viability of the black-box ITP approach using different bilingual resources for translation tasks with closer languages like Spanish and English and more unrelated languages such as English and Czech. The monolingual predictor achieves similar results to in-domain ITP for en→es and cs→en, beats the ITP approach for en→cs, and gets worse results than ITP for es→en; the performance of the monolingual predictor is similar for all the language pairs, but for en→cz, and beats the ITP model when working with harder translation directions (such as English→Spanish, where some morphological information is missing).

[16]Excluding the 2,000 sentences used for the development set.

KSR	en→es	es→en	en→cs	cs→en
Europarl	0.70	0.53	0.78	0.69
Mixed	0.62	0.52	0.75	0.66
News Commentary	0.62	0.52	0.76	0.64
Monolingual	0.60	0.63	0.71	0.63

Table 2.4: KSR values from the automatic evaluation of the black-box ITP approach. In all cases, the test set consisted of sentences extracted from the News Commentary corpus.

2.5. Human evaluation of the heuristic approach

A preliminary evaluation of a real use of the black-box ITP heuristic approach involving 8 non-professional translators (volunteer computer science students) was conducted. For this evaluation, the best performing system was selected: Catalan to Spanish using Apertium as a bilingual resource, and the Forecat web interface.[17] All the users were Spanish native speakers who understood Catalan, but with no experience in translation. A set of 10 sentences (with 211 words in total) divided in 2 blocks in Catalan were randomly extracted from the same corpus used in the automatic evaluation. The test was designed to take around 20 minutes. The black-box ITP system was used with $L = M = 4$, the values found as optimal in Section 2.4.5, and the users interacted with the system via a web interface; only the segment being worked on was visible to the user.[18] Two additional sentences were added at the start of the test so the users were able to practice with the interface, and an additional notification was added between the first and the second block. After completing the test, they were surveyed about the usefulness of the system.

The users were divided into two groups: users 1–4 translated sentences 6–10 with no assistance and then 1–5 assisted by the black-box ITP tool, while users 5–8 translated sentences 1–5 with no assistance and then sentences 6–10 with assistance. The KSR and translation times for each user are shown in Table 2.5. This table also includes the emulated keystroke ratio (ESR), which is the value of KSR obtained by running the automatic evaluator (see Section 2.2) using the sentences entered by each user as the reference translations T; this can be considered as an approximation to the best result achievable with the ITP tool.

All users attained KSRs that were noticeably lower than 1 for the assisted translations and slightly higher than 1 when translating without the ITP system; they made some mistakes while typing, and replanned part of the translation, so they ended up typing more characters than the length of the final translation. However, the KSR values are worse than the ESR values obtained in the automatic evaluation: this shows that, theoretically, even better values for KSR could be attained for those sentences. Note that the ESR on the sentences that were translated using assistance is lower than the ESR in those that were translated without assistance; this happens because the former translations were partially assembled using the

[17] See Annex A.

[18] This approach resembles the approach taken for the automatic evaluation, where the evaluation is segment-based and no context is given.

User	Sentences 1–5			Sentences 6–10		
	KSR	Time (s)	ESR	KSR	Time (s)	ESR
#1	0.49	136	0.22	1.11	137	0.23
#2	0.64	144	0.15	1.21	86	0.22
#3	0.63	209	0.22	1.09	112	0.21
#4	0.37	189	0.22	1.22	199	0.18
#5	1.10	145	0.28	0.37	102	0.15
#6	1.24	150	0.27	0.51	154	0.17
#7	1.15	178	0.30	0.64	147	0.17
#8	1.18	118	0.39	0.58	93	0.15

Table 2.5: Results of the human evaluation. The elapsed time is measured in seconds. Users 1–4 translated the second sentence block (sentences 6–10) without assistance, then the first sentence block (sentences 1–5) with assistance; users 5–8 translated the first block without assistance, then the second block with assistance.

suggestions. Still, there is a wide margin to reduce the keystrokes used during the translation process.

All users manage to greatly improve their KSR when moving from the unassisted to the assisted tasks: the user that saves the most keystrokes (4) goes from 1.22 KSR to 0.37 KSR; the one that saves the least (3) goes from 1.09 to 0.63; a minimum of 46% keystrokes can be saved over the unassisted baseline. The ESR values show that the results can be improved even more; the worst ESR value is 0.23, meaning 77% keystrokes could have been saved if the user made no mistake and accepted all the viable suggestions. Three users (5, 7 and 8) managed to improve their translation speed when using the assistance, but two users (2 and 3) greatly decreased their translation speed when using the assistance; it cannot be concluded that the tool saves time or, on the contrary, makes users spend more time translating.

The users were surveyed to evaluate the following statements using the Likert (1932) scale, from 1 (complete disagreement) to 5 (complete agreement):

- The interface is easy to use (median answer of 5).
- I would use a tool like this in future translations (median answer of 5).
- I have found the suggestions useful (median answer of 4.5).
- The tool has allowed me to translate faster (median answer of 4.5).

It is evident that the evaluators perceived that the ITP system had helped them to translate faster, even when the time values in Table 2.5 show the opposite; the added cognitive load of reading and accepting or rejecting the suggestions make the users perceive the process as faster, even when it is not.

2.6. Shortcomings of the approach

The described heuristic approach relies on having a small amount of different origin positions (b_p) in P_C^S; that is, if there are suggestions in P_C^S coming from many different positions, it is likely that most of them will not be offered. The amount of different origin positions depends on two factors: the letters of the word being currently typed (that is, T_n), and the length of the prefix that has already been typed (that is, m in w_n^m). For example, when translating into English, a fair amount of suggestions will start with *the*, as it is a very common English determinant; thus, when w_n^1 =*t* is typed, the ambiguity of the origin position of the suggestions (or origin ambiguity) will be higher than w_n^1 =*x* (as there are far less words that start with x) or w_n^3 =*tha* was typed.

In order to check if the size of the set of different origin positions grows too large in some scenarios, causing problems to the heuristic, the evolution of the number of different starting positions for the suggestions in P_C^S has been measured. To this end, a new metric is introduced: the origin cardinality, the average number of different origin positions for the suggestions in P_C^S. The origin cardinality for each prefix length and language pair using the same test conditions used to support the third and fourth hypothesis in Section 2.4.3 can be seen in Figure 2.10.

The graphics show that the origin cardinality for short prefixes is quite high, specially when the prefix has a length of 1. Even when the heuristic approach obtains good results, better approaches can be taken: the heuristic can be improved by adding new sources of information, such as the particular letters of the word, but it is hard to extract patterns from this kind of data; machine learning algorithms are good at extracting complex patterns from lots of data. Therefore, a machine learning approach will be used.

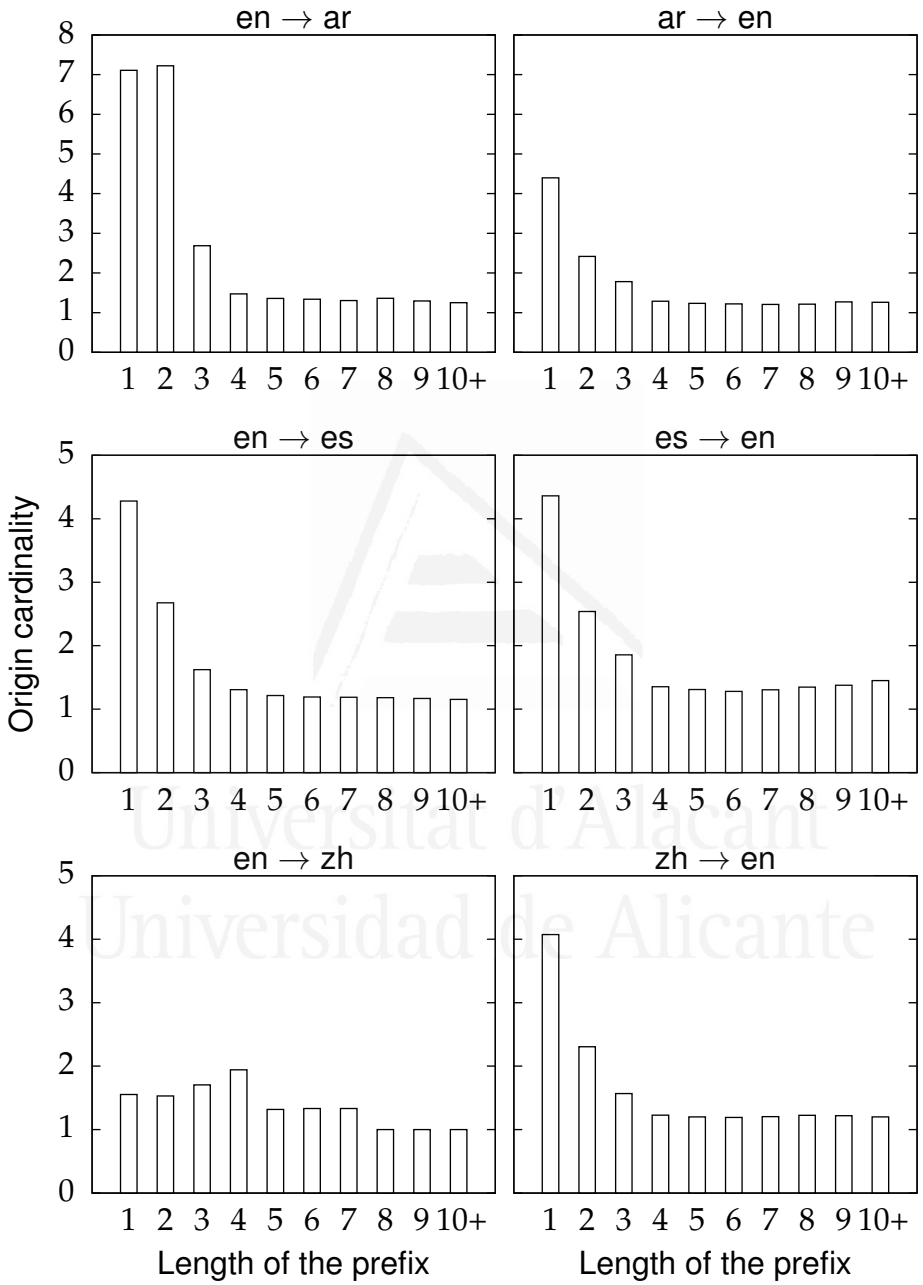


Figure 2.10: Results for the number of different origins for suggestions. In English to Arabic, many words start with "the" (*al*), that is joined with the word, thus the average number of places for prefixes of length 1 and 2 is very high; the remainder of language pairs follow the same trend: the origin cardinality diminishes really fast as the prefix grows.

Chapter 3

A machine learning based approach to black-box ITP

In the previous chapter, a simple heuristic that selects and ranks the suggestions to be shown to the user was presented; however, human-designed heuristics are not guaranteed to be optimal, as the extraction of complex, high-level patterns from vast amounts of data is a task that is usually too complicated for humans. The heuristic approach can be improved: on the one hand, more rigorous and principled models rather than intuitive heuristics can be used; on the other hand, a better ranking of suggestions can reduce the number of suggestions offered (reducing the cognitive effort needed to read and select suggestions), the number of keystrokes or both. Consequently, in this chapter, a proposal to replace the previous intuitive heuristics with a ranker obtained via machine learning is presented.

In this chapter, the machine-learning based approach to black-box ITP and each feature will be detailed in Section 3.1. Later, the experimental setup and results will be described in Section 3.2, and a comparison with glass-box ITP will be described in Section 3.3. Afterwards, the application of feature selection will be outlined in Section 3.4, followed by the description of a human evaluation task with amateur translators in Section 3.5. Finally, a thresholding strategy developed to address the most common complaint in the previous human evaluation will be described in Section 3.6, and the results of a human evaluation with professional translators will be described in Section 3.7, along with a summary comparing both evaluation rounds.

3.1. Machine learning

Machine learning is a set of techniques and algorithms that let computers learn behaviours that have not been explicitly coded. They predict the class (classification) or a value on a continuum (regression) for an instance (or event) that is usually represented by a vector of features (values or labels that describe the different characteristics of the instance); the

prediction is based on a model learnt from a training set, that can be labeled (supervised learning) or unlabeled (unsupervised learning).

In order to improve the results obtained by the heuristic, the problem of ranking and selecting which suggestions to show was described as a machine learning problem, and used a feedforward neural network (Duda et al., 2000, Chapter 6) to solve it. As previously explained in Chapter 1.1.3, a neural network is a statistical model inspired in the workings of the brain: neurons (also named units) take several weighted inputs, process them using an activation function, and produce an output; the neurons are interconnected and organized in layers. Neural networks are capable of representing a continuous function with arbitrary precision, given enough units in the hidden layer (Cybenko, 1989). In this setup, the neural network will have one input layer, with a single unit per feature; one hidden layer where each unit uses the logistic activation function;[1] and one output layer, with one unit and the identity activation function; the output of this neural network can be interpreted as a probability (Richard and Lippmann, 1991).

The definition of features that properly capture the information of the event is a must: if the features do not contain enough information to properly represent the events, the system will not able to correctly learn, thus achieving poor results when used to classify. For this purpose, 30 features, $f_i, 1 \leq i \leq 30$ have been defined, for each suggestion p. Each suggestion is scored on its own, independently of the rest of suggestions in $P_C^S(W_n^m)$, but some features add information about other suggestions, such as suggestions that were used or could have been used to type the prefix (f_{29} and f_{30}), or whether the last word of the prefix was typed or accepted as part of a suggestion (f_{27}).[2] The single output value of this neural network is an approximation for $g(p, W_n^m)$, the goodness function, as explained in Chapter 2.1.1.

Some of the features described here are nominal, and neural networks are unable to directly process them; those features will be transformed into a multiple (as many as classes in the nominal feature) one-hot-encoded binary values, and then treated as numeric. Likewise, binary features (a special case of nominal features with only 2 possible classes) are treated as numeric. After applying this transformation, the final number of input values is 79, but for the English→Arabic scenario, where it goes up to 91.[3] To avoid overcomplicating, the transformation will be ignored during the description of these features.

3.1.1. Description and justification of each feature

Length of the suggestion. The length of the span the suggestion is generated from ($f_1 = e_p - b_p$) and the length of the translated subsegment ($f_2 = |t_p|$), both at the word and the character (f_3, f_4) level. As discussed while explaining the simple heuristic in Chapter 2.4.4,

[1] Different numbers of units in the hidden layer will be explored in order to find the best configuration.

[2] More advanced approaches, such as learning to rank (Hang, 2011), can be used to fully capture the relationship between the different suggestions.

[3] A feature (f_{26}) encodes the starting letter of the suggestion; Arabic uses a different alphabet and has a different number of letters than English, 46.

Example 3.1

To illustrate the different features, the Spanish sentence

el coche blanco está destrozado

and the English sentence

the white car is wrecked

will be used. The following translated subsegments will be available:

Source sentence	el	coche	blanco	está	destrozado
Subsegments of length 1	(p1) the	(p2) car	(p3) white	(p4) is	(p5) wrecked
Subsegments of length 2	(p6) the white		(p7) white is		
		(p8) white car		(p9) is wrecked	
Subsegments of length 3	(p10) the white car				
		(p11) white car is			
			(p12) white is wrecked		

Unless noted otherwise, the typed prefix will be W_5^1 =*the white car is w*.

short suggestions are often viable, but do not save a lot of effort, and long suggestions are seldom viable, but save a lot of keypresses.

Under the conditions described in Example 3.1, for p_3 (t_{p_3} =*white*),

- $f_1 = e_p - b_p = 1$, as t_{p_3} is just one word, *blanco*.
- $f_2 = |t_p| = 1$, as t_{p_3} is just one word, *white* .

Position of the suggestion A set of features relating to the position where each suggestion comes from and where it is offered. The features include the absolute position in the source sentence $f_5 = b_p$ and the position being currently worked on in the target sentence $f_6 = n$, the position normalized by the length of the sentence $f_7 = b_p/|S|$ and $f_8 = n/|T|$,[4] the distance between source and target positions, both with absolute positions $f_9 = n - b_p$ and their normalized counterparts $f_{10} = n/|T| - b_p/|S|$, and their position ratios $f_{11} = n/b_p$. These features help to differentiate suggestions attending to the distance between the origin of the suggestion and where it was used or offered. Both the absolute and relative distances are used in order to help the model to capture information both from short and long sentences: short sentences will have smaller values for absolute differences than longer sentences, but the relative distance will always be in the (0–1) interval. An equivalent set of features is also used for character level positions and distances $\{f_{12}...f_{18}\}$.

[4]During testing, when the final length of T cannot be known, it is assumed that $|T| = |S|$.

Under the conditions described in Example 3.1, for p_3 (t_{p_3} =*white*),

- $f_5 = b_p = 3$, as $b_{p_3} = 3$.
- $f_6 = n = 5$, as the fifth word in T is being written.
- $f_7 = b_p/|S| = 3/5 = 0.6$, as $b_{p_3} = 3$ and $|S| = 5$.
- $f_8 = n/|T| = 5/5 = 1$ during training, as $n = 5$ and $|T| = 5$ is known, and $f_8 = 5/5 = 1$ during test, as $n = 5$, $|T|$ is unknown, and $|S| = 5$.
- $f_9 = n - b_p = 5 - 3 = 2$, as p_3 takes origin on the third word of S and is offered on the fifth word of T.
- $f_{10} = n/|T| - b_p/|S| = f_8 - f_7 = 0.6 - 1 = 0.4$
- $f_{11} = n/b_p = 5/3 = 1.66$, as p_3 takes origin on the third word of S and is offered on the fifth word of T.

Position and length A set of 3 nominalized features $\{f_{19}, f_{20}, f_{21}\}$ relating to the position and the length of suggestion. Each feature takes a value in the $\{\text{short}, \text{long}\} \times \{\text{close}, \text{far}\}$ set. A suggestion is classified as short if it has 2 or less words (that is, $f_2 \leq 2$) or 10 or less characters ($f_4 \leq 10$), and long otherwise. A suggestion is classified as close if it has a distance of 3 or less words between the origin position and where it was used or offered ($f_9 \leq 3$), 20 or less characters away ($f_{15} \leq 20$) or the ratio position ($f_{11} < 1.2$) is lower than 1.2; far otherwise. The feature f_{19} uses word-level length and distance, f_{20} uses character-level length and distance, and f_{21} uses word-level length and ratio. These features help the neural network to capture explicit information about the length and position of the suggestions in order to better extract some patterns, such as long, close suggestions being more useful than long, far ones.

Under the conditions described in Example 3.1, for p_3 (t_{p_3} =*white*),

- f_{19} = short, close, as $f_2 \leq 2$ and $f_9 \leq 3$.
- f_{20} = short, close, as $f_4 \leq 10$ and $f_{15} \leq 20$.
- f_{21} = short, far, as $f_2 \leq 2$ and $f_{11} > 1.2$.

Distance distribution. Given a training set, the average $\overline{x}$ and standard deviation σ values are computed for the distribution of normalized distances and position ratios for the *winning suggestions* set, and four features are defined:

- $f_{22} = (f_{10} - \overline{x})/\sigma$, that is, the normalized distance feature projected into a normal distribution $\mathcal{N}(0, 1)$;

- f_{23}, a nominal feature that has 4 different classes depending on the relationship between f_{10} and the distribution: less than half standard deviation (σ) away from $\overline{x}$, between half and a full σ, between one and two σ, or further away than 2σ;
- two more features (f_{24} and f_{25}) similar to f_{22} and f_{23}, but replacing f_{10} with f_{11}, using their respective average and standard deviation.

Under the conditions described in Example 3.1, for p_3 (t_{p_3} =*white*), and assuming a normal distribution ($\overline{x} = 0$, $\sigma = 1$),

- $f_{22} = (f_{10} - \overline{x})/\sigma = (0.4 - 0)/1 = 0.4$.
- f_{23} belongs to the category where f_{10} is less than half σ away.

Starting letter of the suggestion. The nominal feature f_{26} takes the value of the first letter of the suggestion (ignoring the capitalization) if it belongs to the English alphabet, and replaced with a generic *other* token otherwise. As discussed in Chapter 2.6, the starting letter of a word is related to the size of the set of compatible suggestions P_C^S; this feature, along with the length of the suggestion features ($\{f_1 ... f_4\}$, helps the neural network to learn how to represent this phenomena; e.g., the neural network may learn to be more selective with suggestions that start with common letters (such as *t* in English) than with suggestions that start with uncommon ones.

Under the conditions described in Example 3.1, for p_3 (t_{p_3} =*white*), f_{26} =*w*.

Last action taken The binary feature f_{27} represents the action taken for the previous word: whether it was typed it or it was part of an accepted suggestion.

Under the conditions described in Example 3.1, for p_3 (t_{p_3} =*white*), if p_2 or p_6 were used, f_{27} would be true, otherwise, false.

Relationship to the last used suggestion The nominal feature f_{28} represents the relationship of the current suggestion p with the last one used p'. It takes 5 possible values, depending on the case:

- p ends before p', $e_p + 1 < b_{p'}$
- p is contiguous and is placed immediately before p', $e_p + 1 = b_{p'}$
- p and p' overlap,[5] $b_p \in [b_{p'}, e_{p'}] \vee e_p \in [b_{p'}, e_{p'}]$
- p is contiguous and is placed immediately after p', $b_p - 1 = e_{p'}$

[5]This would mean a given part of S participates on the generation of different parts of T, which, in general terms, is unlikely to be desirable.

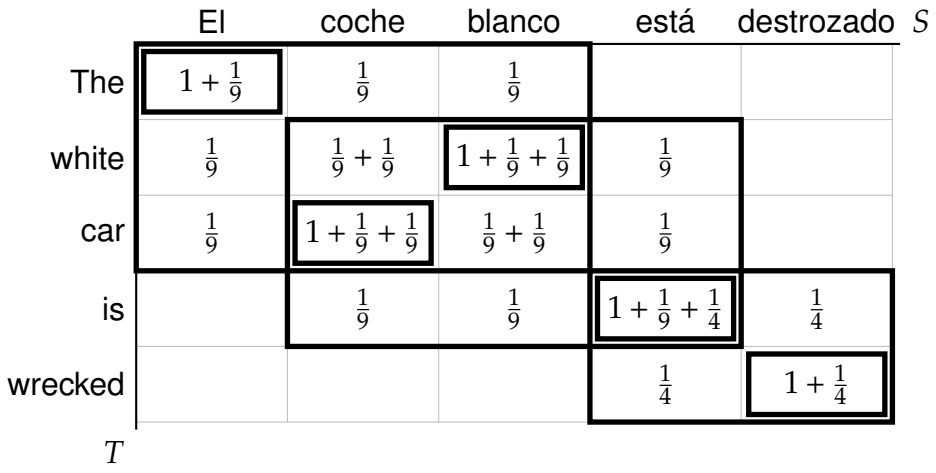


Figure 3.1: Alignment strengths of each position under the conditions described in Example 3.1. Each suggestion is given an alignment strength of 1, that gets evenly split along the surface of the suggestion. Some suggestions like 'white is wrecked' cannot be aligned. Each position has a label with the alignment strength over it; positions with more alignment strength are more likely to be aligned.

- p starts after p', $b_p - 1 > e_{p'}$

At the beginning of the translation, where no suggestion has been used, all the suggestions are deemed as belonging to the last case (starts after p').

Under the conditions described in Example 3.1, for p_3 (t_{p_3} =*white*), f_{28} would belong to the fourth category if if p_2 or p_6 were used, to the fifth if p_1 was used, or to the third if p_3, p_7 or p_{12} were used.

Light alignment model The light alignment model described by Esplà-Gomis et al. (2012) is able to use the information contained in previously existing bilingual resources without any additional training. When compared to GIZA++ (Och and Ney, 2003) in an alignment task, both approaches obtained similar performances. The model relies on a intuitive idea: each subsegment that contains S_j (the j-th word of S) whose translation covers T_y (the y-th word of T), and vice-versa, increases the likelihood (measured in alignment strength) of S_j and T_y to be aligned. An example of how the model computes each alignment strength is shown in Figure 3.1, and how it finds the most likely alignments in Figure 3.2.

In the proposal by Esplà-Gomis et al. (2012), all possible whole-word subsegments of S and T up to a given length are generated then translated using a bilingual resource.[6] This procedure is similar to the one performed to generate the suggestions in black-box ITP, but there is one main difference: while the black-box ITP approach translates the subsegments of S to the target language, the light alignment model also translates the subsegments of T to the source language. The black-box approach cannot use the subsegments of T, as T is only

[6] In the work by Esplà-Gomis et al. (2012), only MT is used.

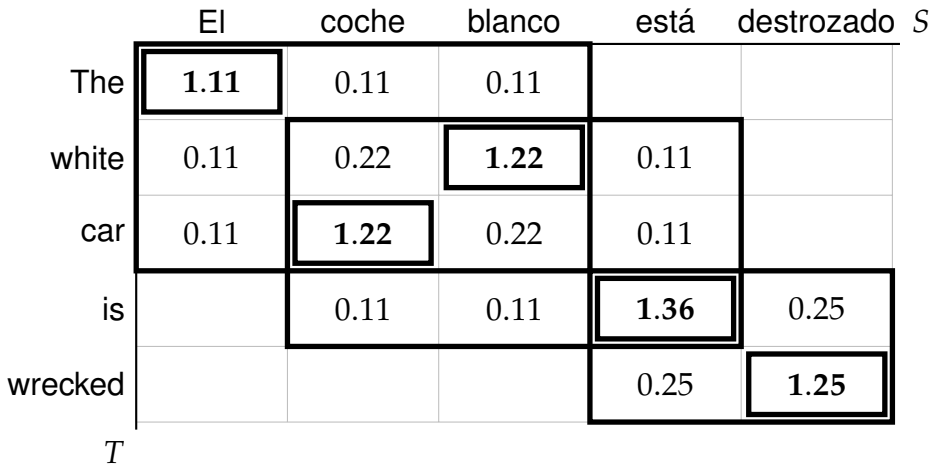


Figure 3.2: Alignment strengths of each position under the conditions described in Example 3.1. The most likely alignment for each pair of words shows the probability with a bold typeface. This figure represents the same data as Figure 3.1, but the scores are shown as decimals rather than fractions.

available as a prefix that gets completed as the user types it, and starts as an empty segment. Translating these subsegments as the translator types is unfeasible due to three reasons:

- It may slow down the system to the point that it could not be effectively used, specially if working with complex MT systems, if the computer has low processing power, or if the translation is performed remotely.
- The translation quality of the last words of the typed prefix may be low, as they lack the posterior context until it gets typed.
- Translating T as it is being typed would increase[7] the cost of translating; this can be costly if using a translation-as-a-service resource.
- Additionally, while it is guaranteed that a bilingual resource that translates the language of S into the language of T exists and is available (otherwise, using the black-box ITP approach would not be possible), it is not guaranteed that a system that performs the translation in the opposite direction exists and is available.

The work by Esplà-Gomis et al. (2012) uses the alignment strength to align source and target words: the higher the alignment strength between a source and a target word, the more likely it is for them to be aligned with each other. This hypothesis is adapted for the black-box ITP approach: suggestions that have a high alignment strength with the position currently being worked on (m) are more likely to be used. To this end, the set of suggestions that have a suffix of W_n^m as a prefix is explored. Let Pr and Suf be the character-level set of prefixes

[7] Assuming the length of S and T are similar and T is typed without mistakes, the cost would be roughly twice as high. As the black-box approach only translates unique subsegments, if there are repeated subsegments, the cost of translating S or T may be lower.

and suffixes of a given string; the set of suggestions that overlap with and extend the end of the current typed prefix W_n^m, extender, is defined as:

$$\text{extender}(W_n^m, P^S) = \{p \in P^S : \Pr(t_p) \cap \text{Suf}(W_n^m) \neq \varnothing\}$$

As discussed by Esplà-Gomis et al. (2012), the light alignment model operates on the idea that each subsegment alignment applies an *alignment pressure*: the larger the surface covered by the subsegment, the weaker the confidence in the alignment. Each subsegment pair is given an *alignment weight* of 1 unit that is split evenly along the surface of the suggestion as measured in square words. So, the force exerted on each position by a suggestion p is:

$$v_p = \frac{1}{(e_p - b_p)|t_p|}$$

For example, in Figure 3.2, *El coche blanco* is translated to *The white car*. As the translation exists, it exerts a pressure of $1/9$ on each of the 9 covered positions. Therefore, there is some evidence that *blanco* and *white* are aligned. However, if this subsegment was the only one available, it would be as likely as *blanco* being aligned with *the*, yet, when the rest of subsegments are factored in (*blanco*, translated to *white*, and *coche blanco está*, translated to *white car is*), the most likely alignment for *blanco* is *white*.

To estimate which position j of S is being currently translated, the model looks at how much alignment strength is exerted on it; the alignment strength, $\omega(j, W_n^m)$ is defined as:

$$\omega(j, W_n^m) = \sum_{p \in \text{extender}(W_n^m)} \begin{cases} v_p & \text{if } j \in [b_p, e_p] \\ 0 & \text{otherwise} \end{cases}$$

As discussed, a high alignment strength is interpreted as a high alignment confidence; positions j of S with a higher weight for the position m of T currently being worked on are more likely to be aligned. Suggestions that overlap with m and collect the most alignment strength on their surface are more likely to be a translation of the part of S being currently translated, hence being more likely to be useful. For example, Figure 3.3 shows that, with W_5^1 =*The white car is w*, and $P_C(W_5^1) = \{$*white*,*wrecked*$\}$, wrecked is more likely to be aligned with the word being currently translated *destrozado* than *white*, hence, more likely to be viable.

Moreover, suggestions that have already been used in the past should be ignored, as they are already aligned with a previous position of S. To this end, the set of accepted suggestions $P_A^S(W_m^n)$ is defined.[8] Having this in consideration, the combined alignment strength "pressing" the area under each suggestion is added and used as a positive feature:

$$f_{29} = \sum_{j \in [b_p, e_p]} \begin{cases} \omega(j, W_n^m) & \text{if } p \notin P_A^S(W_m^n) \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

Additionally, this alignment model can also be exploited to discredit those compatible suggestions that take origin in parts of S that are likely to have been already translated in W_m^n.

[8]The set of accepted suggestions may contain different elements to the ones contained in $\psi^S(M)$; the user does not necessarily select the longest compatible suggestions.

To this end, the model calculates the alignment strength over the already translated part of the sentence $W_{n-1} = T_1 ... T_{n-1}$. Let occurrences$(t_p, T)$ be the function that returns the number of times t_p appears as a subsequence (substring) of T, the set of suggestions that overlap with W_{n-1} is defined as:

$$\text{overlap}(W_{n-1}) = \{p \in P^S : \text{occurrences}(t_p, W_{n-1}) > 0\}$$

As some suggestions may have more than one origin position b_p,[9] there is some uncertainty about which alignment is the proper one. For addressing this problem, the alignment strength is split between the different number of matches,

$$u_p = \frac{v_p}{\text{occurrences}(t_p, W_{n-1})}$$

The past alignment strength function ω_{past} is defined as:

$$\omega_{\text{past}}(j, W_{n-1}) = \sum_{p \in \text{overlap}(W_{n-1})} \begin{cases} u_p & \text{if} j \in [b_p, e_p] \\ 0 & \text{otherwise} \end{cases}$$

Finally, a feature that captures how likely is that a suggestion p is useless as it is the translation of a subsegment of S that has already been translated in W_m^n is defined as:

$$f_{30} = \sum_{j \in [b_p, e_p]} \begin{cases} \omega_{\text{past}}(j, W_{n-1}) & \text{if } p \notin P^S_{\text{accepted}} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

An example of how this both features work is discussed in Figure 3.3.

3.2. Evaluation of the machine learning approach

3.2.1. Modifications of the automatic evaluation

In order to measure the performance of this approach, a method similar to the one described in Chapter 2.2 will be used: an automatic system will "type" a provided translation reference while "evaluating" all the suggestions offered, "accepting" the longest viable suggestion, if any. There is an additional way to interact with the suggestions: a prefix of the suggestion can be selected, one word per keypress. Hence, the cost of selecting the prefix of a suggestion is one keypress per word in the selected prefix plus one keypress for selecting the suggestion prefix. The user may press the `Tab` to select successive words of the suggestion. Therefore, the sets of viable suggestions $P_V^S(W_n^m)$ has to be redefined; the set of viable suggestions whose first $x \leq |t_p|$ words can be used to advance in the translation, $P_{Vx}^S(W_n^m)$, is defined as:

$$P_{Vx}^S(W_n^m) = \{p \in P_C^S(W_n^m) \mid \underset{k \in [1,x]}{\forall \text{k}} \; t_p[k] = T_{n+k}\}$$

[9] This happens when a subsegment of S is repeated, or when one or more different subsegments of S are translated to the same target subsegment.

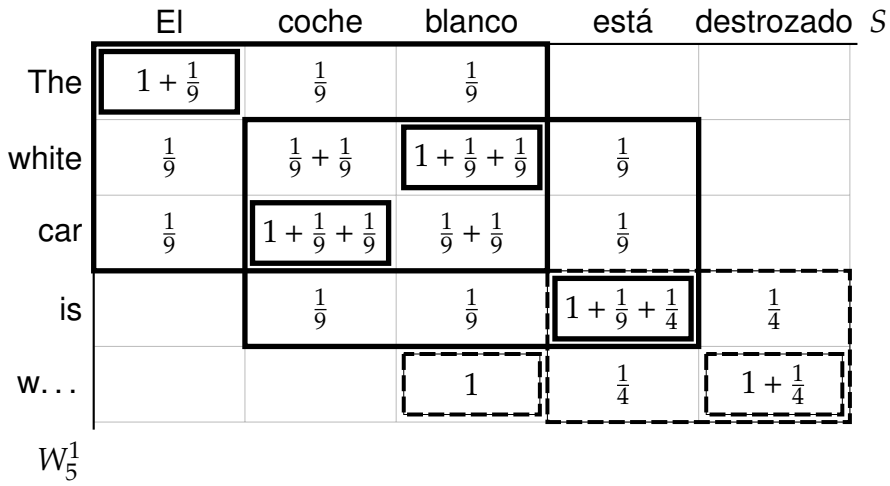


Figure 3.3: Alignment strengths of the different positions under the conditions described in Example 3.1. Rectangles with solid outlines represent suggestions in overlap(W_5), those with dashed outlines represent suggestions in extender(W_5^1). Assuming p_3 (t_{p_3} =*white*) $\notin P^S_{\text{accepted}}$ ('white' has not been used when typing T), *white* has $f_{29} = 1$ (eq. 3.1) and $f_{30} = 1 - (1 + 6/9)$ (eq. 3.2), and p_5 (t_{p_5} =*wrecked*) has $f_{29} = f_{30} = 1 + 1/4$.

For example, $P^S_{V3}(W_n^m)$ are those suggestions whose first three words can be used to advance in T; $P^S_{V3}(W_n^m)$ is a subset of $P^S_{V2}(W_n^m)$.

As now suggestions can be fully compatible or partially compatible with the T, the set of viable suggestions have to be redefined as the intersection between the offered suggestions set and the set of viable suggestions whose first x words can be used to advance:

$$P^S_{OVx}(W_n^m, M) = \begin{cases} P^S_O(W_n^m, M) \cap P^S_V(W_n^m) & \text{if} \quad x = 0 \\ P^S_O(W_n^m, M) \cap P^S_{Vx}(W_n^m) & \text{otherwise} \end{cases}.$$

For example, $P^S_{OV3}(W_n^m, M)$ are the suggestions whose three first words are compatible with T, and $P^S_{OV0}(W_n^m, M)$ is the set of suggestions that can be used by selecting 0 words as a prefix, that is, those that are fully selected.

The winning suggestion (redefined as $\phi^S_*(W_n^m, M)$) is then the one that saves the most keystrokes, that is, the suggestion that maximizes $|t_p|_c - x$ in the combination of the offered suggestion sets, with x as the number of words of the prefix to be inserted. Full suggestions can be used with just one keystroke, therefore the number of keystrokes saved is $|t_p| - 1$ for the suggestions in $P^S_{OV0}(W_n^m, M)$; for the suggestions where only a prefix is compatible, the number of keystrokes saved is further decreased by the number of times the `Tab` key has to be pressed. The winning suggestion ϕ^S_* is redefined as:

$$\phi^S_*(W_n^m, M) = \underset{p \in \bigcup_{x=0}^{\lambda} P^S_{OVx}(W_m^n, M)}{\operatorname{argmax}} (|t_p|_c - x)$$

with

$$\lambda = \underset{p \in P_C^S(W_n^m)}{\operatorname{argmax}}(|t_p|)$$

Finally, the automatic evaluation system output gets updated; now the sequence of actions not only comprise keystrokes and selecting full suggestions, but also selection suggestion prefixes. Those suggestions or prefixes of suggestions that are part of the sequence of actions generated by the automatic evaluation are in $\psi_*^S(M)$.

The same metric used in Chapter 2.2.2, the keystroke ratio (KSR) will be used to quantify the performance improvement.

3.2.2. Corpora and resources

Training, development and test data for the neural network are extracted from a corpus with 15,000 English–Spanish sentences extracted from Corpus Europarl, a collection of proceedings from the European Parliament; 11,000 sentences were used as training set, 1,000 as development or validation set and the remaining 3,000 as test set. The free/open-source statistical MT system Moses (Koehn et al., 2007) was used as a bilingual MT engine capable of providing the translation of the subsegments. Moses was trained over 155,760 independent sentences from the same corpus, following the standard procedure for training a baseline system.[10] The evaluation was conducted for the translation of texts from English to Spanish.

All the experiments in the section will use these corpora and resources, unless explicitly noted. All the differences between the values shown in the figures in this section are statistically significant, calculated using bootstrap resampling (Koehn, 2004) and 1000 iterations with $p = 0.05$, unless explicitly noted.

3.2.3. Experimental setup

For testing the performance of the machine learning based approach, four different configurations were trained depending on the set of features used and the kind of suggestions they will learn to identify. There are two different configurations depending on the set of features used:

- *Full feature set*: all the features discussed on the previous Section 3.1.1.
- *Reduced feature set*: a reduced feature set consisting of only two features, the length of the suggestion (f_2) and the normalized distance (f_{10}), as discussed on the previous Section 3.1.1. This means that the model has access to similar information as that

[10] `http://www.statmt.org/moses/?n=Moses.Baseline`

available to the intuitive heuristic method in Chapter 2.4: therefore, it is easier to analyze which part of the improvement comes from using a machine learning technique rather than a heuristic, and which part comes from the rest of features, that capture information that was not used by the heuristic.

Similarly, there are two different configurations depending on the kind of suggestions the model will learn to identify:

- *Winning suggestions*: the set of *winning suggestions*, that is, those that belong to $\psi_*^S(M)$, the suggestions that get chosen during the automatic evaluation procedure described in Chapter 2.2. Winning suggestions are given a score of 1, and the rest get a score of 0.[11] For the training procedure, only those suggestions that were actually offered are used; that is, those suggestions that could have been offered for a word that was inserted as part of an accepted suggestion are not used to train the model.[12]

- *Viable suggestions*: the set of *viable suggestions*, namely those that belong to $P_V^S(W_n^m)$, the suggestions that could be used for advancing the translation for the current partially translated sentence W_n^m, but not necessarily are part of the suboptimal sequence of actions the greedy automatic evaluation procedure executes. For instance, given S =*Un coche rojo*, T =*A red car*, W_2^1 =*A r*, t_{p_1} =*red* and t_{p_2} =*red car* are both deemed as viable, even when p_2 would save more keystrokes. Viable suggestions are given a score of 1, and the remainder get a score of 0. All the suggestions in P^S are used to train this model.

To generate the training set, using the source sentence and the reference, a program iteratively considers the first letter of each word and evaluates all the possible suggestions; those that fully match the following words of the reference are tagged as *viable* in that context, and those that would be part of the greedy suboptimal sequence of actions that leads to the best performance are tagged as *winning* in that context. Only the first letter of each word is considered to avoid overfitting, for the following reasons:

- The set of viable suggestions $P_V^S(W_n^m)$ does not change with increasing values of m, e.g. for W_1^2=*The r*, W_2^2=*The re* and W_3^2=*The red*, P_V^S will always contain t_{p_1} =*red* and t_{p_2} =*red car*. This could lead to overfitting the positive examples, as they would be repeated once per each character in the word, while the negative examples get filtered as the prefix is typed.

- The suggestions p that get a score of 0 (that is, $p \in (P_O^S(W_n^m, M) - P_V^S(W_n^m))$) can appear multiple times, e.g. for W_2^1=*The r*, W_2^2=*The re* and W_2^3=*The red*, t_p =*redemption* will be in $P_O^S(W_2^n, M) n = [1, 2, 3]$, but will get a score of 0, as it will not be in $P_V^S(W_2^n)\, n = \{1, 2, 3\}$. This could lead to overfitting the negative examples.

[11] During training, the desired output values will be 0 and 1, but during testing the network output will be a real value between 0 and 1 that correlates with the goodness of the suggestion.

[12] This is the usual behaviour of the greedy algorithm described in Section 2.2.1.

M	1	2	3	4	5	6	7	8
Full set, viable	0.87	0.88	0.88	0.88	0.88	0.87	0.87	0.86
Full set, winning	0.92	0.95	0.96	0.96	0.96	0.95	0.95	0.95
Reduced set, viable	0.91	0.94	0.93	0.93	0.93	0.92	0.91	0.91
Reduced set, winning	0.90	0.90	0.89	0.87	0.87	0.86	0.85	0.84

Table 3.1: Pearson correlation coefficients between the minimum squared error (MSE) of the resulting neural network model, computed over the development set, and the final keystroke ratio (KSR) of the automatic evaluation procedure using the neural network model to order the suggestions, for each maximum number of suggestions M and configuration. Every configuration shows strong positive correlation between MSE and KSR, that is, a correlation $\simeq 1$.

3.2.4. Configuration of the neural network

The neural network configuration had one hidden layer with a variable number of units. Neural networks are capable of representing a continuous function with arbitrary precision, given there are enough units in the hidden layer (Cybenko, 1989), but how many units are enough cannot be determined beforehand. Therefore, the performance of the system using different number of units in the hidden layer was explored: $N \in \{2, 4, 8, 16\}$ for the reduced set and $N \in \{2, 8, 32, 128\}$ for the full set. The logistic activation function was used for all the units in this layer. The neural network had as many input units as features after applying the binarization of the nominal features, that is, 79, and only one output unit that, unlike the rest of units, used the identity activation function, as it is performing a regression task.

For the training procedure, backpropagation (Rumelhart et al., 1988) with mean squared error (MSE) as the error function was used to optimize, and no momentum or regularization of any kind were used when performing gradient descent. As neural networks have trouble dealing with local minima (Gori and Tesi, 1992), each neural network has been trained five times with different random initializations in the $[-1, 1]$ range, and three different learning rates $\alpha \in \{10^{-2}, 10^{-3}, 10^{-4}\}$.

Table 3.1 shows that there was a strong correlation between MSE and KSR; as a result of this it can be presumed that the systems with lower minimum squared error MSE will do a better job ranking the suggestions, so the model that achieves the lowest MSE out of the 5 different initializations was selected. While the weights were updated after computing the error of every instance in the training set, the decision to stop the training (also known as convergence condition) is based on the convergence of the MSE[13] of the validation set, in order to minimise the risk of overfitting. These algorithms will be used as implemented in the free/open-source neural network library FANN (Nissen, 2003).

[13] While the KSR could also have been used as a stop condition, the cost of performing the full automatic evaluation every epoch with a representative set is prohibitive.

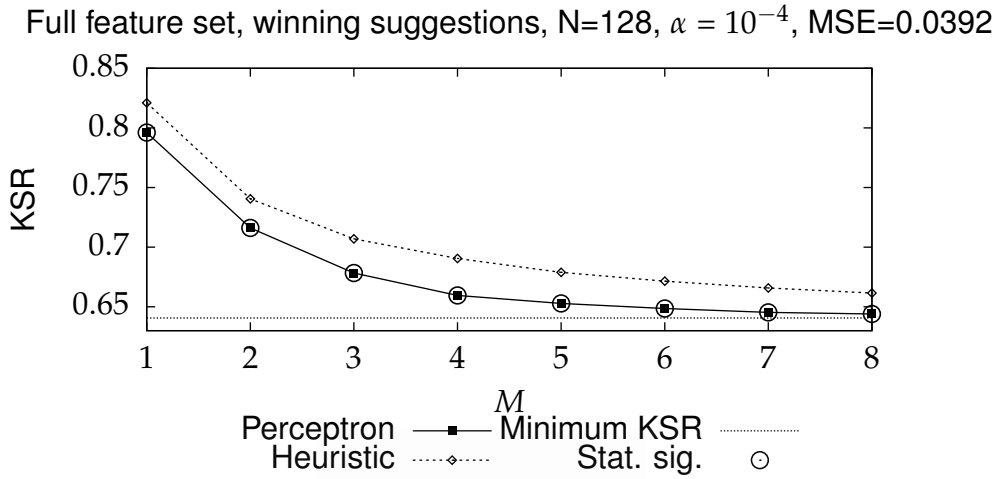


Figure 3.4: KSR for the multilayer perceptron configuration trained using the *full feature set* and the *winning suggestions* set that got the best MSE. The neural network approach beats the baseline (the heuristic approach) in a statistically significant way for every tested value of the maximum number of suggestions M.

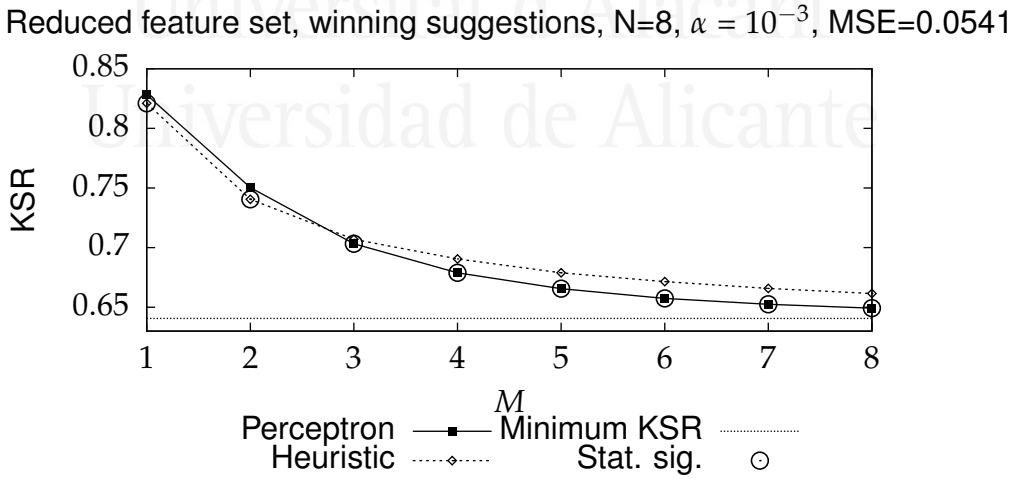


Figure 3.5: KSR for the multilayer perceptron configuration trained using the *reduced feature set* and the *winning suggestions* set that got the best MSE. The neural network approach beats the baseline (the heuristic approach) in a statistically significant way for every tested value of M, but for $M = 1, 2$, where it achieves significantly worse KSR.

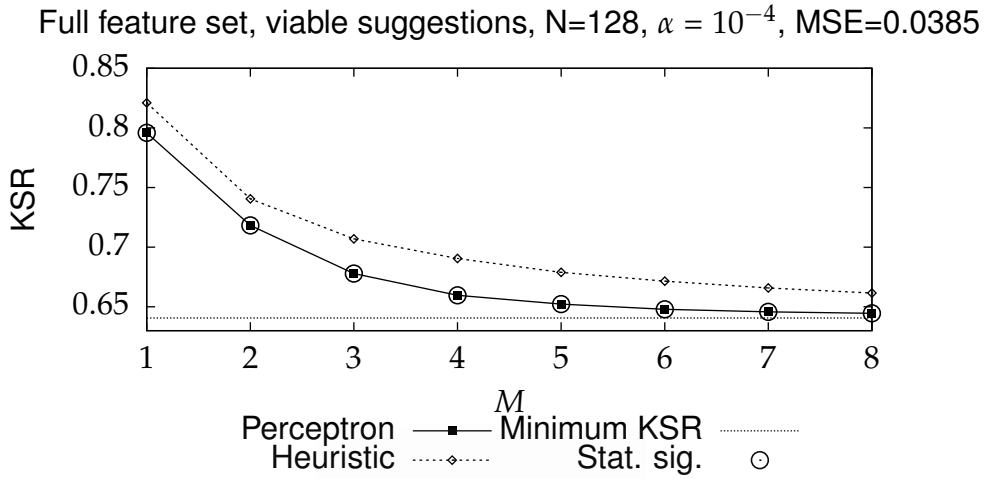


Figure 3.6: KSR for the multilayer perceptron configuration trained using the *full feature set* and the *viable suggestions* set that got the best MSE. The neural network approach beats the baseline (the heuristic approach) in a statistically significant way for every tested value of the maximum number of suggestions M.

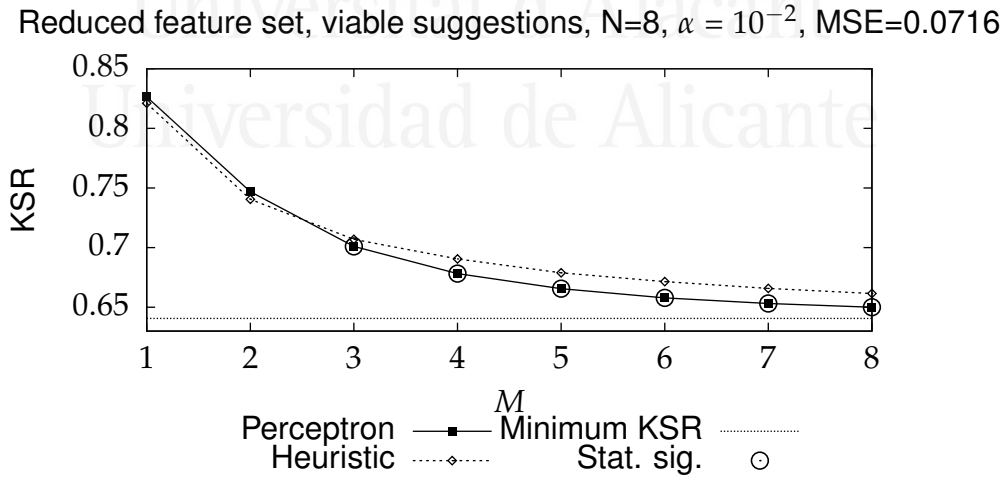


Figure 3.7: KSR for the multilayer perceptron configuration trained using the *reduced feature set* and the *viable suggestions* set that got the best MSE. The neural network approach beats the baseline (the heuristic approach) in a statistically significant way for every tested value of M, but for $M = 1, 2$, where it achieves significantly worse KSR.

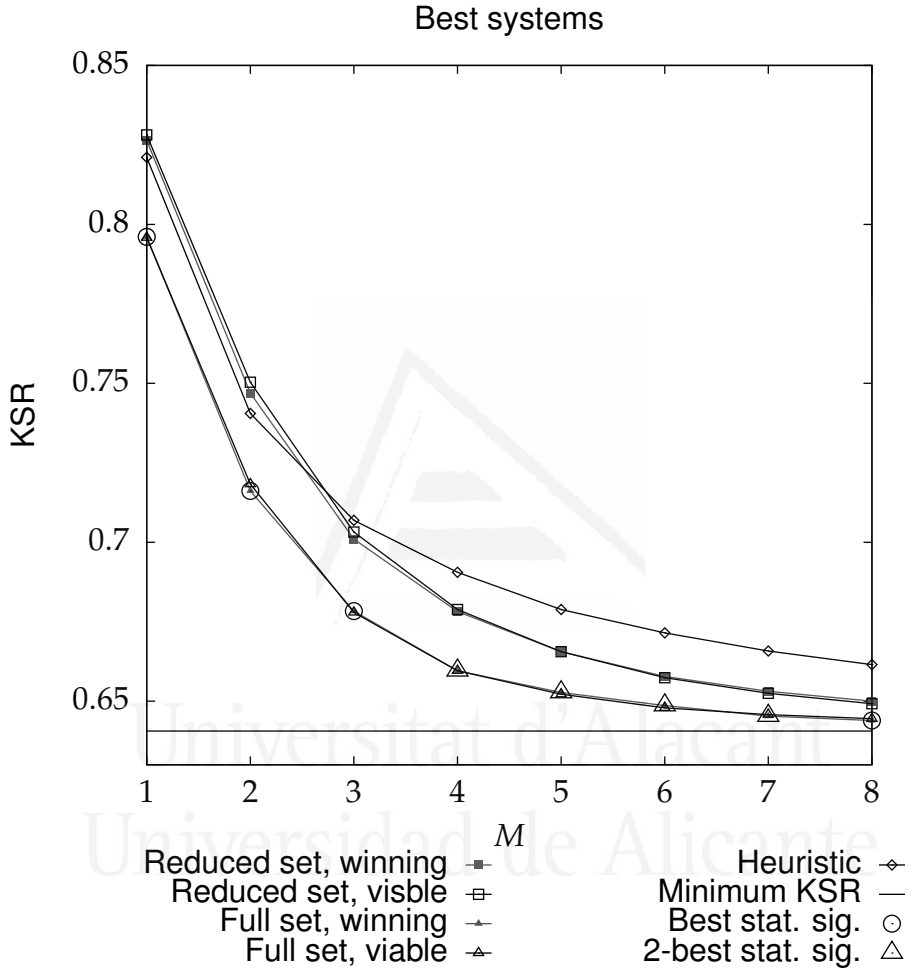


Figure 3.8: KSR for the best multilayer perceptron configuration for each training set. The best models perform similarly regardless if predicting *winning* or *viable suggestions*, even though the MSE is higher for the ones predicting viable suggestions. *Best statistical significance* marks those values of the maximum number of suggestions M where the model using the *full feature set with winning suggestions* is statistically significantly better than the rest; *2-best statistical significance* marks those where it is not significantly better or worse than the model using the *full feature set with viable suggestions*, but both are statistically significantly better than the rest.

3.2.5. Results of the evaluation

Results show that the perceptrons trained with the *winning suggestions* and the full feature set (Figure 3.4) perform notably better than the ones trained with the reduced set (Figure 3.5). While several systems trained with the full feature set performed statistically significantly better than the heuristic approach, no reduced feature set system manages to outperform the heuristic baseline with $M = [1, 2]$. Still, the reduced feature set models achieve better results for the rest of values of M; this confirms that part of the gains are due to the use of a machine learning approach; the fact that the model with the full feature set beats the one with the reduced set confirms that the features that capture information previously not being used are useful for this task.

In every figure, the line "Minimum KSR" denotes the best KSR that can achieved offering all the suggestions using the current automatic evaluation procedure,[14] and circled points denote the best statistically significant KSR. Figure 3.8 compares the performance of the system that performs the best for each different configuration. The models trained with the *viable suggestions* set (Figure 3.6 and Figure 3.7) have a similar performance, albeit slightly worse than the ones trained with the *winning suggestions* set: the best performing systems are not statistically significantly better or worse than the best ones using *winning suggestions* for most values of the maximum number of suggestions M, but, overall, the models trained with the *winning suggestions* perform statistically significantly better than the heuristic baseline more times than the models trained with the *viable suggestions*.

3.3. Comparison with glass-box ITP

So far, the performance of the black-box ITP approach has been discussed without comparing it to any other ITP system. To put the discussion in the context of the state of the art, it has to be compared to other systems currently being developed. For this mean, the black-box approach will be compared with the free/open-source glass-box ITP toolkit Thot[15] (Ortiz-Martínez and Casacuberta, 2014), as previously described in Chapter 1.4, which provides SMT, and ITP as a particular case of SMT where the system is forced to constrain the translation to a given prefix. Thot's ITP system generates a word graph with probabilities using a modified version of the SMT decoder, and searches for the most probable translation constrained by the prefix already typed according to the word graph; an error-correction algorithm is used if the typed prefix is not in the word graph.

To use the same source of information in both tests, the black-box approach will use Thot SMT as bilingual source to generate the suggestions; the black-box approach will be tested in two different configurations: one offering only up to one suggestion ($M = 1$), and

[14] The automatic procedure is greedy and may not obtain the actual best KSR, but it was proven to be very close to the optimal. See Chapter 2.4.6 for discussion.

[15] `http://daormar.github.io/thot/`

Domain adaptation	In-domain			Development	Out-of-domain	
Glass-box ITP/SMT	Test	-		Development	Train	-
Black-box ITP neural network	-	Train	Development	-		
ITP evaluation	Test	-				
Amount of sentences	3 000	10 000	2 000	2 000	1 000 000†	Remaining sentences†

Table 3.2: Distribution of the corpora. The sentences follow the same order as in the original corpus, except for the sentences tagged with †, which are ordered according to the similarity score of the bitext domain adaptation procedure. The top 1 million sentences for the glass-box training set were selected after filtering with the preprocessing tools in Thot. Each corpora uses the corresponding sentences in the table, e.g. the in-domain corpora for the domain adaptation uses the first 15 000 (3 000 + 10 000 + 2 000) sentences of the corpus; the first 3 000 sentences are also used as test corpus for computing the glass-box model BLEU, and for testing the KSR. The development corpus for the domain adaptation procedure is considered in-domain.

another one offering up to four ($M = 4$). Neither is directly comparable to the glass-box approach, that always offers one suggestion that completes the translation of the whole segment. Additionally, it is hard to assess how human translators will interact with the suggestions, e.g. they may read every suggested word, or stop reading after the first word that does not fit with the planned translation.

In order to check the performance of each approach under different scenarios, more language pairs have been added to the evaluation. Parts of the Arabic–English (ar–en), English–Chinese (en–zh) and English–Spanish (en–es) bitexts from the United Nations Parallel Corpus 1.0 (Ziemski et al., 2016) (Corpus 5) have been used to train the glass-box ITP models and the black-box ITP neural network, as well as to provide a test set for the automatic evaluation. Due to processing resources and time limitations, the size of the corpora used to train the glass-box ITP models had to be reduced (the full corpora has approximately 15 million lines for en–zh, 22 million for en-es and 18 million for en–ar); to this end, the bitext domain adaptation procedure described by Axelrod et al. (2011) was used, as implemented in XenC (Rousseau, 2013). This technique minimizes the impact of reducing the size of the training set by keeping the sentences that are more similar to the ones in the test set (each sentence in the out-of-domain corpus gets a similarity score); that is, by only training with "in-domain" sentences, the system trained with less sentences achieves a closer performance to the one trained with all the sentences, sometimes even improving the performance. The distribution of the corpus is shown in Table 3.2.

Thot's training and development sets were lowercased to reduce data sparsity, and tokenized; those sentence pairs that could hinder the training procedure, such as extremely long sentences (more than 80 words) or sentence pairs with disparate lengths, were removed using the preprocessing tools in Thot, as described in its manual;[16] however, the Stanford Tokenizer[17] was used for the tokenization of Chinese, as the tokenizer in Thot does not support this language. The Simplified Chinese corpus was transliterated to the corresponding

[16] http://daormar.github.io/thot/docsupport/thot_manual.pdf
[17] https://nlp.stanford.edu/software/tokenizer.shtml

sequences for the Pinyin input method using Python's `pinyin 0.4.0`,[18] as Simplified Chinese characters are seldom directly typed using a keyboard, but via a phonetic representation (such as Pinyin) or graphical composition (such as Cangjie input method). Thot was compiled to use IBM2 alignment models, and the training procedure used the parameter values in the user manual; a trigram language model and a maximum phrase length of 10 tokens were used. The reader may refer to the paper by Ortiz-Martínez and Casacuberta (2014) for more information about Thot's architecture. The BLEU (Papineni et al., 2002) scores for the resulting models (computed using the Thot toolkit over the evaluation set) are: 0.49 for en→es, 0.47 for es→en, 0.43 for en→ar, 0.33 for ar→en, 0.23 for en→zh and 0.19 for zh→en.[19]

The glass-box model[20] always offers one suggestion that completes the translation of the whole segment; as the black-box approach, users can accept the full suggestion, or select a prefix of it word by word (by using the `Tab` key). Therefore, the suggestion will be longer at the start of the task, and will shorten as the translation is carried out. The average length of the glass-box model suggestions offered s during the automatic evaluation over the 6 different translation task was 20 words. The black-box model offers at most one suggestion ($M = 1$; if no suggestion is compatible with the typed prefix nothing is offered) with a maximum source subsegment length of $L = 4$; the final length of the suggestion depends on the language pair and the words being translated. On average, the black-box model offered 2.3 words, or 1.4 words if the steps in which no suggestion is offered are also considered. The results obtained when allowing the black-box model to show up to 4 suggestions ($M = 4$) will also be shown, as this is the value used during the human evaluation described in Section 3.5; the black-box model with $M = 4$ shows on average 7.5 words (combining the length of the up to 4 suggestions), or 5 words if those steps where no suggestion is available are considered. In both cases, the user can accept a full suggestion or a prefix of one of them.

The monolingual predictor baseline described in Section 2.3 has also been used. Two different language models were trained:

- One, with the same 1 000 000 sentences used to train the SMT Model, that will be referred as "monolingual".

- A second one trained with the million sentences used to train the SMT system plus the sentences labeled as "remaining sentences" in Table 3.2, that were excluded when training the glass-box ITP/SMT model, that will be referred as "monolingual$_f$". The size of these corpora amounts to 17 969 869 sentences for Arabic–English, 20 969 372 for English–Spanish and 15 420 942 for English–Simplified Chinese.

[18] `https://pypi.python.org/pypi/pinyin`

[19] Even though 1 million sentences are too few for SMT, each of the resulting models used around 6 GB of RAM while on use. This was most of the 8 GB available in the system used for human evaluation; a bigger model would have made the system unusable for the evaluation.

[20] The glass-box strategy makes use of the inner workings of a tightly coupled SMT system, unlike the black-box one, that only uses the output of any bilingual resource.

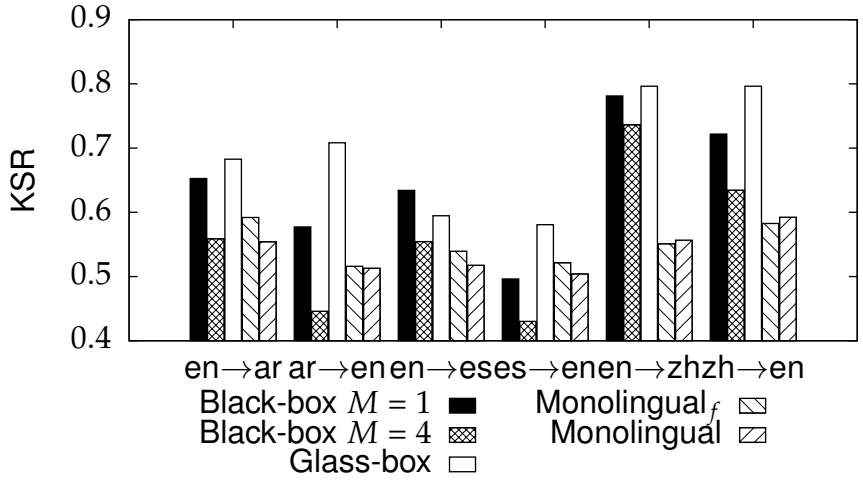


Figure 3.9: KSR values for the automatic evaluation. All differences between the black-box and glass-box values are statistically significant ($p \leq 0.05$); most of the differences between the monolingual predictors and other strategies are statistically significant, unless explicitly noted in the discussion.

Extensive automatic evaluation for all six language pairs with both the black-box and the glass-box approaches has been performed, using all the sentences in the evaluation set described previously. As in the previous automatic evaluations, the statistical significance of the results of the different models was tested using paired bootstrap resampling (Koehn, 2004) with 1000 iterations and $p \leq 0.05$. The results of the automatic evaluation are shown in Figure 3.9. The black-box system using $M = 4$ outperformed the glass-box strategy by a wide margin, even when it had no access to all the information contained in the SMT system and, on average, showed less than half of the words to the user; the black-box system with $M = 1$ still outperformed the glass-box system for every task but en→es, and showed on average less than a fourth of the words of the glass-box approach. The black-box and glass-box approaches have closer performances when translating from English, as the corpora was originally written in English and then translated; for en→es, the translation process is simpler and the glass-box method offers better suggestions.

The monolingual predictor baseline using the full corpus performs:

- Better than both ITP approaches in the English–Simplified Chinese scenario.
- Better than both the ITP approaches in the English→Spanish scenario but for the black-box ITP with $M = 4$, where the statistical significance test fails, meaning they are not meaningfully different.
- Better than both the ITP approaches in the English–Arabic scenario but for the black-box ITP with $M = 4$, which is statistically significantly better than monolingual$_f$.

Even when the monolingual prediction task is harder than the ITP task, the performance of the baseline predictor is great, specially for English–Simplified Chinese, where ITP fails to

obtain good results. Yet, these models are trained on between 15 and 20 times more data than the bilingual resource.[21]

The monolingual predictor using the target side of the corpus used to train the SMT and glass-box ITP model (labeled as monolingual) shows a similar performance pattern:

- Better than both the ITP approaches in the English–Simplified Chinese scenario, albeit worse than monolingual$_f$.
- Better than both the ITP approaches in the English→Spanish scenario.
- Better than both the ITP approaches in the English–Arabic scenario but for the black-box ITP with $M = 4$, which is statistically significantly better than monolingual$_f$ for English→Arabic, and not significantly different for Arabic→English.
- Not significantly different than monolingual$_f$ for Arabic→English.

As the 1 000 000 sentences used to train the SMT model were selected to be the most similar to the test set, the performance of the language model trained with these corpora is similar to the one trained with the full corpora. The results show the same trend as the rest of monolingual prediction evaluations (such as the one in Section 2.4.7): the performance of the language model predictor is similar for all the different scenarios, and beats the ITP approaches for those language pairs that present more problems to SMT (such as English–Simplified Chinese).

3.4. Feature selection

While the current model has a relatively small amount of features (30), feature selection can lead to improvements; particularly as some of them represent similar information, such as the distance features (e.g. f_{22} is f_{10} projected on a different space). The benefits of feature selection are twofold: firstly, it removes potentially misleading features that may lead to an overfitted model or towards local minima; secondly, it reduces both the temporal and spatial complexity of the training and classifying procedure. For these reasons, feature selection algorithms have been used over the full set of features.[22] To this end, a correlation-based feature subset selection algorithm (Hall, 1998) was used, as implemented in Weka (Hall et al., 2009). This greedy method chooses the best feature set by iteratively choosing the feature that both maximizes the correlation with the target class and minimizes the correlation with the other features already in the best feature set. The feature selection algorithm reduced the original 79 features explained in Section 3.1.1 to 4 for en→zh, 13 for zh→en and en→es,

[21] Additional data may be found and added to the model with no problem, as obtaining monolingual data is a trivial task for most languages.

[22] The way feature selection algorithms work is by discarding those features that are redundant (a set of features carrying very similar information) or irrelevant (a feature that carries little to no relevant information); features are discarded until discarding one of them would produce a significant information loss.

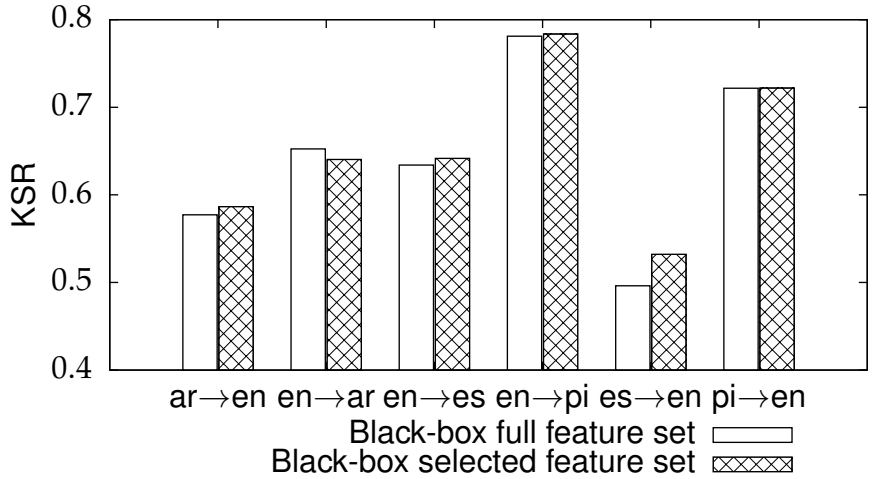


Figure 3.10: KSR values for the automatic evaluation. All differences between the values are statistically significant ($p \leq 0.05$).

and 16 for es→en; and the original 91 features of en-ar to 4 for en→ar and 13 for ar→en. The most frequently selected features are the positive pressure feature, some of the position and length features, and the relationship to the last used suggestion.[23] The performance of the systems using the full and the reduced feature sets that use the same setup as described in the previous Section 3.3 can be seen in Figure 3.10; the difference between the full and the reduced feature sets is minimal (less than 0.05 KSR).

3.5. Preliminary human evaluation

In order to further validate the results of the automatic evaluation of the glass-box and black-box approaches, a human evaluation was performed. This not only supports the results previously obtained, but also provides with metrics for a real world translation scenario. Initially, a preliminary evaluation with amateur translators and a smaller task was performed; later, an evaluation with professional translators was carried out. The users were given the guidelines included in Annex D.

3.5.1. Profile of the users

All eight test subjects U_1–U_8 were computer science researchers that were working in the University of Alacant as technical or research staff at the time.[24] All of them except for U_5 claimed to be experienced typists; experienced typist may write faster, achieving better

[23] Features f_{29}, f_{19} to f_{21} and f_{28} respectively.

[24] The profile of the users is similar to the ones that performed the human evaluation in Chapter 2.5, but no user took part in both tests.

results on the test. All of them were native Spanish speakers, and self-assessed themselves to have an R2/R2+ level (limited working proficiency) of English in the Interagency Language Roundtable scale for reading.[25] None of them had any kind of translation education or was familiar with the domain of the corpora. All of them resorted to using Google Translate[26] to look up the translation of single words or short phrases, except for U_1, who used the online version of the Cambridge English dictionary[27], and U_7, who preferred Linguee[28]. Most users consulted domain-specific terms such as *guidelines*, *compliance* or *interim*. They were supervised by a researcher during the test. The instructions included all the ways they could use the suggestions and stressed that users were not obliged to accept one of the suggestions offered, but that they should also avoid ignoring them altogether. The users had access to the ITP suggestions exclusively: no other kind of CAT (such as translation memories) was available during the test. Still, they were free to consult any resource they needed, such as online dictionaries or MT tools.

3.5.2. Corpora and resources

For this translation task, the same corpora used in the comparison between the black-box approach (Forecat) and a state-of-the-art glass-box approach (Thot) in Section 3.3 were used; the same Thot translation model and the best performing black-box ITP neural network model (*full feature set* with *winning suggestions*, $N = 128$ and $\alpha = 10^{-4}$) were used for the evaluation.

Unlike in the human test presented in Chapter 2.5 (where the web interface of Forecat was used) both approaches were integrated into OmegaT,[29] an open-source CAT tool. OmegaT is widely used by translators, and its interface and workflow is similar to the one of most commercial CAT tools; therefore, by carrying out the test in this platform, the conditions are closer to a real translation task. Both the black-box approach (Forecat[30]) and the glass-box approach (Thot[31]) were integrated into OmegaT as plugins. See Annex A for more information about the software developed and released as part of this thesis.

A preexistent plug-in[32] was used to automatically and non-obstructively log all the actions that the users take while translating. The users were informed that all their actions would be logged so that not only derivative data (such as the final translation or the translation time) would be extracted, but also enough information to reproduce their exact actions (such as key presses, mouse actions, hotkeys used and interaction with the menus). Still, the plugin works

[25] A proficiency scale available at `http://govtilr.org/skills/ILRscale4.htm`

[26] At the time (July 2016), Google still used the SMT-based translation service. Nowadays, it is based on a neural machine translation system. The tool is available at `https://translate.google.com/`.

[27] `http://dictionary.cambridge.org/`

[28] `https://www.linguee.com/`

[29] `https://omegat.org/`

[30] `https://github.com/transducens/Forecat-OmegaT`

[31] `https://github.com/transducens/thot-omegat`

[32] `https://github.com/mespla/OmegaT-SessionLog`

transparently and it is not noticeable during a normal translation task, so users could work without feeling restricted or monitored.

3.5.3. Translation tasks

The first 20 English sentences with lengths between 15 and 25 words in the English–Spanish test set have been selected: this range of lengths excludes those sentences that are too long to be easily understood by non-native speakers and those so short that are either hard to translate isolated from their context or do not present any kind of challenge to the translators. The sentences were arranged in 4 blocks SB_1–SB_4 of 5 sentences each. The 4 blocks were presented to the 8 users using 4 different modalities: the *induction task* let them familiarize with the interface and both suggestion models, and users were encouraged to ask as many questions as they needed on this task; the *unassisted task* offered no suggestions whatsoever; the *black-box task* used the black-box model, offering up to 4 suggestions ranked using the best neural network configuration, and the *glass-box task* used the glass-box model, offering a sentence-completion suggestion using the typed prefix as a constraint. The blocks are distributed so that each block was translated by two users under each different modality.

The suggestions[33] are offered to the users in a drop-down list as they type the translation; these suggestions can then be accepted by selecting them using the arrow keys and pressing the enter key, by using a hot-key combination (`Alt`+p, with p being the position on the list) or with the mouse. Another hot-key (`Tab`) is used to select a prefix of the current suggestion, word by word. All the actions performed by the human translator, such as typing one character or selecting a full suggestion either with the mouse or the keyboard cost one keystroke, but selecting the prefix of a suggestion has a cost of one keystroke per word (`Tab` has to be pressed once per word) in the selected prefix plus one additional keystroke for accepting the prefix.

3.5.4. Results of the human evaluation

3.5.4.1. Quantitative evaluation

For quantifying the performance of the users, several measures were used:

- The *total translation time*, measured in seconds. This corresponds to the time elapsed while they were working in a certain segment. To preserve the truthfulness of the results, users were asked to work exclusively in the segment that was selected: they should avoid performing actions that would add time to a different segment than the one selected, such as advancing to the next segment then proofreading the one they just

[33] The glass-box strategy always offers one suggestion, the completion of the translation. The black-box strategy offers between none and four ($M = 4$) suggestions with a few words each.

SB_1		Time (s)	$\|T\|_c$	$\|T\|_c$/s	Ks	Ks/s	KSR	ESR
Unassisted	U_2	996	666	0.67	996	1.00	1.50	–
	U_6	687	736	1.07	996	1.45	1.35	–
Glass-box ITP	U_3	523	603	1.15	830	1.58	1.38	0.74
	U_7	467	604	1.29	583	1.25	0.97	0.76
Black-box ITP	U_4	715	567	0.79	747	1.04	1.32	0.68
	U_8	601	581	0.97	717	1.19	1.23	0.70

SB_2		Time (s)	$\|T\|_c$	$\|T\|_c$/s	Ks	Ks/s	KSR	ESR
Unassisted	U_1	527	637	1.21	996	1.89	1.56	–
	U_5	477	677	1.42	690	1.45	1.02	–
Black-box ITP	U_2	626	636	1.02	686	1.10	1.08	0.71
	U_6	642	631	0.98	686	1.07	1.09	0.67
Glass-box ITP	U_3	575	570	0.99	537	0.93	0.94	0.75
	U_7	466	547	1.17	548	1.18	1.00	0.65

SB_3		Time (s)	$\|T\|_c$	$\|T\|_c$/s	Ks	Ks/s	KSR	ESR
Black-box ITP	U_1	613	677	1.10	686	1.12	1.01	0.62
	U_5	542	639	1.18	686	1.26	1.07	0.65
Glass-box ITP	U_2	732	618	0.84	819	1.12	1.33	0.68
	U_6	605	635	1.05	819	1.35	1.29	0.77
Unassisted	U_4	667	606	0.91	782	1.17	1.29	–
	U_8	594	644	1.08	783	1.32	1.22	–

SB_4		Time (s)	$\|T\|_c$	$\|T\|_c$/s	Ks	Ks/s	KSR	ESR
Glass-box ITP	U_1	513	615	1.20	819	1.60	1.33	0.49
	U_5	524	595	1.13	819	1.56	1.38	0.67
Unassisted	U_3	297	646	2.17	765	2.57	1.18	–
	U_7	396	660	1.67	681	1.72	1.03	–
Black-box ITP	U_4	478	612	1.28	661	1.38	1.08	0.69
	U_8	392	647	1.65	807	2.06	1.25	0.66

Table 3.3: Performance of the users with the different sentence blocks for the unassisted task (without a mark), the black box task (○) and the glass box task (□).

Unassisted		Time (s)	$\|T\|_c$	$\|T\|_c$/s	Ks	Ks/s	KSR	ESR
SB_2	U_1	527	637	1.21	996	1.89	1.56	–
	U_5	477	677	1.42	690	1.45	1.02	–
SB_1	U_2	996	666	0.67	996	1.00	1.50	–
	U_6	687	736	1.07	996	1.45	1.35	–
SB_4	U_3	297	646	2.17	765	2.57	1.18	–
	U_7	396	660	1.67	681	1.72	1.03	–
SB_3	U_4	667	606	0.91	782	1.17	1.29	–
	U_8	594	644	1.08	783	1.32	1.22	–

Black-box ITP		Time (s)	$\|T\|_c$	$\|T\|_c$/s	Ks	Ks/s	KSR	ESR
SB_3	U_1	613	677	1.10	686	1.12	1.01	0.62
	U_5	542	639	1.18	686	1.26	1.07	0.65
SB_2	U_2	626	636	1.02	686	1.10	1.08	0.71
	U_6	642	631	0.98	686	1.07	1.09	0.67
SB_1	U_3	523	603	1.15	830	1.58	1.38	0.74
	U_7	467	604	1.29	583	1.25	0.97	0.76
SB_4	U_4	478	612	1.28	661	1.38	1.08	0.69
	U_8	392	647	1.65	807	2.06	1.25	0.66

Glass-box ITP		Time (s)	$\|T\|_c$	$\|T\|_c$/s	Ks	Ks/s	KSR	ESR
SB_4	U_1	513	615	1.20	819	1.60	1.33	0.49
	U_5	524	595	1.13	819	1.56	1.38	0.67
SB_3	U_2	732	618	0.84	819	1.12	1.33	0.68
	U_6	605	635	1.05	819	1.35	1.29	0.77
SB_2	U_3	575	570	0.99	537	0.93	0.94	0.75
	U_7	466	547	1.17	548	1.18	1.00	0.65
SB_1	U_4	715	567	0.79	747	1.04	1.32	0.68
	U_8	601	581	0.97	717	1.19	1.23	0.70

Table 3.4: Performance of the users with the different tasks. The dashed lines split the different blocks.

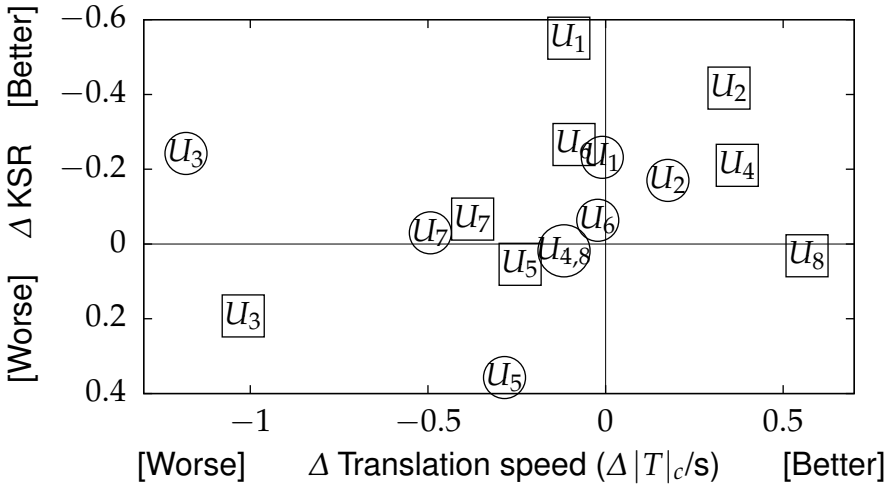


Figure 3.11: Absolute increase of KSR and $|T|_c$/s of the glass-box (○) and the black-box (□) tasks against the unassisted task. U_4 and U_8 got grouped because they attained very similar performances with the glass-box system.

translated, or reading and planning the translation of the next segment before selecting it.

- The *size*, in characters, of the final translation the users provided ($|T|_c$)
- The *number of keystrokes*, including insertions, deletions, cursor movements and the usage of suggestions (KS).
- The *keystroke ratio*, KSR = KS$/|T|_c$.
- The *translation speed*, measured in characters per second ($|T|_c$/s).
- The *emulated keystroke ratio* (ESR): the KSR obtained by the automatic procedure described on the previous Section 3.2.1 using the same conditions under which the human test was carried and the translation provided by the user as a reference. The ESR can be seen as an optimistic lower bound[34] for the KSR, as it assumes that the users took all the viable suggestions and made no mistakes while typing the translation, that is, that no inserted character was deleted.

The results of the human evaluation are shown in Table 3.3 and Table 3.4;[35] the rows corresponding to the induction task are blank, as those results are not relevant: users could use as much time as they wanted in this task, and proceeded to the unassisted task when they

[34]The automatic evaluation approach used to evaluate the system follows a greedy approach, what means that the obtained solution is suboptimal; an optimal solution with a lower cost could have been found if a exhaustive strategy was used instead. Even thought, the actual difference between the greedy and exhaustive approaches was computed in Chapter 2.4.6, and was found to be quite small.

[35]Both tables contain the same information, but presented differently: the first table makes the comparison between text blocks easier; the second table makes the comparison between tasks easier.

decided to; some translated all the segments, while others tested the different approaches in one or two segments. An analysis of the differences in translation speeds and KSR of each method and user is shown in Figure 3.11. Only U_2 managed to translate both faster (measured in $|T|_c$/s) and with less effort (measured in KSR) with both techniques; U_4 managed to do so only with the black-box method. On average, when compared to the unassisted task, the evaluators saved 10% keystrokes and were 4% faster with the black-box approach, and saved 15% keystrokes and were 12% slower with the glass-box one; black-box suggestions proved therefore less useful at saving keystrokes, but they allowed translators to translate faster.

ESR values show that the users could have theoretically saved 51% and 69% respectively if they had used the compatible suggestions and made no mistakes while typing, compared to their actual performance. Users only had a few minutes to familiarize with OmegaT and the assistance technologies, and it could be expected that their performance would rise as they get more familiar with the tools. A recent study by Autodesk[36] considers experience the most single important factor in translation productivity; as the users get more experienced, the gap between KSR and ESR will close: not only users will make less mistakes and better avoid rethinking their translations, but also they will trust the suggestions more, leading to an incremented suggestion usage and a reduction of the translation effort.

After the tests, users were asked to sort the tasks according to their perceived speed of translation and ease of translation. Users U_1, U_4 and U_8 perceived the black-box system as faster and more helpful than the others; the remainder preferred the glass-box system; U_4 thought the glass-box system led to faster translations, yet it made the translation task harder than without assistance; finally, U_5 thought the black-box system made the task both harder and slower. Users' perceptions strongly contrasted with the measurements: only U_2 was faster with both methods compared to unassisted translation (0.67 $|T|_c$/s), though glass-box (0.84 $|T|_c$/s) was incorrectly perceived to be faster than black-box (1.02 $|T|_c$/s); and U_4 correctly ranked black-box (1.28 $|T|_c$/s) as the fastest task. Users have a hard time assessing their real translation speed, as they feel like they are working harder (and faster) when reading, accepting and discarding suggestions, while, when measured, they actually translate slower due to this same actions. Slower users (such as U_2) get the most benefit from the assistance, and fast users (such as U_3) get hampered by the suggestions.

3.5.4.2. Qualitative evaluation

Finally, the users were asked to provide some open feedback. User U_4 strongly disliked the OmegaT tool. Users were presented with a unassisted task after experimenting with the induction task, where both kinds of assistance were active; they felt annoyed due the lack of suggestions, and some expressed that the unassisted task was the hardest one, even when the sentences themselves were different from user to user. Most of them expressed that the first full-sentence suggestion the glass-box system provided was very useful for planning the translation; these suggestions were equivalent to looking for a whole-sentence MT translation,

[36] `http://langtech.autodesk.com/productivity.html`

which is very useful for non-professional translators. However, some complained those suggestions were too long and unwieldy.

Some users complained about suggestions being offered too often, specially when none of them were useful. Some users praised the tool as they were able to operate it using only the keyboard; they all are experienced coders and most work in environments operable without a mouse. However, none of them used the `Alt+`p option for accepting specific suggestions from the drop-down list; the temporal cost of using the arrows to highlight the suggestion to insert is fairly low, and the users were not actively seeking to minimize the number of keystrokes. The option for using the prefix of a suggestion by pressing `Tab` was never used in the black-box block, as most suggestions were either totally useful or not useful at all, and the temporal cost of deleting the suffix of a selected translation is fairly low; however, when they reached the glass-box block, they started to use this option, as most times the suggestions were too long to be useful and required to delete a large suffix, but the first few words could be easily selected by using the `Tab` key.

3.6. Confidence thresholds

A common complaint in the preliminary human evaluation was the overabundance of suggestions. In the previous experiment, the $M = 4$ most promising suggestions (the ones that get a higher score from the neural network) are being offered to the user; less suggestions could be shown by reducing the value of M, but by doing this:

- If all the suggestions in $P_C^S(W_n^m)$ have low goodness scores $g(p, W_n^m)$, less promising suggestions will be suggested.
- If more than M suggestions in $P_C^S(W_n^m)$ have high goodness scores, promising suggestions will not be suggested.

In order to avoid this, a threshold will be used: the value of $M = 4$ will be kept, but only suggestions that have a goodness score higher than some threshold will be offered.

Using the same conditions and resources as in Section 3.3, the distribution of goodness scores given by the neural network model to the winning suggestions from the automatic evaluation was extracted; from this distribution, the vigintiles (the 5% quantiles)[37] were computed, and automatic evaluation with each quantile as threshold was performed. Figure 3.12 shows an example of distribution of goodness scores of the winning suggestions and the quantiles.

[37]The values of the goodness score of the 18 suggestions that split the distribution in 20 equally sized groups; as several suggestions have the same score, it is possible than some groups have more elements than others, or even that two contiguous quantiles have the same value; e.g. if we have the distribution $[1, 1, 1, \mathbf{1}, 1, 1, 2]$, the median (the 50% quantile) corresponds to the bolded element, but its value splits the sample in two uneven groups, one with six elements and one with one.

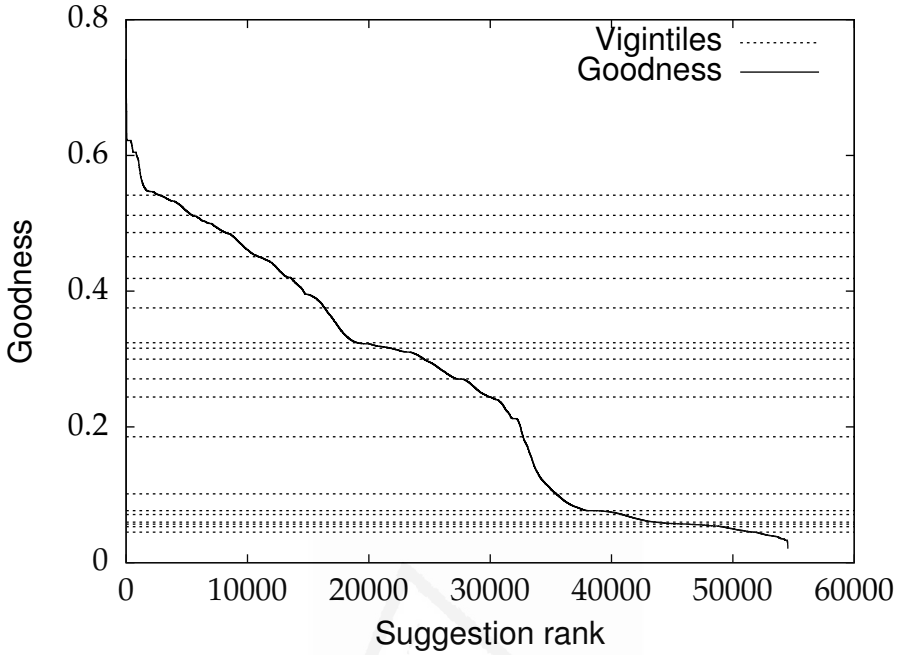


Figure 3.12: Goodness of each winning suggestion in the test set and the vigintiles for English→Spanish.

Results for the tests for English–Spanish, English–Arabic and English–Simplified Chinese can be seen in Figure 3.13; each datapoint corresponds to a threshold, with the point with the smaller ASR and KSR being the one without a threshold, and the point at the other end of the line the one with the most strict threshold (the 19th quantile, that is, only the suggestions whose score is in the 5% top are offered). Some graphs may have less than 20 points: some quantiles have the same KSR threshold, as previously explained. In general, using stricter thresholds leads to an increased ASR and a decreased KSR: this means less suggestions groups where no suggestion was useful were offered to the user, but the user needed more keystrokes to complete the translation. Some parts of the graphs show a different behaviour: when increasing the threshold, the ASR improves, but the KSR is only slightly affected (if at all); for example, in the English→Arabic plot, the ASR improves from 0.45 to 0.75 while the KSR keeps around 0.70 as the threshold increases; this means the user goes from seeing a 54% of sugggestion lists with no viable suggestion to only a 20% without major KSR differences, what would make the user experience more pleasant. The opposite also happens: in the same graph, if the threshold is increased, the KSR increases (from 0.72 to 0.85) while the ASR decreases slightly (from 0.8 to 0.78); this means the user had to use more keystrokes to type the translation.

Reducing the ASR is beneficial for the user, as less suggestion groups where no suggestion is useful will be shown, instilling user confidence, but reducing the KSR means that the user will have to type more, leading to a potentially less useful system. A hypothesis is drawn from this observations:

- That fast-typing translators (due to high typing speed, deep knowledge of the languages or domain involved or other factors) will benefit more from stricter thresholds that offer very few high-probability suggestions; they will lose less time reading the suggestions both because there would be less suggestions offered, and because, as those would be the highest quality ones, would be likely to be accepted with less considerations from the user.
- That slow-typing translators will benefit more from looser thresholds, that allow more suggestions to be offered; the user can afford to spend more time reading suggestions, as it would have taken more to type the text of the accepted suggestion than reading all the suggested ones.
- Looser thresholds may also be good for translators that do not have a good knowledge of the domain or language pair, as the translation alternatives presented may introduce the user to previously unknown translations for some terms or better options than the initially planned translation.

The hypothesis will be tested in the next section, where a strategy using a fixed threshold will be used in the test with professional translators.

3.7. Evaluation with professional translators

To further validate the results obtained both during the automatic evaluation in Section 3.3 and the preliminary human evaluation in Section 3.5, an evaluation round using professional translators performing an English to Spanish translation task was designed and carried out. Even when the automatic results were already corroborated with untrained translators, professional translators may work in a different way, obtaining a different performance. The black-box approach tries to assist both amateur and professional translators when performing a translation task, but each user group has different needs that may require to modify how the assistance works.

The design of this evaluation was inspired in the human evaluations performed on different ITP tools in the state-of-the-art, that are described in AnnexB. The users were given the guidelines included in Annex E.

3.7.1. Profile of the users

For this task, 8 professional translators Z_1–Z_8 with different backgrounds were selected and hired by Soleil Traducciones.[38] All of them are native Spanish speakers, except for Z_3, whose mother language is Flemish, but has lived in Spain for 20 years and is highly proficient in Spanish. All of them are used to work with OmegaT. A detailed profile of the users and

[38] http://soleiltraducciones.com

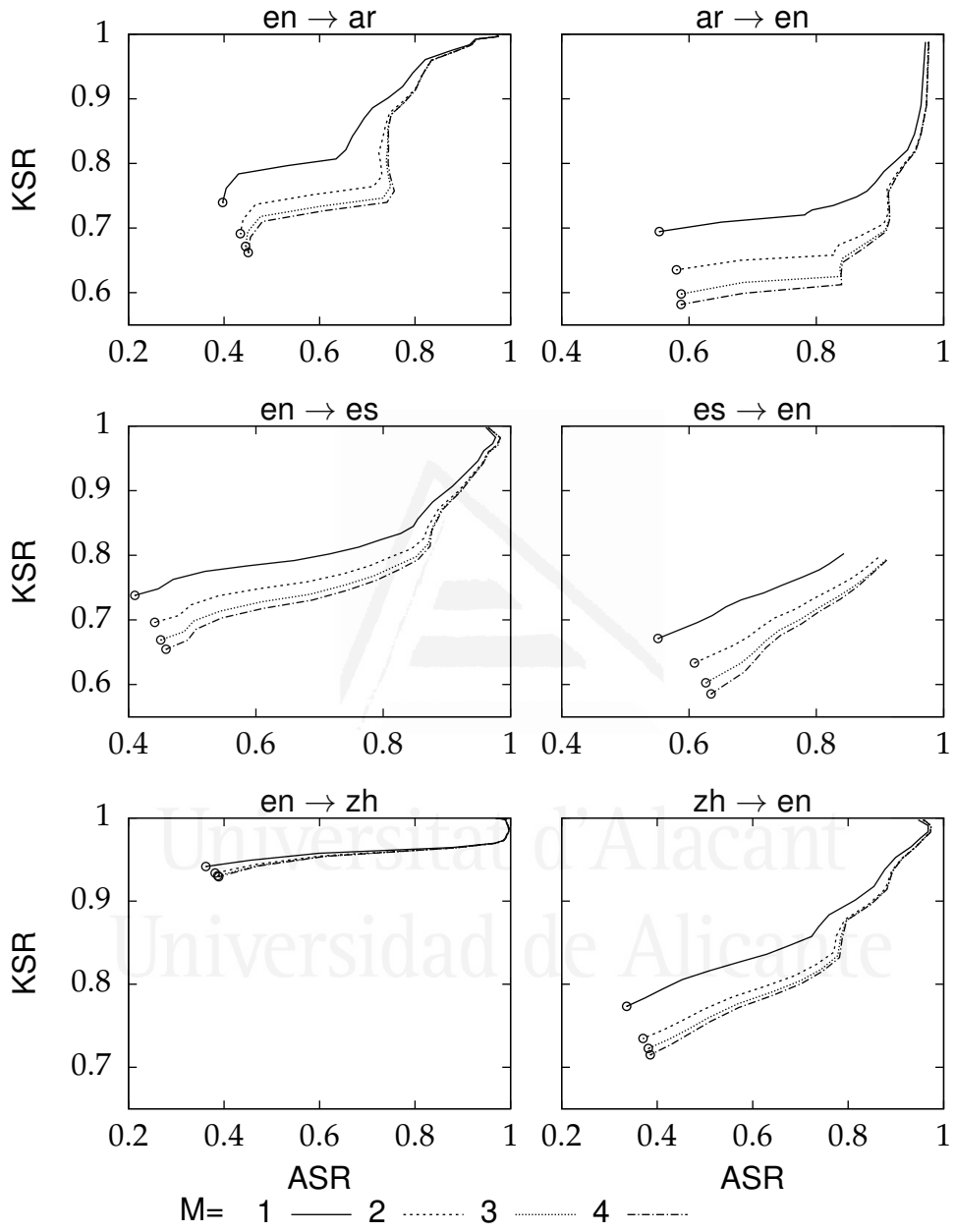


Figure 3.13: Results for the different thresholds. The data points start with a threshold of 0 (that is, no suggestions are removed) in the bottom left corner (marked with ○), and evolve towards the last vigintile moving, usually, towards the top right corner; that is, stricter thresholds usually lead to increasing the KSR (meaning more keystrokes were needed to complete the translation) and increased ASR (that is, less suggestions lists with no viable suggestions were offered). The line with the worse values (higher KSR and lower ASR) is $M = 1$; the one with the better values is $M = 4$.

	User	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	Z_8
	Translation experience	10y	15y	15y	2y	3m	10y	11y	10y
Used resources	Linguee	X	X	X	X	X	X	X	X
	IATE	X	X	X					
	Wikipedia	X	X				X		
	Google Translate	X			X	X	X	X	X
	DRAE		X						
	WordReference				X	X	X		
	Collins Dictionary					X			

Table 3.5: Profile of the users and the resources they used during the test. All the users have continuously worked as translators, but for Z_4.

the resources they used for the test when translating can be seen in Table 3.5. The translators were free to use any resource they were used to, but excluding systems that needed to be installed in the local machine; they used the following resources:

- Linguee:[39] a bilingual dictionary that offers translations for the different meanings of each word.
- IATE:[40] the European multilingual database.
- Wikipedia:[41] the free online encyclopedia Wikipedia. While it is not a translation resource, many articles are available in multiple languages; the user can look for a specific term in source language then visit the target language article to obtain an accurate translation.
- Google Translate:[42] a web service that uses neural machine translation.
- DRAE:[43] The *Diccionario de la lengua española* (Spanish language dictionary) is a monolingual Spanish dictionary curated by the *Real Academia Española*, the official institution that regulates the Spanish language.
- WordReference:[44] Wordreference is a extended multilingual dictionary that offers multiple resources to help translators, such as the different meanings of the word with the most common translation and compound forms.
- Collins Dictionary:[45] A multilingual dictionary curated by the education company Collins.

[39] https://www.linguee.com/
[40] http://iate.europa.eu
[41] https://www.wikipedia.org/
[42] https://translate.google.com/
[43] http://dle.rae.es/
[44] http://www.wordreference.com
[45] https://www.collinsdictionary.com/

Domain adaptation	News Commentary					United Nations
SMT	-			Development	Train	
Black-box ITP neural network	Train	Development	-			
Evaluation	-		Test	-		
Sentences	10 000	2 000	15 000	2 000	162 441	7 835 559 †

Table 3.6: Distribution of the corpora used for training the models that were used for human evaluation and for extracting the test sentences. The sentences follow the same order as in the original corpus, except for the sentences tagged with †, which are ordered according to the similarity score of the bitext domain adaptation procedure. The top 7 835 559 sentences of the United Nations for moses training set were selected after filtering with the preprocessing tools in moses. The total size for the filtered Moses train set is 8 million sentences.

3.7.2. Corpora and resources

For this evaluation round, sentences extracted from the News Commentary corpus (Corpus 3) and the United Nations corpus (Corpus 5) were used. The United Nations Corpus was scored and sorted according to their similarity to the sentences in the News Commentary corpus, according to the bitext domain adaptation procedure described in the Section 3.2.2. The distribution of the sentences can be seen in Table 3.6.

A baseline Moses system[46] was trained using parts of the corpora tagged as Moses Train and Development in Table 3.6. Some additional steps were performed: the language model was binarized[47] and the phrase table[48] and the lexical reordering tables[49] were compacted (Johnson et al., 2007).[50]

The suggestions that were used in the human evaluation task were generated beforehand and kept in a cache file; this procedure makes all the suggestions available in a very short timespan (in the order of ten milliseconds), and can be realistically applied to any task performed using the black-box approach. The CacheTrans-OmegaT plugin was used to provide this functionality. See Annex A for more information about the software developed and released as part of this thesis.

3.7.3. Translation tasks

As in the previous task, OmegaT and the Forecat-OmegaT plugin were used. No other kind of assistance was enabled, except for the translation memory that gets automatically built as the segments of a file get translated and cannot be deactivated; still, the translators were instructed to ignore the translation memory information, as there were no fuzzy matches

[46] Trained as explained in `http://www.statmt.org/moses/?n=Moses.Baseline`

[47] `http://www.statmt.org/moses/?n=Moses.Optimize#ntoc14`

[48] `http://www.statmt.org/moses/?n=Advanced.RuleTables#ntoc3`

[49] `http://www.statmt.org/moses/?n=Advanced.RuleTables#ntoc4`

[50] Otherwise, the translation model used more than the 64Gb of RAM available on the system that performed the translations.

with a fuzzy score over the default threshold. The SessionLog-OmegaT plugin[51] was used to transparently log all the actions of the users; the process is unnoticeable for them, but they were informed that their actions inside OmegaT would be logged with enough detail to reconstruct their exact actions, and that these sequences of actions will be used in the discussion of this task.

The sentences for the different translation tasks were shuffled, as, if the sentences formed a discourse, translators could experience an increase in translation speed solely due to being more accustomed to the domain, e.g. the specific vocabulary used in the block of segments is retrieved faster and easier from memory, specially if they had to check a translation resource the first time they saw the term. By not keeping a discourse, users are prevented from using any short-term memory they may have built while translating the previous sentences.

The sentences were further filtered in order to remove those sentences that may be exceedingly challenging to translate on their own:

- Sentences with less than 20 tokens were discarded, as they may not be long enough to carry enough information to be translated as isolated units.
- Sentences with more than 80 tokens were discarded, as they may be too convoluted and severely hamper the performance of the users.
- Sentences with any kind of acronyms, initialisms or uncommon abbreviations were manually removed, as they could confuse and distract the users.
- Sentences with ambiguous pronouns were manually removed, as they may be too complex to translate without reading the actual context of the segment.

The sentences were split into six different sentence blocks: three induction blocks (PB_1, PB_2, PB_3) with 20 sentences each, and three testing blocks (TB_1,TB_2,TB_3) with 100 sentences each. The induction blocks are intended to let the users warm up and get used to the different kinds of assistance

There were six different tasks: the three induction tasks (PT_1,PT_2,PT_3) and the three translation tasks (TT_1,TT_2,TT_3). The total task duration was three hours, split as following:

- 5 minutes for the induction task 1 (PT_1), without any kind of assistance, so that the users could warm up. All the induction tasks (PT_1, PT_2 and PT_3) were logged, but the logged data was not used.
- 40 minutes for the task 1 (TT_1), without any kind of assistance.
- a 15-minute break.
- 5 minutes for the induction task 2 (PT_2), with ITP assistance: up to $M = 4$ suggestions, ranked according to the best neural network model.

[51] https://github.com/mespla/OmegaT-SessionLog

	PT_1	TT_1	PT_2	TT_2	PT_3	TT_3
Z_1, Z_2, Z_3	PB_1	TB_1	PB_2	TB_2	PB_3	TB_3
Z_4, Z_5, Z_6	PB_2	TB_2	PB_3	TB_3	PB_1	TB_1
Z_7, Z_8	PB_3	TB_3	PB_1	TB_1	PB_2	TB_2

Table 3.7: Distribution of the text blocks for each user and task.

- 40 minutes for the task 2 (TT_2), with the same kind of assistance as in PT_2.
- 15 minutes for answering a short questionnaire and break.
- 5 minutes for the induction task 3 (PT_3), with ITP assistance and threshold: up to four suggestions, ranked according to the best neural network model, and a threshold of 0.27, the value of the first quartile. With the threshold set in the first quartile, the users will get around 25% of the suggestions, when comparing with PT_2 and TT_2.
- 40 minutes for the task 3 (TT_3), with the same kind of assistance as in PT_3.
- 15 minutes for answering a short questionnaire and a longer one with freeform answers.

As in the preliminary human test described in Section 3.5, the users were shown how to interact with the suggestions, specially when dealing with the drop-down list where the suggestions appeared, as none of them was used to work with it. Details about the exact behaviour of each assistance were not given: they did not know TT_3 would have less suggestions than TT_2, or how the suggestions were created, sorted and offered; they did know that TT_1 would be used as an unassisted benchmark.

The blocks were distributed so that every block is translated by, at least, two different users as part of the same task, e.g., the block TB_1 will be translated without assistance by the users Z_1, Z_2 and Z_3, with ITP assistance by Z_7 and Z_8, and with ITP assistance with a threshold by Z_4, Z_5 and Z_6. The distribution can be seen in Table 3.7. It was made clear that they did not have to translate all the 100 sentences in that timespan, but to translate a subset of them at a comfortable speed. Before the test, it was estimated that they would translate, at least, 10 sentences from each task. In the actual test, users translated between 16 and 49 sentences.

3.7.4. Results of the human evaluation

3.7.4.1. Quantitative evaluation

The same metrics used in Section 3.5.4 will be used to measure the performance of the users. The detailed results for each user and task can be seen in Table 3.8 and Table 3.9.[52]

[52]Both tables contain the same information, but presented differently: the first table makes the comparison between text blocks easier; the second table makes the comparison between tasks easier.

As previously explained, users were expected to translate at least 10 sentences per task. For this reason, two different comparisons will be made: one for the performance of the first 10 sentences, and another one for the performance of the complete task. In this evaluation, users managed to translate between 16 and 49 sentences per task.

While the rest of the users typed the translations in TT_1, Z_5 copied and pasted extensive blocks of automatically translated text using the bilingual resources, specially Google Translate; as the corpus used for the tests has been available for a long time, and used in different state of the art SMT papers and competitions, it is presumed that it was used to train the NMT model of Google Translate; hence, the machine translation output was of high quality, and "post-editing" like Z_5 did proved to be highly productive. However, Z_2 did not follow this strategy and obtained a similar translation speed to Z_5. Due to a error in the setup process, Z_6 could not start TT_2 at the same time as the rest of users; the issue took a bit of time to be fixed, what resulted in the task time being reduced from 40 to 35 minutes.

The increase in translation speed and KSR can be seen in Figure 3.14, and the same metrics for the first 10 segments (the estimated minimum number of segments the translators would be able to translate during the task) can be seen in Figure 3.15. User Z_5 is not included in the graphs due to the different approach in TT_1; Z_5 translated slower (50% slower) and using more keystrokes (80%) when translating with assistance than when translating without. Still, as seen in Table 3.9, Z_5 obtains the best KSR and the third best translation speed in TT_2.

In TT_2, all users' (but Z_5) KSR improve, and three (Z_3, Z_4 and Z_6) manage to translate faster. Therefore, it can be concluded that the tool helped some translators to translate faster and with less effort. Moreover, some users improved the performance as they moved on from the first 10 sentences (Figure 3.15) to all the sentences they managed to translate (Figure 3.14): only Z_6 obtains a worse translation speed and KSR when comparing the first 10 segments and all of them; the rest of users kept or improved both the metrics.

The ESR values show that, if the users had made no mistake and used all the compatible suggestions, they could have saved between 25% and 65% keystrokes over their obtained results. It is expected that the gap between ESR and KSR closes as the users get more used to the tool, but not totally, as a part of that difference comes from typing mistakes and rethought translations.

The results in TT_2 were promising, but TT_3 shows that the threshold strategy was not well received by the users. Four users managed to improve the translation speed (Z_2, Z_3, Z_4 and Z_6), but all the users had to use more keystrokes than in TT_2. The user that managed to save the most keystrokes in TT_3 (Z_7) did worse than the one that saved the least keystrokes in TT_2 (Z_1). A comparison of the percentage of text that was inserted by accepted suggestions compared to the total length of the sentence can be seen in Figure 3.16. Additionally, most users disliked this task; this will be further discussed in the qualitative evaluation the next section.

TB_1		Sentences	Time	$\|T\|_c$	$\|T\|_c/s$	KS	KS/s	KSR	ESR
TT_1	Z_1	37	2357	5199	2.21	5991	2.54	1.15	0.67
	Z_2	47	2473	7016	2.84	8199	3.32	1.17	0.71
	Z_3	20	2326	3268	1.41	3922	1.69	1.20	0.74
TT_3	Z_4	37	2063	5983	2.90	5882	2.85	0.98	0.59
	Z_5	28	2132	4365	2.05	3743	1.76	0.86	0.54
	Z_6	16	1907	2728	1.43	3545	1.86	1.30	0.61
TT_2	Z_7	23	2398	3642	1.52	4847	2.02	1.33	0.78
	Z_8	24	2334	3804	1.36	4995	1.78	1.31	0.82

TB_2		Sentences	Time	$\|T\|_c$	$\|T\|_c/s$	KS	KS/s	KSR	ESR
TT_2	Z_1	28	2121	4321	2.04	4337	2.04	1.00	0.66
	Z_2	40	2125	6014	2.83	6109	2.88	1.02	0.69
	Z_3	24	2136	3657	1.71	3053	1.43	0.83	0.57
TT_1	Z_4	49	2276	6890	3.03	8319	3.66	1.21	0.78
	Z_5	30	2282	4355	1.91	5662	2.48	1.30	0.79
	Z_6	18	2141	3168	1.48	5257	2.46	1.66	0.84
TT_3	Z_7	26	2386	4318	1.81	6180	2.59	1.43	0.65
	Z_8	23	2385	3766	1.58	5044	2.11	1.34	0.71

TB_3		Sentences	Time	$\|T\|_c$	$\|T\|_c/s$	KS	KS/s	KSR	ESR
TT_3	Z_1	31	2314	4964	2.14	6045	2.61	1.22	0.81
	Z_2	44	2272	7276	3.20	8292	3.65	1.14	0.83
	Z_3	23	2273	3709	1.63	4434	1.95	1.20	0.84
TT_2	Z_4	44	2371	6490	2.74	7996	3.37	1.23	0.62
	Z_5	48	2345	6941	2.96	1032	0.44	0.15	0.58
	Z_6	19	2343	2992	1.28	4910	2.10	1.64	0.73
TT_1	Z_7	24	2300	3767	1.60	4504	1.91	1.20	0.62
	Z_8	16	2291	2904	1.27	3283	1.43	1.13	0.68

Table 3.8: Performance of the users with the different text blocks. The dashed lines separate the groups of users with the same task in the text block. The ESR column for TT_1 was computed using the same assistance as in TT_2; the ESR without assistance would always be 1, as the automatic evaluation system perfectly types the sentence without mistakes. The ESR column for TT_3 uses the same threshold as the users.

TT_1		Sentences	Time	$\|T\|_c$	$\|T\|_c$/s	KS	KS/s	KSR	ESR
TB_1	Z_1	37	2357	5199	2.21	5991	2.54	1.15	0.67
	Z_2	47	2473	7016	2.84	8199	3.32	1.17	0.71
	Z_3	20	2326	3268	1.41	3922	1.69	1.20	0.74
TB_2	Z_4	44	2371	6490	2.74	7996	3.37	1.23	0.62
	Z_5	48	2345	6941	2.96	1032	0.44	0.15	0.58
	Z_6	19	2343	2992	1.28	4910	2.10	1.64	0.73
TB_3	Z_7	26	2386	4318	1.81	6180	2.59	1.43	0.65
	Z_8	23	2385	3766	1.58	5044	2.11	1.34	0.71

TT_2		Sentences	Time	$\|T\|_c$	$\|T\|_c$/s	KS	KS/s	KSR	ESR
TB_2	Z_1	28	2121	4321	2.04	4337	2.04	1.00	0.66
	Z_2	40	2125	6014	2.83	6109	2.88	1.02	0.69
	Z_3	24	2136	3657	1.71	3053	1.43	0.83	0.57
TB_3	Z_4	37	2063	5983	2.90	5882	2.85	0.98	0.59
	Z_5	28	2132	4365	2.05	3743	1.76	0.86	0.54
	Z_6	16	1907	2728	1.43	3545	1.86	1.30	0.61
TB_1	Z_7	24	2300	3767	1.60	4504	1.91	1.20	0.62
	Z_8	16	2291	2904	1.27	3283	1.43	1.13	0.68

TT_3		Sentences	Time	$\|T\|_c$	$\|T\|_c$/s	KS	KS/s	KSR	ESR
TB_3	Z_1	31	2314	4964	2.14	6045	2.61	1.22	0.81
	Z_2	44	2272	7276	3.20	8292	3.65	1.14	0.83
	Z_3	23	2273	3709	1.63	4434	1.95	1.20	0.84
TB_1	Z_4	49	2276	6890	3.03	8319	3.66	1.21	0.78
	Z_5	30	2282	4355	1.91	5662	2.48	1.30	0.79
	Z_6	18	2141	3168	1.48	5257	2.46	1.66	0.84
TB_2	Z_7	23	2398	3642	1.52	4847	2.02	1.33	0.78
	Z_8	24	2334	3804	1.36	4995	1.78	1.31	0.82

Table 3.9: Performance of the users with the different task blocks. The dashed lines separate the groups of users with the same blocks of text in the task. The ESR column for TT_1 was computed using the same assistance as in TT_2; the ESR without assistance would always be 1, as the automatic evaluation system perfectly types the sentence without mistakes. The ESR column for TT_3 uses the same threshold as the users.

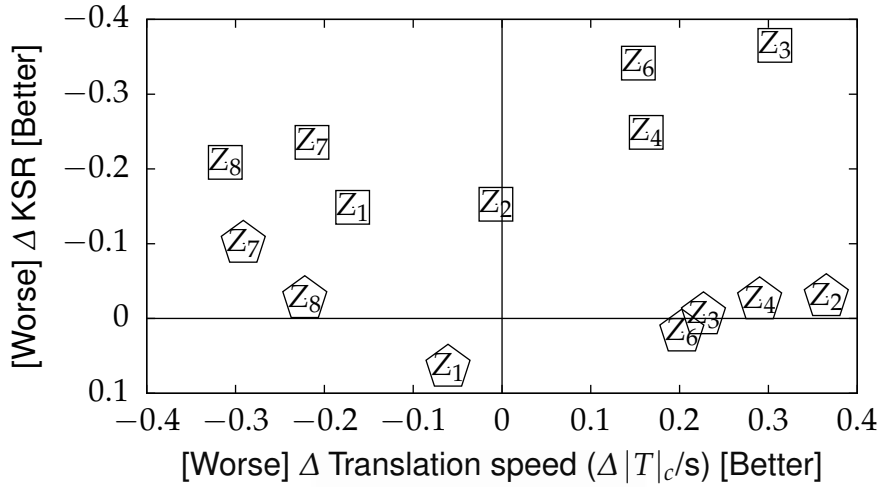


Figure 3.14: Absolute increase of KSR and $|T|_c$/s of the assisted task (□) and assisted task with thresholds (⬠) against the unassisted task. The KSR of the task with thresholds is worse than the one without; two users (Z_1 and Z_6) obtain worse results than with the unassisted task. Z_2, Z_3, Z_4 and Z_6 translate faster with assistance.

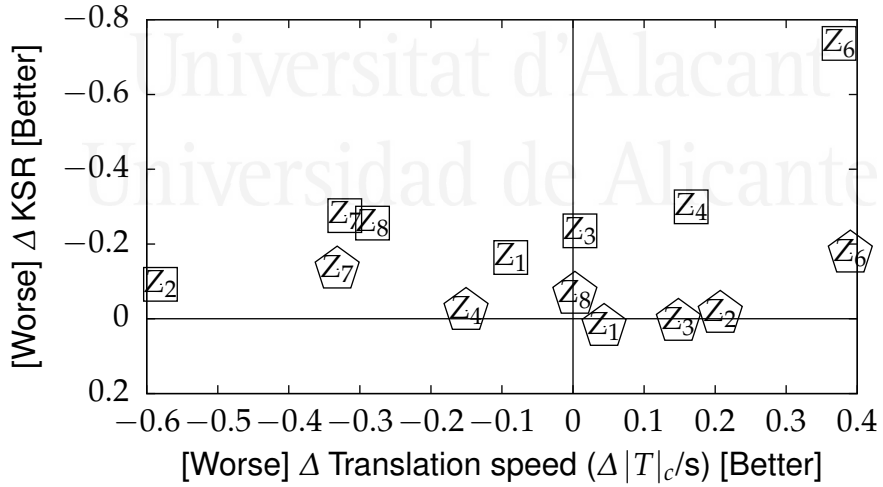


Figure 3.15: Absolute increase of KSR and $|T|_c$/s of the assisted task (□) and assisted task with thresholds (⬠) tasks against the unassisted task for the first 10 sentences. The values are similar to the ones obtained from all the sentences, with one exception: Z_2. The difference comes from the unassisted task performance; the user translated the first 10 unassisted sentences notably faster (3.12 $|T|_c$/s) than the first 10 assisted ones (2.54 $|T|_c$/s), but obtained the same speed (2.83 $|T|_c$/s) when taking into account all the sentences.

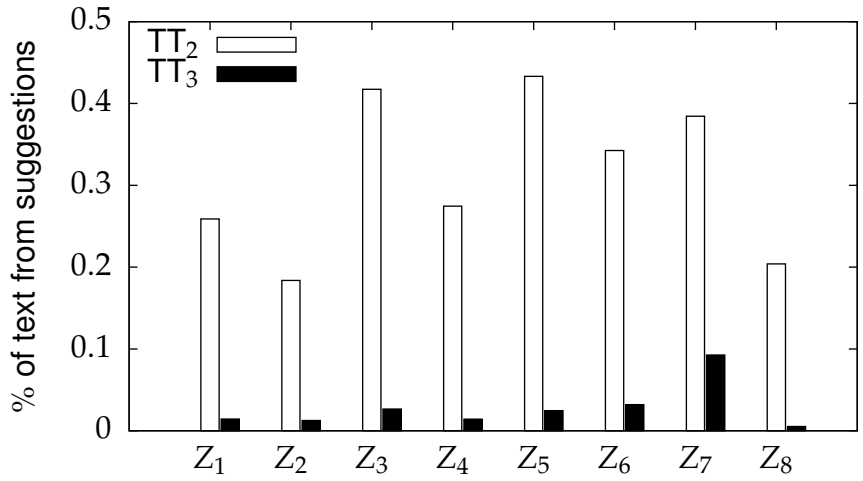


Figure 3.16: Percentage of text inserted from accepted suggestions in the final translation provided by the users. The low performance of TT_3 is reflected in the low percentage of text inserted from suggestions.

3.7.4.2. Qualitative evaluation

After TT_2 and TT_3, the users answered the following questionnaire:

- Answering with a Likert scale from 1 ("Strongly disagree") to 4 ("Strongly agree"):
  - Q_A: Suggestions helped me to translate faster
  - Q_B: Suggestions helped me to translate with less effort
  - Q_C: Suggestions helped me to improve translation quality
  - Q_D: Suggestions made the translation task more enjoyable
- Answering with yes/no
  - Q_E: I prefer to get shorter suggestions
  - Q_F: I prefer to get longer suggestions [53]
  - Q_G: I prefer to get suggestions more often
  - Q_H: I prefer to get suggestions less often
  - Q_I: I prefer to get more suggestions at the same time
  - Q_J: I prefer to get less suggestions at the same time

The results of the questionnaires can be seen in Table 3.10. The user who obtained the most improvement in TT_2, Z_3, was the one that praised the tool the most. Overall, users think

[53] Even when at first glance Q_E and Q_F may seem contradictory, they are not; users may want to get suggestions with more disparate lengths (answering yes to both), or they may want to keep the length of the current suggestions (answering no to both). Users were told this before answering the first questionnaire.

TT_2	Q_A	Q_B	Q_C	Q_D	Q_E	Q_F	Q_G	Q_H	Q_I	Q_J
Z_1	3	3	2	3	2	1	1	2	1	2
Z_2	3	3	2	3	1	2	1	2	2	1
Z_3	4	4	4	4	1	1	1	1	2	1
Z_4	2	3	3	3	1	2	1	2	2	1
Z_5	3	3	2	3	1	2	1	1	1	1
Z_6	3	3	2	3	2	1	1	2	1	2
Z_7	2	3	3	2	2	1	1	2	1	2
Z_8	3	3	2	3	1	2	2	1	2	1
Median	3	3	2	3	1	1.5	1	2	1.5	1

TT_3	Q_A	Q_B	Q_C	Q_D	Q_E	Q_F	Q_G	Q_H	Q_I	Q_J
Z_1	2	2	2	2	1	2	2	1	2	1
Z_2	2	2	2	1	1	2	2	1	2	1
Z_3	1	1	1	1	1	2	2	1	2	1
Z_4	2	2	2	2	1	2	2	1	1	1
Z_5	2	2	2	2	1	2	2	1	2	1
Z_6	1	1	1	1	1	2	2	1	2	1
Z_7	4	4	4	4	1	1	1	1	1	1
Z_8	2	2	2	2	1	2	2	1	2	1
Median	2	2	2	2	1	2	2	1	2	1

Table 3.10: Answers of the users to the different questions. TT_1 had no questionnaire. Z_3 really liked the approach taken in TT_2, while Z_7 really liked the one taken in TT_3; they both showed strong agreement when asked if the respective approach made them translate faster, with less effort, improved the translation quality and made the task more enjoyable. The rest of the users did not have strong opinions, but clearly preferred the strategy without thresholds (TT_2) over the one with (TT_3). Most of them think suggestions helped them to translate with less effort (Q_B) and made the task more enjoyable (Q_D), a majority thinks it helped to translate faster (Q_A), and half thought it helped them to improve the quality of the translation (Q_C). In TT_2, most users preferred to be shown less suggestions (Q_G); in TT_3, most users preferred longer suggestions (Q_E) suggested more often (Q_F) and with more suggestions at the same time (Q_I).

	Fastest	Slowest	Easiest	Hardest
Z_1	TT_1	TT_3	TT_2	TT_3
Z_2	TT_2	TT_1	TT_1	TT_3
Z_3	TT_2	TT_1^*	TT_2	TT_1^*
Z_4	TT_1	TT_2	TT_2	TT_1
Z_5	TT_1	TT_3	TT_1	TT_3
Z_6	TT_2	TT_3	TT_2	TT_3
Z_7	TT_3	TT_2	TT_3	TT_2
Z_8	TT_2	TT_3^*	TT_2	TT_3^*

Table 3.11: Answers to the speed and ease of use questionnaire. As with the preliminary evaluation, users have a hard time correctly assessing the task where they translated faster and slower; only one user managed to correctly chose the task with the highest translation speed (Z_3 and TT_2). User's dislike of TT_3 probably also affected the rankings. (*) Z_3 and Z_8 consider TT_3 useless, and comparable to not getting assistance at all (TT_1).

the strategy in TT_2 helps them to translate with less effort and make the translation process more enjoyable, but they have different perceptions about the quality of the final translation: half of them think the tool helped them to improve the quality and half of them thinks it did not.

Users disliked the strategy used in TT_3. All of them either disliked or strongly disliked the approach, except for Z_7 that strongly liked it, what is unexpected, given that Z_7 performance in TT_3 was worse than on TT_2.

Finally, after the second questionnaire, they were given a different questionnaire with three questions:

- Sort the systems according to your perceived translation speed.
- Sort the systems according to your perceived translation ease.
- Comments and suggestions.

Users answers to the first two questions can be seen in Table 3.11.

Most users complained about TT_3 in the comments section, and thought the suggestions got offered "too late": the thresholded model did seldom offer suggestions for the first or second letter of each word, but, instead, offered suggestions with multiple words after most of the first word was typed. Another common kind of suggestion were function words. The fact users disliked TT_3 can be explained by two facts: the first one, that they just finished TT_2, that offered suggestions more often; the second one, that they did not know exactly the kind of assistance they were going to get in TT_3. If users knew that they would get less (but more promising) suggestions, they may have reacted differently to this task; some users admitted this would have been the case. The only user that liked TT_3, Z_7, complained about all the time lost reading the suggestions in TT_2: as less suggestions were being offered in

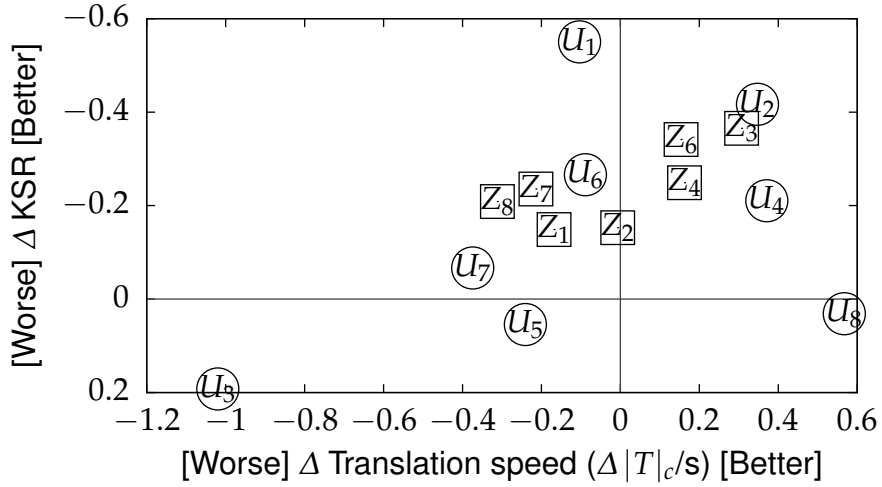


Figure 3.17: Comparison of $\Delta|T|_c$/s and ΔKSR for the professional translators Z_1–Z_8 and the untrained translators U_1–U_8 (from the previous experiment described in Section 3.5). Results are similar for both user groups: 3 untrained translators and 4 trained ones translate faster with assistance, and most manage to reduce the KSR.

TT_3, the user had less to read. The rest of users did not have this complain; they got used to the suggestions in TT_2, and disliked the lack of them in TT_3.

Some users asked to "freeze" the suggestions, that is, to keep showing the suggestions offered at the start of the word even after they have typed some more letters of the prefix, even when the offered suggestions would not be compatible with the typed prefix; asked about this, the main problem they had was with suggestions being shown changing too often (once per keypress). User Z_4 complained about concordance problems: as the black-box approach splits the original segment in subsegments up to a given length, sometimes the context needed to keep the concordance is lost (e.g. translating form English, a word should be feminine, but no word in the subsegment has a gender source). Z_1, Z_2 and Z_4 think a balance between TT_2 and TT_3 would be a better approach.

3.7.5. Discussion of the human evaluation

Both the evaluation round with professional translators and the one with untrained translators (described previously in Section 3.5) prove that the black-box ITP approach saves effort to the translators, and, sometimes, makes them work faster. Figure 3.17 compares the performance of both user groups: both test rounds obtained similar results, with most of them saving keystrokes (up to 55% less keystrokes) and some of them time (up to 20% faster translation). They all had similar complaints about the tool offering too many suggestions, and commented the usefulness of the suggestions when they got translation alternatives that improved their planned translation.

The first round of testing proved that, under similar testing conditions, the black-box approach beats Thot, a state-of-the-art ITP glass-box approach, when comparing the translation speed. This proves that not only the black-box approach is viable for saving time and keypresses, but also an appropriate choice for some scenarios where it is possible to train the underlying model needed for glass-box ITP.[54]

The second round of testing proved that the threshold strategy is not adequate for reducing the number of suggestions shown. Even when the automatic experiments show that this approach is viable, and that it could be used to finely tune the trade-off between KSR and ASR, the approach was not well received by the users. The approach needs to be analyzed and reworked in order to be useful.

[54] In the other scenario, where, for any reason, it is not possible to train the underlying model (reasons such as lack of bilingual information for the specific domain), glass-box ITP is not an alternative, and using black-box ITP is the only option.

Chapter 4

Concluding remarks and future work

4.1. Concluding remarks

Interactive translation prediction (ITP) is a computer-assisted translation modality where a computer assists a human translator by offering one or more possible continuations for the translation that is being carried out, with the objective of saving effort and time. State-of-the-art ITP systems follow a glass-box approach to generate the context-aware suggestions: they use the inner workings of an underlying statistical machine translation (SMT) engine. This means the ITP approach has access to most (if not all) the information contained in the SMT model, but there are several scenarios were training such system may not be possible, particularly when there is not enough bilingual corpora to train a good enough model.

In this dissertation, an approach that aims to solve this problem has been presented. The proposed black-box approach does not need to use a particular underlying SMT model to generate the suggestions, unlike the glass-box approach; instead, it can use any bilingual resource, such as machine translation systems (including SMT), dictionaries, glossaries, or phrase tables. The resources are used to translate all the possible subsegments of the source sentence up to a given length; therefore, any resource capable of providing with one or more translations per segment can be used. Initially, a heuristic approach to rank and select which suggestions should be shown to the user was used; then, a more principled approach based on neural networks was used to significantly improve the performance of the ITP system. The automatic evaluation performed shows that between 20% and 50% keystrokes can potentially be saved, compared to unassisted translation, regardless of whether the task consists in translating between related languages such as Spanish–Catalan and Spanish–English, or more unrelated languages such as Arabic–English and Simplified Chinese–English.

Three different rounds of human evaluation have been performed:

- The first round tested the heuristic approach with untrained translators. [Chapter 2.5]

- The second round tested the neural network approach with untrained translators, and compared the performance of the tool with a state-of-the-art glass-box ITP system in similar conditions. [Chapter 3.5]

- The third round tested the neural network approach with professional translators. This round also tested an approach where suggestions under a certain threshold did not get offered. [Chapter 3.7]

Users saved around 15% keystrokes in all three evaluations, but could have saved more (up to 70%) if they had used all the compatible suggestions. Most users think the tool helped them to translate, and that using the assistance is a pleasant experience, regardless of their particular profiles.

4.2. Future research lines

In this section, multiple future research lines will be described. Some of the proposals have been explored with preliminary experiments to illustrate their viability.

4.2.1. Inclusion of new bilingual resources

The performance of the black-box ITP approach has been tested using rule-based MT,[1] and phrase-based statistical MT.[2] But different resources may also be used to provide the translation needed to generate the suggestions, such as:

- Dictionaries and glossaries. This kind of resources may require lemmatization or stemming, as they may not be able to handle inflected forms.

- Phrase tables extracted from phrase-based statistical machine translation systems.

- Translation memories aligned at subsegment level.

- Neural MT.[3]

Testing new resources will not only prove their usefulness when used in black-box ITP tasks, but also enable the usage of the tool in new translation scenarios.

[1] Apertium(Forcada et al., 2011)

[2] Moses(Koehn et al., 2007) and Thot(Ortiz-Martínez, 2011)

[3] If it is possible to modify it, the NMT engine can provide neural glass-box ITP with a simple modification in the decoder process. But the cost of using neural ITP is high, as the system has to generate new suggestions for every keystroke; using NMT to translate the subsegments for black-box ITP may be a much cheaper alternative, and the translations can be performed beforehand and cached in case the NMT system is slow.

4.2.2. Combination of bilingual resources

Related to the previous task, using more than one resource is possible, but has not been tested. Two approaches are proposed when combining bilingual resources:

- Translating all the subsegments with all the available bilingual resources. The suggestions would then be chosen by the neural network, e.g., by adding a feature that represents the bilingual resource (or kind of resource) that produced the translation so the neural network can identify the kind of suggestions that are more successful from each resource.

- Choosing which bilingual resource will be able to provide a more useful translation for each subsegment, a technique named translation brokering (Sánchez-Martínez, 2011). Unlike the previous approach, this technique does not increase the number of words translated. It can be specially interesting if the criteria can be modified to also include the price of the translation, so only the more complex translations are performed using expensive resources, using simpler, cheaper ones for the easier translations.

The inclusion of multiple resources at the same time may improve the performance of the tool.

4.2.3. Improving the segmentation

Currently, the source sentence is segmented in all the possible subsegments up to a given length. The number of segments translated may be reduced without decreasing the performance by using more principled ways of segmenting; for example, the segmentation may pay attention to syntactical information, ignoring segments that contain part of a subordinate clause.

Additionally, according to empirical research (Carl et al., 2011), translators segment the translation task in smaller units; it was found that a translator first reads the segment and gets a general plan, and then alternatively focuses on part of the segment (named fixation unit) and types its translation (named production unit). Experienced translators tend to translate longer units than translation students, and are also able to better overlap the time spent producing an unit with the fixation on the next one; both these reasons are regarded as the main contributions to translation speed. The accuracy of the suggestions may be improved by identifying the user's fixation units, and segmenting according to them.

Another strategy that can be used is to translate the full segment, then subsegment the translation. This approach would only be useful for the systems developed in this dissertation if the bilingual resource used to translate the segment is capable of providing alignment information, as both the heuristic and the neural network need information related to the origin position of each suggestion.

4.2.4. Learning to rank

The current ranking method score the suggestions as independent events.[4] But, suggestions may be interdependent on some level: e.g., they may partially cover each other, use a synonym for a term or different morphological information. Additionally, treating each suggestion as an independent event can lead to scenarios where similar suggestions have high scores; this maximizes the expectancy of each one of them being useful, but may not leave space for a suggestion that scores worse but fits better with the user translation.

Therefore, a method that learns to rank a list of suggestions, rather than individually score them, may improve the performance of the system; not only the system would have more information to score each suggestion, but also it may lead to a better strategy where diversity among the offered suggestions is enforced by the ranking algorithm. To this end, the strategy proposed by (Hang, 2011) can be adapted.

4.2.5. Trusted suggestions

A technique for reducing the number of suggestions offered to the users was explored in Chapter 3.6, but the strategy was not well received by the professional translators. Therefore, a different approach for reducing the number of suggestions has to be designed. Taking inspiration on how translation memories work, a pool of trusted suggestions is created.

The trusted suggestions pool gets populated by adding those suggestions that were used or could have been used to advance in the translation; that is, those that would be considered viable if the automatic evaluation system described in Chapter 2.2.1 was run with the user translation as a reference are also added. Otherwise, the suggestions pool would remain empty, as, if it is empty, no suggestion is offered, hence no suggestion can be accepted and added to the pool.

- The trusted suggestions are more likely to be useful in the following translations, operating under the same assumption translation memories use: that similar segments will appear in the context.

- The trusted suggestions match segments typed by the user; therefore, the suggestions are adapted to the writing style of the translator. This can be an issue if the style of the user is too different from the one generated by the bilingual resources, as suggestions will be seldom offered.

This method needs an additional training phase, as, at the start, the trusted suggestions pool will be empty, and no suggestion will be offered. As the user translates more and more segments, the pool will increase in size. If there is an already translated corpus available, it is advisable to pretrain the model and start translating with a populated trusted suggestions pool.

[4] Some features used in the neural network model are affected by the rest of suggestions, but each suggestion is treated as an independent event.

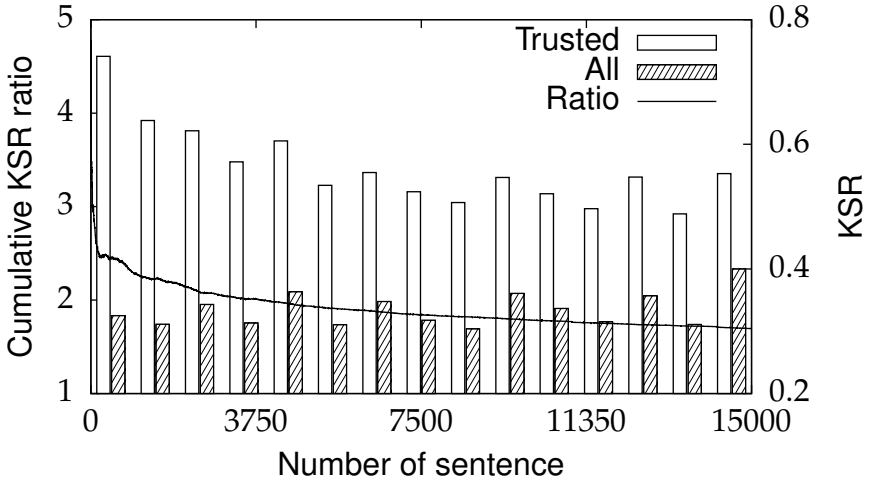


Figure 4.1: Evolution of the KSR (in blocks of 1 000 sentences) and the ratio between the accumulated keystrokes using only the trusted suggestions and the accumulated keystrokes using all the suggestions (one data point per sentence) as the trusted suggestion pool increases for the Catalan to Spanish evaluation.

Using the same conditions as in Chapter 2.4, and starting with an empty trusted suggestion pool, an automatic evaluation was performed offering only suggestions that are already in the pool. After each sentence was typed, the suggestions in the viable suggestion set were added to the pool; for measuring the performance, the cumulative KSR will be used, that is, the total number of keys pressed from the first sentence to the current one over the total length in characters of the translated sentences. As the cumulative KSR is an average for all the previous sentences, the KSR for groups of 1 000 sentences is also used; these values show how the distance between the KSR obtained using the trusted suggestions and the KSR obtained using all the suggestions evolve. The results of the test can be seen in Figures 4.1 to 4.4.

As expected, with an empty trusted suggestion pool, the KSR of the trusted method is worse than not using the strategy; but, as the pool gets populated, the values get closer. Obviously, the first few sentences create a steep decline of the ratio, as many common segments, such as function words and nouns that appear frequently in the domain, get added to the pool. The value does not stop improving as more sentences get translated and more segments get added to the trusted suggestion pool, but the grouped KSR shows that the KSR of the trusted strategy for the final sentences (where it is mostly populated) is around 1.5 times the KSR of using all the suggestions. The ASR for the trusted strategy is higher than the one using all the suggestions, specially for $M = 1$. Therefore, using the trusted strategy leads to using more keystrokes (around 50% more), but the amount of suggestion pop-ups where no suggestion is viable gets greatly reduced, specially for low values of M.

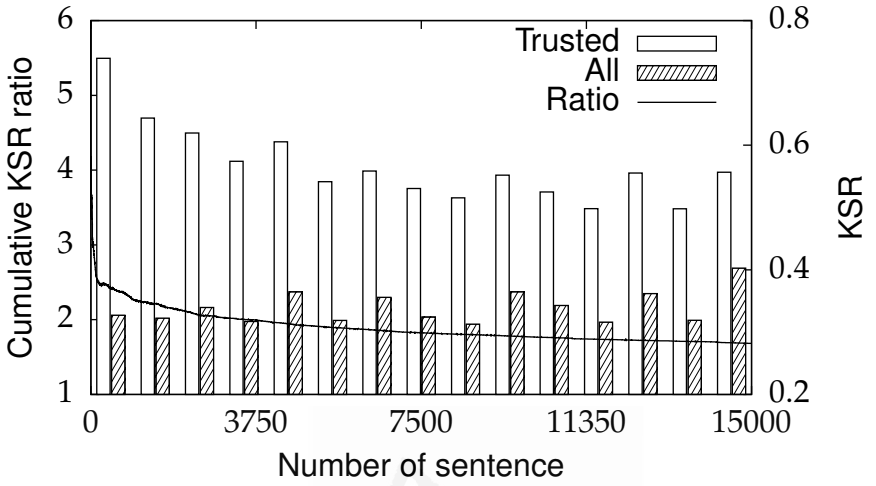


Figure 4.2: Evolution of the KSR (in blocks of 1 000 sentences) and the ratio between the accumulated keystrokes using only the trusted suggestions and the accumulated keystrokes using all the suggestions (one data point per sentence) as the trusted suggestion pool increases for the Spanish to Catalan evaluation.

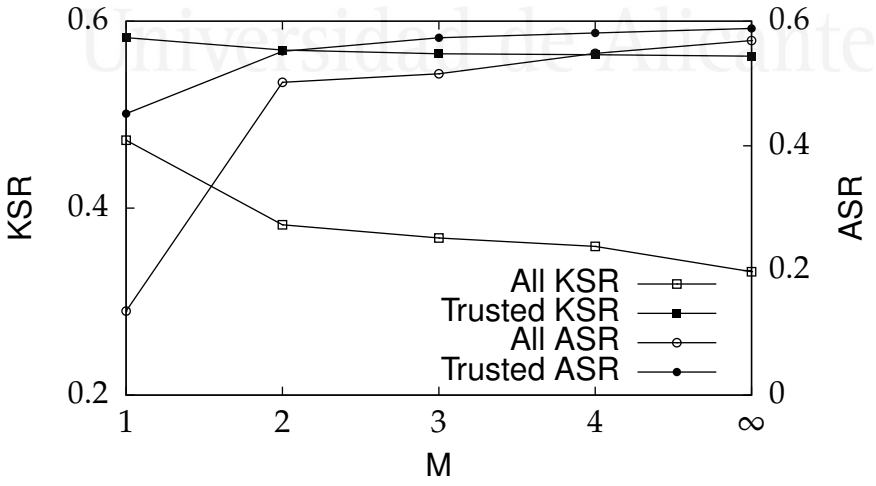


Figure 4.3: KSR and ASR for the different values of M for the Catalan to Spanish evaluation.

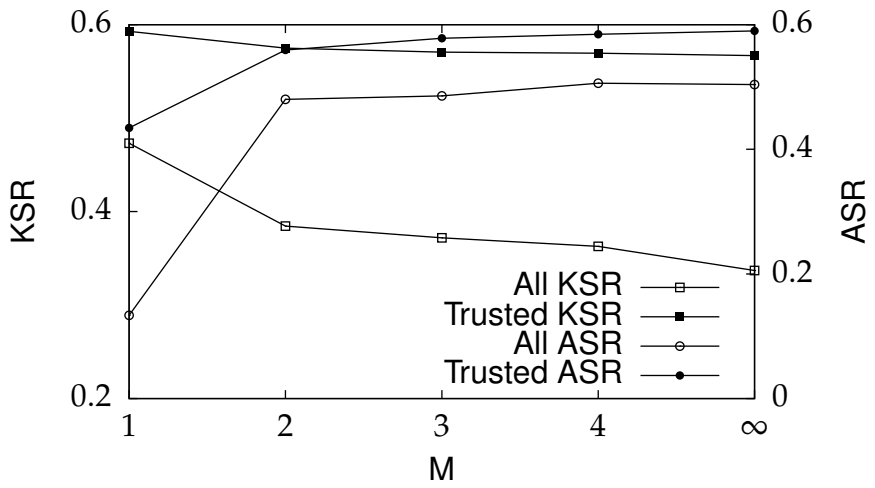


Figure 4.4: KSR and ASR for the different values of M for the Spanish to Catalan evaluation.

4.2.6. Merging suggestions

A different way of reducing the number of suggestions is to remove redundancy. One source of redundancy comes from offered suggestions that are the prefix of another offered suggestion. Initially (in the automatic evaluation procedure described in Chapter 2.2.1), only full suggestions could be used; if a suggestion prefix was useful, the user would have to select the full suggestion, then delete the incorrect suffix, what could lead to a higher keystroke than just typing the prefix.[5]

Later , the possibility of selecting a suggestion prefix was added (in the automatic evaluation procedure described in Chapter 3.2.1). Rather than selecting the whole suggestion, the user can select a prefix of any suggestion, word by word, with a cost of one keystroke per word in the prefix. This cost is still higher than selecting a whole suggestion, but lower than deleting the suffix in most cases.

For example, using $M = 2$, if the suggestions “the orange sports car is” and “the orange sports” are offered, the user can choose the full suggestion “the orange sports” with a cost of one keystroke, or select the first three words of “the orange sports car is” with a cost of four keystrokes; either way, there are keystrokes savings (16 for the first action, 13 for the second); both suggestions can be merged in order to free up a space in the suggestion list that can be used by another suggestion.

Two different ways of merging suggestions have been defined, attending to how probably the merging will create the previously explained problem:

[5]Even thought, the temporal cost of deleting the suffix may be lower than the temporal cost of typing the suggestion; lacking empirical research, it is assumed that the cost is similar for every keystroke.

Rank	Original suggestions	Soft compact	Hard Compact
1	the	the	the red car
2	the red car	the red car	
3	the red		

Table 4.1: Example of the compacting heuristics. The suggestions are sorted in the same way they are offered (by descending score). In the soft column, "the red car" is not merged with "the" because "the" has a higher score. The hard heuristic ignores the scores and merges them all.

- A soft way that only merges suggestions if the longer one has a better rank than the shorter one; if the shorter one has a larger probability, it is kept on the list, as accepting the full suggestion has a lower cost, otherwise, the shortest one gets removed.

- A hard way that merges suggestions regardless of the length; if the shortest suggestion is higher in the suggestions list, it gets replaced by the longer one, otherwise, the shortest one gets removed. This strategy is riskier than the another, and is more prone to creating scenarios where a suggestion that could be selected gets replaced by a suggestion with only a compatible prefix, forcing the user to use more keystrokes selecting the prefix.

An example can be seen in Table 4.1.

The automatic evaluation of both approaches using the same setup as in Chapter 3.3 was performed. Results are shown in Figure 4.5. In most cases, the soft heuristic improves the KSR slightly, albeit not in a statistically significant way. The hard heuristic improves the KSR for low values of M, while it worsens it for bigger values; when a shorter compatible suggestion gets replaced by a longer incompatible one, a few more keystrokes have to be used to select the compatible prefix.[6] The ASR values are worse for the strict merging than for the other cases. The average length of the selected prefix is 1.23 words for the non-compacted suggestions, 1.24 words for the soft compacted and 1.9 words for the hard compacted ones.

4.2.7. Managing used and rejected suggestions

The number of suggestions can also be reduced by exploiting the information the user provides when interacting with the suggestions: by considering if the user already accepted a suggestion, or rejected it multiple times, certain suggestions can be hidden, or even removed from the suggestion pool, reducing the cognitive load. Still, it is worth noting that, during the different rounds of human testing, users did not specifically complain about accepted or rejected suggestions being offered again, but they did complain about suggestions being offered too often.

There are different ways in which user actions can be taken into account:

[6] Accepting a suggestion has a cost of 1, but accepting a prefix has a cost of 1 plus 1 for each selected word.

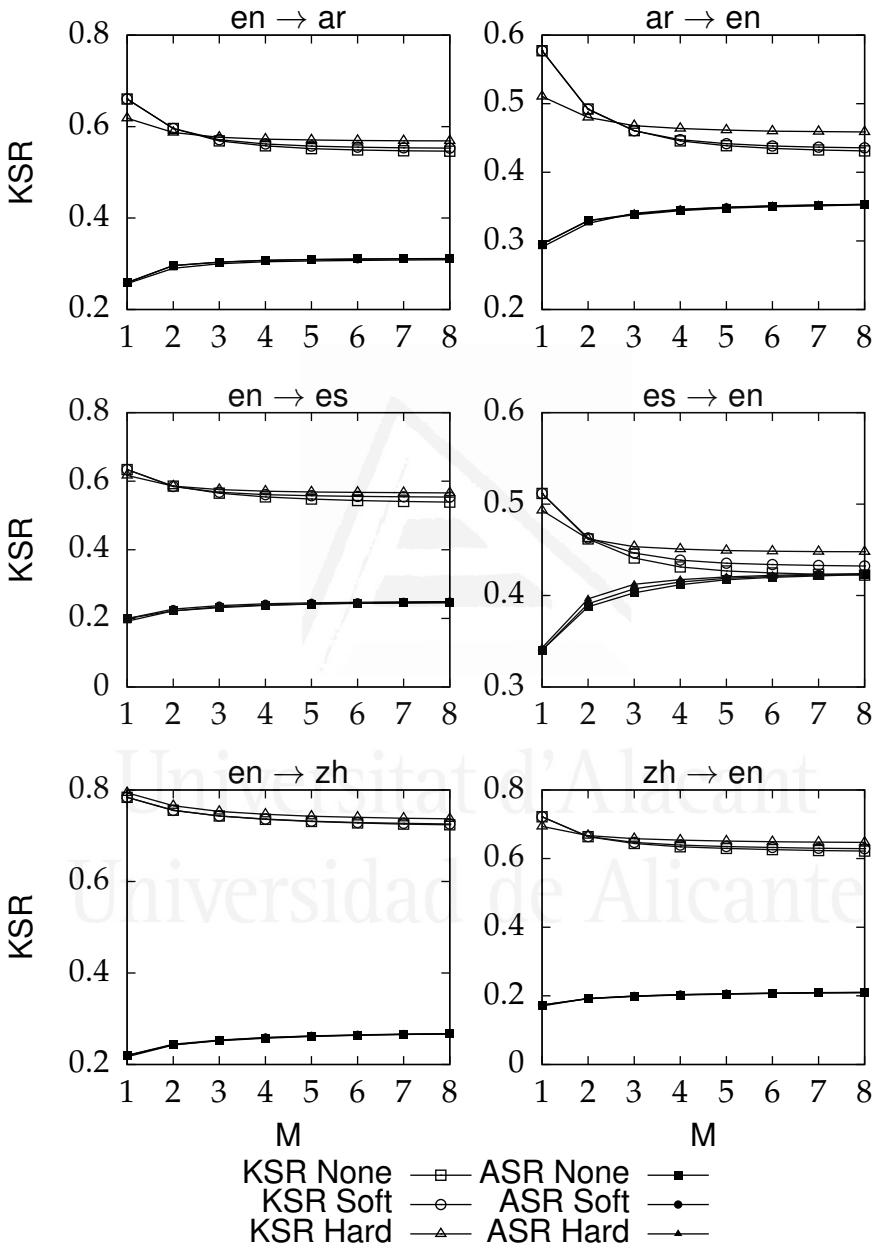


Figure 4.5: KSR and ASR of the different compacting strategies (no compacting, soft compacting, hard compacting).

- Used suggestions from the suggestion set can be removed; that is, if a suggestion is the translation of only one source subsegment, it is no longer offered. There is one exception: if there are two suggestions with the same t_p (but different b_p or e_p), they cannot be told apart, as there is a lack of explicit alignments between the source sentence and the translation; therefore, when such suggestions get used, they cannot be removed from the suggestion pool. This approach is considered safe, as it will never remove viable suggestions from the suggestion set.

- Additionally, suggestions that overlap with a previously used suggestion can be removed; e.g. if the suggestion "the red car" is used, the suggestions"the red", "red car", or even "red car is mine" may be removed. Three different approaches have been devised following this idea:
  - Removing the suggestions that exactly match with a previously used one. This approach would be similar to the previously discussed one, but would ignore the exception.
  - Removing the suggestions that are contained in (or exactly match with) a previously used suggestion.
  - Removing the suggestions that overlap in any way with a previously used suggestion.

- Finally, if a suggestion gets ignored often, it may also get demerited, so it stops showing.

The first and second points are already taken into account by by the feature f_{30} of the neural network (see Chapter 3.1.1); those suggestions whose source subsegments overlap with the source subsegment of a used suggestion get demerited; still, if there are no better suggestions, demerited suggestions may still show. Removing them from the suggestions set will completely avoid this.

Under the same conditions as in Chapter 2.4, the approaches were automatically tested. While most methods of removing suggestions improve the KSR, the improvement is in the order of 10^{-3}, and none is statistically better or worse than not removing the used suggestions at all. Similarly, the methods improve the ASR, but, with small gains and results that are not statistically better or worse than not removing the used suggestions.

4.2.8. Highlighting the already translated part of the source segment

While not directly related to the ITP task, some of the state-of-the-art glass-box ITP systems offer this function. If a suggestion that unequivocally covers a part of the source segment[7] is used, the covered words can be highlighted; more advanced methods like the light alignment model described in Chapter 3.3 can be used to provide an stochastic alignment for the words that cannot be unequivocally aligned.

[7] If more than one suggestion has the same t_p, it is not possible to unequivocally tell the origin of the suggestion.

4.2.9. Better modeling of user actions

The current automatic evaluation emulates a reasonable, yet fairly unrealistic translator, and fails to take into account the time spent reading the suggestions. The KSR measures the amount of saved keystrokes, and the ASR measures how many times the user is presented with a list of suggestions where none of them are useful. Also, the mere act of reading a suggestion is not enough to use or discard it; the user has to think if the suggestion:

- Matches the planned translation.
- Improves the planned translation, e.g. by offering more precise terminology the translator did not think about.
- It is partially compatible with the planned translation and has to be edited, e.g. the suggestion has a verb with an incorrect tense. The user has to choose between accepting the suggestion then editing it or ignoring the suggestion.

Also, the user may ignore the suggestions altogether for different reasons, such as when typing a sequence of characters without stopping to read the suggestions between each keypress, or when the user gets too many useless suggestions and starts ignoring them.

Devising a better user model can lead to more accurate automatic evaluation that can be used to further improve the system.

A method using fuzzy alignments and the effort needed to accept a suggestion has been proposed (Foster et al., 2002). The model assigns a probability of acceptance and a potential benefit to each suggestion, and lets the user not only type and accept suggestions, but also remove an arbitrary suffix of an accepted suggestion.[8] The benefit depends on the length of the suggestion, the length of the removed suffix and the cost of reading the suggestion; the reading cost is also taken into account for suggestions that are shown but not selected. Using a greedy approach, they look for the best sequence of words according to the alignment model, then offer only those suggestions that are expected to have a positive benefit. Human evaluation showed that up to 10% keystrokes may be saved by using this method, compared to unassisted translation.

Software engineering techniques such as GOMS (Goals, Operators, Methods and Selection rules) (Card et al., 2000) can also be used to better describe how the users interact with the tool and get better measurements; GOMS is a methodology that describes how a human interacts with a computer. In this case,

- The *goal* is to generate a translation
- The *operators* are typing keys on the keyboard, using the mouse for moving the cursor, selecting text, deleting, accepting suggestions, etc. and the mental operator, that represents the time the user is thinking or assessing the situation

[8] This approach is similar to using tab to select the prefix word by word.

- The *methods* are the different ways a part of the translation can be inserted: by typing it completely, by accepting and editing one or more suggestions, etc.
- The *selection rules* are the ones the translator applies for taking the best sequence of operators for completing the goal.

There is a specialization of GOMS called GOMS-KLM (keystroke level modeling) where all the operators are keystrokes and mouse actions, including virtual keyboards and other forms of input, taking into account the time needed for shifting from one method to another (like grabbing the mouse or going back to the keyboard). GOMS-KLM could help us to assign a standard temporal cost to each operation and method involved in the translation process; thus, in constitutes a better approximation to the real cost of the task.

Finally, the automatic evaluation system can be adapted to better mimic how a human translator planned translation changes when suggestions are offered: the current approach uses a fixed, immutable reference, but a human translator may be more flexible and change the planned translation in different ways, such as reordering the sentence, or using a different lexical selection. Improving the evaluation system so it is more flexible will make the evaluation more human-like, but multiple references for the each sentence may not be available, and creating them may be too expensive.

Appendix A

Free/open-source software released as part of this PhD thesis

All the software developed along with this thesis has been released as free/open source software under the GNU General Public License version 3.[1] The software was developed with the objective of evaluating the different algorithms and approaches described in the thesis; as the software is publicly available, the methods results obtained can be reproduced, and the methods can be improved. Additionally, any person can freely use all of them for any purpose they desire.

A.1. Forecat

`Forecat`[2] is both a command line Java application to test the performance of the different approaches developed in this work and a web server that lets users interact with the ITP tool. `Google Web Toolkit`[3] was used to implement the web server part.

The command line application implements all the different approaches developed in this dissertation, and it was used to conduct all the automatic experiments described in this work. The web server offers three different interfaces:

- A webpage that runs all of the black-box ITP processes on the server; this approach is suited for users that run Forecat locally, or for setups where the server running Forecat is significantly more powerful than the clients. The web page acts as a thin client in this mode: the server is queried for new suggestions with every keystroke.
- A webpage that runs most of the black-box ITP processes on the client; this approach is suited for deploying Forecat on a big scale, or for environments with latency problems.

[1] `https://www.gnu.org/licenses/gpl-3.0.en.html`
[2] `https://github.com/transducens/forecat`
[3] `http://www.gwtproject.org/`

The server only performs one part of the task, the translation of the subsegments; the rest of processes are run client-side. As the suggestions are being ranked and offered locally, suggestions are offered almost instantly.

- Web services, with the same functionality as the webpage that runs all the processes server-side.

The web interface was used to carry out the experiments in Section 2.5.

The C/C++ `FANN` library[4] is used for loading querying the neural network model, via `JNA`[5], using code from `FANNJ`.[6] `FANN` was also used to train the neural network models.

A.2. Forecat-OmegaT

`Forecat-OmegaT`[7] is a plugin for the free/open source TM tool `OmegaT`[8] that adds the black-box ITP assistance of `Forecat` into `OmegaT`. Any MT engine available (either native or via plugins) in `OmegaT` can be used to translate the subsegments. The plugin can be configured to use the simple heuristic described in 2, or to load a neural network model and use the machine-learning based approach described in 3. `Forecat-OmegaT` was used to conduct all the human experiments described on Chapter 3.5 and Chapter 3.7. Like in `Forecat`, `FANN` is used to interact with the neural network model.

A.3. Apertium-cli-OmegaT

`Apertium-cli-OmegaT`[9] is a plugin for the free/open source TM tool `OmegaT` that makes a local installation of `Apertium` available to be used from OmegaT using the Apertium command line interface and the Java Process class. It was used as an aid to implement `Forecat-OmegaT`.

A.4. Cachetrans-OmegaT

`Cachetrans-OmegaT`[10] is a plugin for the free/open source TM tool `OmegaT` that pulls translations from a cache file. It acts similarly to a translation memory, but only returns

[4] `http://leenissen.dk/fann/wp/`
[5] `https://github.com/java-native-access/jna`
[6] `https://github.com/krenfro/fannj`
[7] `https://github.com/transducens/forecat-omegat`
[8] `http://www.omegat.org/`
[9] `https://github.com/transducens/apertium-cli-omegat`
[10] `https://github.com/transducens/cachetrans-omegat`

perfect matches. When a translation task is received, all the subsegments up to the desired maximum subsegment length L (see Section 2.1) can be translated and stored in a file; then, when the user performs the translations, the ITP assistance will be available instantly, as there will be almost no translation delay. `Cachetrans-OmegaT` was used to conduct the human experiments on Section 3.7.

A.5. Thot-OmegaT

`Thot-OmegaT`[11] is a plugin for the free/open source TM tool `OmegaT` that integrates the ITP and SMT toolkit `Thot`,[12] both as a MT system and as ITP assistance in `OmegaT`, using the `Thot` server. It was used to conduct the human experiment that compared Forecat and Thot described on Section 3.5.

[11] `https://github.com/transducens/thot-omegat`
[12] `https://daormar.github.io/thot/`

Appendix B

ITP human evaluation history

Automatic evaluation can only give an estimate of the performance of a system. For obtaining real figures, human evaluation has to be carried out. In this section, the methodology and results of multiple ITP human evaluations in the state-of-the-art will be described. This analysis was used to better design the human evaluation carried out in this dissertation, specially the one performed in Chapter 3.7.

B.1. TransType and TransType2

B.1.1. First TransType trial

The first published human trial for interactive translation prediction (ITP) was carried out by the TransType project (Langlais et al., 2000). The test evaluated the first TransType prototype; 10 volunteers (4 of them translation instructors or professional translators and the other 6 graduate students) tested the tool for around 1 hour, translating from English to French, one user at a time, over the span of three weeks. Up to 7 single-word compatible suggestions were offered to the user, even with an empty prefix; the user could then accept one of the suggestions using the mouse or a hotkey, or ignore the suggestions altogether and keep typing. Around one hundred sentences from the Hansard corpus (an edited verbatim report of the proceedings of the House of Commons of Canada) was used in the test.

The test was divided into 4 different stages:

- (15 min) Introduction to TransType, with an in-depth explanation of how TransType works and specific translation instructions, emphasizing that the users were not to worry about formatting, instead to «focus on producing a version which [sic] substance would require a normal review» (Langlais et al. (2000), p. 4). The user could then ask any questions they had about the test.

- (5 min) The user worked with TransType in silent mode (meaning no suggestions were offered) to familiarize themselves with the capabilities of the editor.
- (20 min) The user worked with TransType suggestions, and the actions were logged. This stage was further subdivided into two parts:
  - During the first part, the user translated while TransType offered the suggestions, but no further assistance was available.
  - During the second part, the TransType staff gave tips when appropriate; specifically, they suggested to use «the following strategy: accepting long enough completions using the mouse» (Langlais et al. (2000), p. 6), even when «some subjects were really disturbed when asked to use the mouse instead of the keyboard to accept the completions.» (Langlais et al. (2000), p. 6).
- (5 min) "Briskels" (hard-coded multi-word suggestions) were offered along with the TransType suggestions in a separate region of the interface.

The tests showed users lost 35% productivity which respect to unassisted translation, excluding pauses longer than 30 seconds,

> These long pauses generally occurs [sic] when subjects asked questions during the test. In practice, we removed 34 pauses for a total duration of around 27 minutes over a total duration of approximately 6 hours (Langlais et al. (2000), p. 5)

and saved 30% keystrokes compared to unassisted translation; users could have saved up to 75% keystrokes if they had used all the viable[1] suggestions. The faster translation speed of the unassisted task was attributed to users not having to read the suggestions when translating.

> [...] the typing speed reduction [...] may lie in the fact that a user has to perform a task that he/she does not have to do in the first stage, that is, reading the completions! This may disturb the work of a translator more than we expected. (Langlais et al. (2000), p. 6)

Some behaviour trends were found: the assistance made the translators move back and forth less, as the suggestions helped them to make less mistakes; the average used suggestion length was of 5 characters, and 90% of suggestions were accepted before typing any character or with just 1 character typed; most times, the users did not even perceive the suggestions pop-up, as they were focused on typing a sequence of words; whenever they stopped to read the suggestions, a stop (on average, less than 2 seconds) appeared in the log.

Finally, the users were asked to provide feedback about the tool. All users thought the interface was friendly enough, but one user loathed the suggestion system. About the

[1] A viable suggestion is one that lets us advance in the translation being typed.

intrusiveness of the pop-up, 3 users claimed that the pop-up was not an issue, and 6 claimed it was intrusive, but «according to some users, it happens often that TRANSTYPE has a positive impact on the quality of the translation, notably by proposing a word that they were not thinking of, or by encouraging the translator to validate when appropriate full words instead of abbreviations they would otherwise use.» (Langlais et al. (2000), p. 6) they also asked for the option to disable the suggestions when reformulating part of the sentence. When asked if they felt they were faster using TransType, 5 users said that they were faster, 2 were unsure, and 2 said they were slower; in the actual performance, only one user managed to translate faster.[2] When asked for general feedback, users complained about TransType missing some inflections and about some calques being offered (it was found later that the problem was originated in the MT engine). Users widely praised the "briskels" feature, but asked for the suggestions to be shown along the rest in the pop-up;[3] multiword suggestions became part of the second TransType prototype.

B.1.2. Second TransType trial

A second human evaluation with 9 translators[4] using the improved TransType prototype was carried out (Langlais et al., 2002). The prototype had two main improvements:

- Support for lexicons was added. A general-purpose lexicon with multiword entries was used to emulate an SMT phrase table; a task of this evaluation also used medical domain specific texts and provided a specialized lexicon.
- Improvements on the interface, replacing the functionality of the enter key[5] and adding new hotkeys to ease the translation process.

The tests used 100 isolated sentences from the Hansard corpus, using sentences that were not too short, ambiguous or hard to translate:

> We excluded the sentences that had been used during the training of the language and the translation models and also removed sentences that were too long, contained too many complicated proper names or numbers, etc. Finally, we inspected the selected sentences in order to remove those that we found to be ambiguous or difficult to translate without larger context (e.g. sentences with ellipses, etc). (Langlais et al. (2002), p. 3)

An excerpt from the Health Canada Internet webpage was used for the task with the specialized lexicon. The users did not had access to dictionaries or any other kind of translation resource.

[2] A common trend that presents in most trials.

[3] In this evaluation task, the briskels appeared on a different part of the interface.

[4] The concrete profile of the users is not described in the paper.

[5] Formerly, the enter key was used to change to the next segments, but users often tried to use the enter key to accept a suggestion.

> No other resource than TRANSTYPE was available to the user at the time of translating. In particular, and despite the fact that some users were disturbed by this, we did not provide them with a dictionary. Consultation of a dictionary would have made the timing of our experiments difficult to automatize. (Langlais et al. (2002), p. 3)

Guidelines similar to the ones used in the previous trial were given to the users. The test had 3 stages:

- (5–8 min) The user worked with TransType in silent mode (meaning no suggestions were offered) to familiarize with the capabilities of the editor.
- (12–20 min + 5 min) The user worked with TransType suggestions, including those in the lexicon, and the actions were logged; 5 additional minutes were given to the user prior to the start of the trial so they could get used to work with the suggestions.
- (5–8 min + 5 min) The user had to translate the medical domain excerpts, using both the suggestions and the specialized lexicon. Again, 5 additional minutes were given to the user prior to the start of the trial to get used to the interface and reduce user's hesitations.

Users' productivity (in words per hour) was lowered by a 17%, but they typed 31% less keystrokes compared to unassisted translation; up to 68% could have been saved if they had used all the viable suggestions. Only 23% of suggestions were selected using the mouse; when using the keyboard to choose the suggestion, 39% of times the first suggestion was the selected one. The user that was the slowest when translating without assistance was the one who improved the most when using the suggestions. While the expected output of the third stage was a better performance than on the second one (as the lexicon was finely tuned), the users were actually slower; translating those sentences was a harder task for them because of the different domain[6] and the unfamiliarity with the glossary:

> The different domain of discourse surely had an important effect on this result. We also observed that translators often started the translation of a unit differently from the terms we had inserted in the lexicon and thus TRANSTYPE could not propose an appropriate completion from its lexicon. We suspect this would not have been observed if the user himself had compiled the lexicon. (Langlais et al. (2002), p. 3)

As in the first trial, only 10% of the suggestions were chosen after typing 2 or more characters, and a pause (on average, 1.5 s) was present whenever a suggestion was used.

Unlike in the first test, no user was able to be faster with assistance, even though some thought they were so. All of them thought the lexicon was a very useful feature, and the

[6] This task used the medical domain; the translators were not used to this domain. The rest of the tasks used the Hansard corpus.

ability to customize it would justify using TransType on their everyday work, even though they also thought it was affecting negatively the quality of the translation, specially when the suggestions made literal translations easier.

B.1.3. TransType2 trial

The TransType project got a second grant and got renamed as TransType2 (Macklovitch, 2006), including more language pairs (English–German and English–Spanish), and different underlying SMT models. TransType2 had 5 rounds of user trials over 18 months, each round taking around 2 weeks to complete. During the test, a total of 40 000 words extracted from the Xerox corpora were translated by each user, who were also provided with a 750 word glossary. Each round was split into a number of different sessions, each one lasting half a working day, and having 2000–2500 words translated. The first 3 rounds had 2 senior translators in each of the 2 different sites where the tests took place; the next 2 rounds added one more, up to 3 senior translators in each place. The two sites were

- Celer Soluciones,[7] where English to Spanish translation was performed
- Société Gamma,[8] where English to French translation was performed

The first two rounds were used to introduce the users to TransType2.

The third round was the first one that had real working conditions, with one session of dry-run (without any kind of assistance), five of measured unassisted translation, then four more sessions using two different modes,

> [...] one in which TT2 [TransType2] generated shorter, multiple completions and the other in which it generated a single, full-sentence completion. (It turned out that the users expressed a clear preference for the latter configuration, saying that having to read through and evaluate multiple predictions caused them to lose an undue amount of time.) (Macklovitch (2006), p. 3)

Three out of four users managed to improve the performance over the dry-run.

The fourth round, that included 1 more translator per site (as already explained), was carried out during 10 consecutive days. The first session was used to let the users remind how to use the tool, and the second one was a dry-run. The translations produced were considered of deliverable quality, and 5 out of 6 users improved the performance in 7 out of the 8 sessions; however, a high degree of full-sentence overlapping between the training sentences and the sentences translated by the users was found; an average of a 20% productivity gain was achieved when removing the repeated sentences. Some argued

[7] http://celersol.com/
[8] http://www.societe-gamma.com/

> [...] that the dry-run productivity figures that were used as a baseline in ER4 [the fourth round] may have been unfairly low, since the one dry-run session in that round had been scheduled on the first day and the translators' performance seemed to gradually improve over the ten-day trial period (Macklovitch (2006), p. 3)

The fifth round addressed this problem by removing all the overlap between the sentences used to translate and the training set. Such as the one before, it was comprised of 10 sessions, with the first one being a refresher and the second one a dry-run. A second dry-run was added near the end, as the users may be more familiar with the corpora: by then the productivity was higher during the second dry-run than during the first one, and even than during the assisted tasks; 3 users achieved their best productivity on this session; users were 9% faster with assistance, and 45% faster on the second dry run, when compared with the first one (927 words/hour, 1012 words/hour and 1346 words/hour for the first dry-run, the assisted tasks and the second dry-run, respectively). Later, it was found that

> In segmenting the test corpus into 2000-word chunks, we had blithely assumed that all the resulting portions would be of more or less equal difficulty, seeing that all were drawn from a similar set of Xerox manuals. But as it turned out, the text that we inadvertently selected for DR2 [the second dry-run] was much easier to translate than all the others in that round. We later measured the average length of the sentences in this text and discovered they were shorter than those of any of the other test files in ER5 [the fifth round]. (Macklovitch (2006), p. 4)

One user disregarded the instructions and failed to complete the dry-runs; 4 of the other 5 saw some improvement over the first dry-run, but, on average, where slower than the second dry-run.

The tool had an option (accessible via a hotkey) that let the users to add comments spontaneously (opening this option stopped the clock used to measure the translation performance). Two main complaints were found in the comments:

- The first one had to do with the repetition of erroneously translated sentences in the test procedure:

  > When the system's initial prediction on these sentences was not to the translators' liking, they would modify it a first time; and later, when that same sentence re-occurred within the file, they found they had to make the same corrections over again. This evidentiates the need of some translation memory-like functionality, where already performed translations are recalled and used as suggestion over the ITP output. (Macklovitch (2006), p. 4)

- The second one had to do with the repetitive nature of the corpora: even when the sentences did not appear again verbatim, sentences with really similar structures that only differ in one or two words do appear;

> Moreover, there is a good likelihood that TransType will reproduce the problem which the translator initially corrected every time it reoccurs, since the system's underlying language and translation models remain unchanged during a working session. This too, the participants found particularly frustrating. "Why can't the system learn from my corrections?" they asked over and over again. (Macklovitch (2006), p. 5)

Simple translation memory functionality would not be able to address this complaint; on-line, adaptive learning is deemed as the correct solution, albeit a difficult one.

B.2. Caitra

B.2.1. Caitra trial

Caitra (Koehn, 2009a) also had a human test with French–English translation and 10 translators (students or staff of the university of Edinburgh), 5 of them were native English speakers and the other 5 were native French speakers. They were paid a fixed amount for the tests that used 192 sentences extracted from the 2009 EACL workshop, which belonged to the newspaper domain; the sentences were split into 5 blocks of roughly 40 sentences or 1000 words, and contained one to three complete documents (news pieces); each one of these blocks was translated under different conditions:

- Unassisted
- Post-editing
- Options: this mode shows translation options, a table with multiple translations for each word or phrase that are colour coded to represent the probability the translation model assigns to that option
- Prediction: this mode predicts the next word or phrase the user is going to translate and offers it as a suggestion
- Prediction + options: this mode offers the functionality of both the Options and Prediction modes

Additionally, professional translators fluent in both French and English were used to assess the quality of the translations, using a simple metric: the translation is correct, that is, it «[...] represents a fully fluent and meaning-equivalent translation of the source.» (Koehn (2009a), p. 249) or it is not. On average, each sentence-translation pair was qualified 5 times.

Users did not have to complete the tasks in a controlled environment; rather, they could complete them using any computer (as Caitra only requires a web browser) during a 2 week timeframe. The results show that 4 users managed to be faster with any of the three kinds of assistance, and 2 were faster on the unassisted task. The study splits the users in 3 groups:

slow translators, who became faster (and some even improved the quality of their translations) when assisted, fast translators, whose translation speed also improved slightly, and users that rejected the use of suggestions. The study found that the performance steadily improved when translating the first five sentences, but the improvement stalled for the rest of them. Some users managed to translate some sentences by typing only 10% of the sentence and completing the other 90% by using the translation table. On average, users spend 4.4 s/word, 2.7 s/word, 3.7 s/word, 3.2 s/word and 3.3 s/word during the unassisted, post-editing, options, prediction and prediction+options tasks respectively. They also saw an increase of 5% on the quality of the translations when comparing unassisted with the rest of modes.

There were also some interesting findings about translation quality. Only half the translations were deemed as good enough to pass.

> After querying some of the human judges, we were left with the impression that they were overly critical ("this translation sounds funny to me"), and may also be tempted, when given ten translations at a time, to label half of them as correct and the other half as wrong: an implicit ranking of the translations. (Koehn (2009a), p. 250)

Still, the assisted translations were deemed as acceptable more often than the unassisted ones.

B.2.2. Caitra monolingual translators trial

A second test with a different goal[9] was conducted using Caitra: rather than using bilingual translators, they used translators that did not understand the source language (Koehn, 2010a). The task had 10 students from the University of Edinburgh and 6 professional bilingual translators (3 for Chinese–English and 3 for Arabic–English) translating 8 stories with roughly 10 sentences each from Chinese and Arabic to English. The source text was shown to help the users with punctuation and numerals, and the prediction+options assistance mode was used. In this case, the translators that evaluated the quality of the translations were monolingual, and had to assess if the translations were good or not using a provided reference translation. Monolingual translators were often able to produce translations of the same quality as professional bilingual ones: in the Arabic task, 35% of the sentences produced by the monolingual translators were deemed as correct, against 61% for the ones produced by the bilingual ones; in the Chinese task, 21% of the sentences produced by the monolingual translators were correct, against 66% of the bilingual ones. However, some inconsistencies in the quality evaluation were found:

> One puzzle is the low score for the professional human translators, as only two thirds of their translations were deemed to be correct. The example (a) [The

[9] This goal is outside the scope of this research project.

> example (a) used as reference the sentence "Torrential Rains Hit Western India, 43 People Dead", and the user-provided translation "Heavy Rains Plague Western India Leaving 43 Dead". None of the judges considered it a good translation.] shows such a translation, and it is hard to tell why it was deemed wrong by all three judges who looked at it. (Koehn (2010a), p. 542–544)

Other reasons for the low amount of correct translations were given: translators making mistakes (such as *mm* incorrectly translated as *cm*), translators not being thorough in their efforts (producing sentences that are not correct in English), names that got mistranslated, wrong relationship between entities (which is hard to correct without understanding the source language), and MT critical errors (where the MT output quality is too low for the translators to create a proper translation).

B.3. CASMACAT

Three field trials were conducted in the CASMACAT project, one every year (Mesa-Lao and Carl, 2012; Marcos Iglesias et al., 2013; Alabau et al., 2014) during the three years of the project. After conducting the first trial, that focused on post-editing, the evaluators suggested an autowrite function:

> **Autowrite functions**: An automatic write function could be implemented so as, while you start to edit a piece of text, the system automatically proposes a series of terms, phrases and sentences that are mined from existing TM or other translation alternatives offered by the MT. As the post-editor types, the system would dynamically predict alternative TM/MT translations that best complete the part of the sentence being post-edited. Separate areas of the editor could present all the different matches. This autocomplete feature is believed to speed up writing time by limiting the number of keystrokes and avoiding mistype errors. (Mesa-Lao and Carl (2012), p. 8)

To this end, they decided to test ITP for post-editing purposes.

Three different parameters were used to measure the performance of the users:

- *Fdur*, the "total production time": total translation time excluding pauses greater than 200 s, normalized with respect to the length of the produced segment
- *Kdur*, the "duration of coherent keyboard activity": the total translation time, excluding pauses greater than 5 s normalized with respect to the length of the produced segment
- *Pdur*, the "duration of coherent keyboard activity": the total translation time, excluding pauses greater than 1 s normalized respect to with the length of the produced segment

B.3.1. CASMACAT second field trial

The second field trial (Marcos Iglesias et al., 2013; Sanchis-Trilles et al., 2014) included an interactive post-editing task: as the user validates or mends the prefix, the system automatically generates an alternative completion for the rest of the translation.[10] The second trial took place in Celer Soluciones,[11] a Spanish translation company that also took part in the TransType2 trials, where 9 translators with diverse profiles (1 to 5 years of training, 2 to 20+ years of experience translating, all with post-editing experience, 3 of them also participated in the first trial) took part. They had to translate 3 blocks of 1 000 words each, working with three different assistance modalities:

- Traditional post-editing (P)
- Post-editing with interactivity (PI): the suffix to the right of the cursor updates whenever the user modifies the text
- Post-editing with advanced interactivity (PIA): similar to PI, but additional visual aid was provided to the user, such as visually showing the alignments between source and target words, or predicting the length of the final translation.

In this evaluation, P was faster than PI, and PI was faster than PIA (averages of 22 s, 27 s and 30 s *Kdur* for P, PI and PIA respectively),[12] but the fact that the users learnt how to use the tool during the evaluation was not taken into account. The users were also questioned about the tool: 3 preferred to work without any kind of assistance because they found the changes in the suffix distracting, 3 preferred PI and 4 preferred PIA. The users complained about the lack of a spell-checker and a concordancer, as well as about the system being unable to learn what kind of mistakes they were fixing.

B.3.2. CASMACAT third field trial

The third field trial (Alabau et al., 2014) was comprised of two different studies: a longitudinal study spanning 6 weeks, and an online/active learning study (that had a pre-trial and a proper trial).

Longitudinal study The longitudinal study was performed by five translators (one that mainly used post-editing, 3 that barely used post-editing, and one without post-editing experience) that translated 146 000 words during 6 weeks, alternating P and PI. The translators

[10] This is just a reinterpretation of how TransType 2 or Thot work: rather than telling the users that the system would offer a dynamically computed completion for the prefix of the translation, they told them the text to the right of the cursor (the suggestion) would automatically update as the text to the left gets edited (the prefix).

[11] `http://celersol.com/`

[12] When measuring actual translation time, the averages are 104 s, 80 s and 117 s for P, PI and PIA respectively; but this measures are skewed due to extreme outliers.

were given very specific guidelines and worked from home (which was usual for them), except for the first and last sessions, where an eye-tracking assembly was set in the main office and used to track the subjects' gaze as the task was being completed.

During this test, the translators were 5% faster and used 1% less keystrokes[13] when post-editing than when using PI (Average of 505 s *Fdur* for P, 529 s *Fdur* for PI). The users reported that they found it hard to adapt to the PI workflow, often typing long corrections without looking at the suggestions. While the eye-tracking setup severely affected the performance («[...] using an eye-tracker involved limited head movement and sometimes recalibration during the process of post-editing was necessary. Together, these aspects may have had a negative effect on participants' productivity, in other words, the data might show a lab effect.» (Alabau et al. (2014) p. 8)), it was useful to better understand how the users were translating. When using *Kdur* as the metric, P performance was roughly constant for all the different tasks distributed during the 6 weeks, but PI performance steadily improved, meaning the translators were adapting to PI; an extrapolation shows that PI performance would be faster than P in 8–12 weeks.

Finally, the users praised the PI mode for letting them to find equivalents and translation alternatives they did not think about, but only one user preferred it over P; 2 users saw no benefit at all in PI. They also said PI forced them to work against their PE training, and that it requires a controlled typing speed in order to notice and read the suggestions.

> [...] P01 and P02 provided feedback along these lines: "having to post-edited [sic] with interactivity demands a controlled typing speed and this is difficult to achieve when you are an experienced touch typist". Advanced touch typists need to be aware of the fact that they will only benefit from ITP when they stop overwriting most of the suggestions offered by the system. (Alabau et al. (2014) p. 19)

Pre-trial The second study was focused on the effects of a PI system that had new online and active learning capabilities. A pre-trial where 5 users translated from English to Danish was conducted using 3 different modes:

- Traditional post-editing (P)
- PI with online learning (PIO)
- PI with active learning (PIAL)

PIO statistically significantly increased the translation speed (2.5 words/hour more than PI), but the effort reduction (measured in keystrokes per segment) was not statistically significant.

[13] Average of averages, actual data not available.

> On the negative side, dynamic corrections at the lexical level were not always appropriate. [...] This inappropriate dynamic correction then had to be revised [...], which lead to decreased efficiency in the post-editing process. (Alabau et al. (2014) p. 12)

The test concluded that the results obtained using PI with online learning (PIO) were more promising than the ones obtained with active learning (PIAL).

Third field trial The field trial used 7 translators, 4 of those also took part in the longitudinal study. The test had the users translating medical specialized texts from the EMEA corpus from English to Spanish, while using an eye-tracking set-up. The users were also provided with a specialized bilingual concordancer as a external tool, and told they could use any other tool they needed (the most used being Linguee). During the test, the changes in the model due to online learning were automatically propagated to the rest of non-validated segments, a functionality that the users praised, as they saw how the system learned the specialized medical terminology. The online learning strategy helped the users to translate using less insertions (from 79.53 to 68.73 insertions per segment) and deletions (from 70.71 to 36.94 deletions per segment), and the total translation time was slightly lower but not statistically significant (using the Z-test, *Fdur* Z=-1.745, p=0.081; *Kdur* Z=-0.524, p=0.601). When the data was analyzed, it was found that users were using external tools to check some of the post-edited option; after removing the time used to query the external tools, the difference was significant (*Fdur* Z=-3.148, p=0.002; *Kdur* Z=-2.524, p=0.012)

B.4. Lilt

B.4.1. Lilt field trial

An human evaluation of Lilt (Green et al., 2014) was also carried out. The test had 16 professional translators for French to English, and 16 for English to German, that were paid \$0.085 per word, and \$10 for completing the training module; 3 users had to be excluded from the results, as one misunderstood the instructions, another one skipped the training module, and a third one triggered a bug in the application and did not get logged. Users were under time pressure: the translation interface had a visible idle timer with the maximum allowed time of 3 min, and the translations that surpassed that threshold were also discarded. The trial focused on testing the performance of a *predictive translation memory* (PTM), that combines a traditional TM with MT based ITP, against traditional post-editing. The MT system was tuned against the domain of the test corpus. Each task had 3000 words, and the users were encouraged to complete it in the same day.

Users were slower using PTM than traditional PE (French→English: 46 s/sentence for PE, 54.6 s/sentence for PTM, 18% slower; English→German: 51.8 s/sentence for PE,

63.3 s/sentence for PTM, 22% slower), but, as in the CASMACAT field tests, the performance of the users steadily improved as the test was being carried out, without surpassing the performance of PE. Users reported that after enough training they could be faster using PTM, but that it is more work intensive than PE. They were also surprised about the quality of the output compared to Google Translate, as they did not know the MT engine was tuned for this task. The translators also found that PTM constrained their translation progress and made it more formulaic and less stylistic, as some suggestions did not fit with the planned translation; 8 users reported that PTM is the most helpful system, and 11 think PE helped them to translate faster. More than 99% of the editing events were performed via the keyboard; for French→English, 71% of the characters come from the suggestions, and 18% from key presses; for English→German, 65% come from the suggestions and 34% from key presses.

Appendix C

Corpora used

C.1. Corpus DGT-TM

The Directorate-General for Translation (DGT) is an European Commission entity that provides translation services for all the official European languages. The corpus is a translation memory containing the *Acquis Communautaire*, the European common law, and uses English as a pivot language: most languages have around 2 million translation units, but for the official languages of countries that joined the European Union between 2007 and 2011, such as Romania and Bulgaria, that have fewer units (Steinberger et al., 2012).

C.2. Corpus El Periódico

El Periódico is a general-interest newspaper that was first published in 1978. Since 1997, two different editions get published daily with identical contents, one in Spanish and a second one in Catalan. The news are originally written in Spanish, then translated automatically and post-edited. This corpus is comprised of 819 437 sentences extracted from news pieces of *El Periódico de Catalunya*,[1] that were automatically aligned.

C.3. Corpus Europarl

The Europarl v7 Corpus (Koehn, 2005) is a collection of proceedings from the European Parliament. The corpora was built in a mostly automatic process: the Proceedings of the European Parliament (available in HTML) were crawled to obtain individual documents, then the documents were aligned between the different languages, tokenized, segmented, and the

[1] `http://www.elperiodico.cat/`

segments got aligned. The corpora has documents in 21 different European languages paired with the English translation; the size varies between 2 million sentences for English–French to 400 000 sentences for English–Bulgarian.

C.4. Corpus News Commentary

The News Commentary Corpus is a corpus created as training data resource for the Conference for Statistical Machine Translation Evaluation Campaign.[2] The corpus is comprised of political and economic commentary extracted from Project Syndicate,[3] a non-for-profit organization focused on providing high-quality advocacy journalism to media, paid for press organizations in developed countries and for free for those in the developing world. The corpus contains documents in 11 different languages paired with the English translation; the size varies between 200 000 sentences for languages like French, Russian and Spanish to 600 sentences for Japanese.

C.5. Corpus United Nations

The United Nations Parallel Corpus 1.0 (Ziemski et al., 2016) is comprised of official records and other parliamentary documents of the United Nations that are public domain, available in the six official languages of the United Nations: English, Spanish, French, Arabic, Russian and Simplified Chinese. The corpus has between 15.9 million sentences for Simplified Chinese and 18.8 million for Russian.

[2] http://www.statmt.org/wmt16/
[3] https://www.project-syndicate.org/

Appendix D

Guidelines for the second human evaluation

Instructions for the OmegaT task

Translated from Spanish. The actual handout is included in page 131.

OmegaT is a free/open-source commercial CAT (computer assisted translation) tool. The two CAT modalities that you are going to evaluate have been integrated into OmegaT as plugins.

The task consists in translating 15 sentences extracted from the English wikipedia to Spanish. Try to avoid word-by-word translation or transliterations, and focus on generating native-looking translations. Pretend that the translations will be published in the Spanish wikipedia.

Try to plan the translation in such way you do not have to go back to the already translated part (either with the mouse or with the arrow keys).

Once you are done translating the sentence, press `Enter` to start working in the next one. Do not read or plan the translation of the next sentence before selecting it, as it would modify the results of the test.

As you translate, you may get suggestions. To accept a suggestion, you can:

- Choose the one you want to use with the arrow keys then press `Enter` to accept it.
- Choose the one you want to use and use `Tab` to select it word by word: the selected part will be highlighted; press `enter` to select the suggestion, or the arrow keys to choose a different suggestion.
- Press `Alt`+# (the # appears at the start of the suggestion) to accept a suggestion.

- Click on a suggestion to accept it.

If no suggestion is useful, keep typing the translation. You can use `Backspace` to delete part of the suggestion after selecting it.

Questions

You must answer questions 1–4 using the Likert scale (1: strongly disagree – 5: strongly agree). In questions 5 and 6, sort the three different systems. Question 7 is open ended.

1. Is the interface comfortable?
2. Did the interface help you translating?
3. Would you use the tool if you need to translate texts?
4. Did suggestions help you?
5. Order the systems by your perceived translation speed.
6. Order the systems by the quality of the suggestions.
7. Suggestions.

Instrucciones para la tarea con OmegaT

OmegaT es una plataforma comercial de CAT (computer assisted translation) de código abierto. Los métodos de CAT que vas a probar han sido incorporados a la plataforma como plugin.

La tarea consiste en traducir 15 frases de la wikipedia en inglés a español. Procura evitar traducir palabra a palabra o transliterar, generando traducciones naturales. Piensa que la traducción será publicada en la wikipedia en español.

Procura planificar la traducción de tal forma que no tengas que volver para teclear en lo ya traducido (ya sea con el ratón o con las flechas de dirección).

Una vez termines de traducir la frase, pulsa `Intro` para pasar a la siguiente. No leas o planifiques la traducción de la siguiente frase sin antes pasar a ella, ya que modificaría los resultados del test.

Mientras traduces, es posible que se te ofrezcan sugerencias. Para aceptar una sugerencia, puedes:

- Seleccionar la que quieres con las flechas de dirección y pulsar `Intro` para usar toda la sugerencia
- Seleccionar la que quieres con las flechas de dirección y pulsar `Tab` para seleccionar palabra a palabra la sugerencia: la parte de la sugerencia que será usada aparecerá resaltada; pulsa intro para aceptarla, o pulsa las teclas de dirección para seleccionar otra sugerencia
- Pulsar `Alt`+N^{o} (el N^{o} aparece al principio de la sugerencia) para usar toda la sugerencia
- Hacer click para usar toda la sugerencia

Si ninguna sugerencia te parece util, sigue tecleando la traducción. Puedes usar la tecla `Retroceso` para borrar parte de la sugerencia tras elegirla.

Preguntas

Las preguntas 1–4 usan la escala likert (1: muy en desacuerdo - 5: muy de acuerdo). Las preguntas 5 y 6 ordenan los 3 sistemas. La pregunta 7 es abierta.

1. La interfaz que has utilizado, ¿es cómoda?
2. ¿Te ha ayudado a traducir?
3. En caso de tener que traducir textos, ¿usarías la interfaz?
4. ¿Te han ayudado las sugerencias?

5. ¿Con qué bloque has traducido más rápido? ¿Y mas lento?
6. ¿Qué bloque te ha dado las mejores sugerencias?
7. Sugerencias

Appendix E

Guidelines for the third human evaluation

Instructions

Translated from Spanish. The actual handout is included in page 136.

OmegaT is a free/open-source commercial CAT (computer assisted translation) tool. The two CAT modalities that you are going to evaluate have been integrated into OmegaT as plugins (changes that add new functions or extend already existing functions).

The evaluation duration is of three hours, and it is split in three different tasks. Each task has a fixed duration, and is composed of 100 unrelated sentences. You do not need to translate all the sentences in each task, but to translate at a pace where you can generate translations that can be published, without skipping any sentence. Do not worry about formatting, as the translations' format will be revised before publishing.

Each task has 2 phrases: a previous phase of five minutes to get familiar with the kind of assistance and ask questions, and a second one of 40 minutes that will be monitored. Each phase has an independent OmegaT project.

As you translate, you may get one or more options to continue the translation. If any of them are useful, you can

Durante alguna de las tareas es posible que se os sugieran una o más opciones para continuar la traducción. Si alguna sugerencia es útil para continuar la traducción, se puede:

- Choose the one you want to use with the arrow keys then press `Enter` to accept it.

- Choose the one you want to use and use `Tab` to select it word by word: the selected part will be highlighted; press `enter` to select the suggestion, or the arrow keys to choose a different suggestion.

- Press `Alt`+# (the # is the position of the suggestion in the list) to accept a suggestion.
- Click on a suggestion to accept it.

If no suggestion is useful, you can ignore it and keep typing the translation. The accepted suggestions can be then edited, but try to not select suggestions that would take more work to edit than typing the text. In the tasks with assistance, you should try not to go back to modify the already translated part, as no assistance method can cope with this behaviour.

When the translation is complete, you can go to the next one pressing `Enter`. You should not plan the next translation before selecting the next segment, or selecting the next segment then checking the previous one.

Questionarie between tasks

	Strongly disagree	Disagree	Agree	Strongly agree
Suggestions helped me to translate faster				
Suggestions helped me to translate with less effort				
Suggestions helped me to increase the quality of the translation				
Suggestions made the translation task more enjoyable				

	Disagree	Agree
I would rather get shorter suggestions		
I would rather get longer suggestions		
I would rather get suggestions more often		
I would rather get suggestions less often		
I would rather get more suggestions offered at the same time		
I would rather get less suggestions offered at the same time		

Final questionarie

Sort the three tasks according to your translation speed.

Slower		Faster

Sort the three tasks according to the easiness of the task.

Harder		Easier

Please, add suggestions and coments about the CAT modality that would lead to faster and effortless translations.

Please, add any other comment you may have.

Instrucciones

OmegaT es una plataforma de CAT (computer-assisted translation) libre/de código fuente abierto. Los métodos de CAT que vas a probar han sido incorporados a la plataforma como extensiones (modificaciones que añaden nueva funcionalidad o extienden alguna que ya exista).

La prueba se durará unas 3 horas, durante las cuales se trabajará en 3 tareas diferentes. Cada tarea tiene una duración fija, y se compone de 100 frases independientes. No es necesario traducir todas las frases de cada bloque, sino traducir a un ritmo al cual seáis capaces de generar traducciones que puedan ser publicadas, sin saltarse ninguna. No os preocupeis por el formato de las traducciones, suponed que será revisado y reparado antes de ser publicadas.

Cada tarea tiene 2 fases: una fase previa de 5 minutos para familiarizarse con el tipo de asistencia y plantear dudas, y una segunda de 40 minutos que será monitorizada. Se proveerá de un proyecto de OmegaT para cada fase.

Durante alguna de las tareas es posible que se os sugieran una o más opciones para continuar la traducción. Si alguna sugerencia es útil para continuar la traducción, se puede:

- Usando las flechas, seleccionar una de las sugerencias ofrecidas, y pulsar `Intro` para usar toda la sugerencia.
- Pulsar Tab para seleccionar palabra a palabra la sugerencia que está seleccionada: la parte de la sugerencia que será usada aparecerá resaltada; pulsa intro para aceptarla.
- Pulsar `Alt` + la posición de la sugerencia en la lista para usar toda la sugerencia (por ejemplo, `Alt`+3 elige la tercera sugerencia de la lista).
- Hacer click con el ratón para usar toda la sugerencia.

Si la sugerencia no es útil, se puede ignorar y continuar tecleando la traducción. Podéis editar parte de la sugerencia tras elegirla, pero procurad que sean ediciones sencillas que no requieran más trabajo del que te ahorra aceptar la sugerencia. En las tareas con asistencia, se debe procurar no volver atrás para modificar parte de la traducción, ya que ninguno de los métodos de asistencia está preparado para este comportamiento.

Una vez la traducción esté completa, se puede pasar a la siguiente pulsando `Intro`. Se debe evitar planificar la traducción del siguiente segmento sin seleccionarlo, así como pasar al siguiente segmento antes de dar por válido el segmento actual.

Cuestionario entre secciones

Este cuestionario se pasa al final de cada sección (menos la tarea de introducción). Se usa una escala Likert de 4 puntos (1 = muy en desacuerdo, 4 = muy de acuerdo), o de 2 puntos (1 = en desacuerdo, 4 = de acuerdo).

	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo
Las sugerencias me han ayudado a traducir más rápido				
Las sugerencias me han ayudado a traducir con menos esfuerzo				
Las sugerencias me han ayudado a mejorar la calidad de la traducción				
Las sugerencias han hecho la tarea de traducción más agradable				

	En desacuerdo	De acuerdo
Prefiero que se ofrezcan sugerencias más cortas		
Prefiero que se ofrezcan sugerencias más largas		
Prefiero que se ofrezcan sugerencias más a menudo		
Prefiero que se ofrezcan sugerencias con menor frecuencia		
Prefiero que se ofrezcan más sugerencias a la vez		
Prefiero que se ofrezcan menos sugerencias a la vez		

Cuestionario final

Ordena las 3 tareas (U para sin asistencia, B para la primera prueba con asistencia, BT para la segunda prueba con asistencia) en función de la velocidad de traducción.

Más lento		Más rápido

Ordena las 3 tareas (U para sin asistencia, B para la primera prueba con asistencia, BT para la segunda prueba con asistencia) en función de la facilidad de traducción.

Más difícil		Más fácil

Por favor, indica sugerencias y comentarios sobre la técnica usada que puedan llevar a mejorar la velocidad y facilidad con la que traduces los textos.

Por favor, indica otros comentarios sobre la prueba que consieres oportunos.

Index of abbreviations

Index of symbols

Bibliography

Alabau, V., Carl, M., García Martínez, M., Mesa-Lao, B., Ortiz-Martínez, D., Rodrigues, S., and Schaerrer, M. (2014). D6.3: Analysis of the third field trial. Technical report, CASMACAT. `http://www.casmacat.eu/uploads/Deliverables/d6.3.pdf` Accessed: 30/05/2017.

Alabau, V. and Casacuberta, F. (2012). Study of electronic pen commands for interactive-predictive machine translation. In *Proceedings of the International Workshop on Expertise in Translation and Post-editing Research and Applications*.

Alabau, V., González-Rubio, J., Leiva, L., Ortiz-Martínez, D., Sanchis-Trilles, G., Casacuberta, F., Mesa-Lao, B., Bonk, R., Carl, M., and Martinez, M. G. (2013). User evaluation of advanced interaction features for a computer-assisted translation workbench. In *Proceedings of Machine Translation Summit XIV*.

Axelrod, A., He, X., and Gao, J. (2011). Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 355–362.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representations (ICLR)*.

Brousseau, J., Drouin, C., Foster, G. F., Isabelle, P., Kuhn, R., Normandin, Y., and Plamondon, P. (1995). French speech recognition in an automatic dictation system for translators: the transtalk project. In *Eurospeech*.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Card, S. K., Moran, T. P., and Newell, A. (1980). The keystroke-level model for user performance time with interactive systems. *Commun. ACM*, 23(7):396–410.

Card, S. K., Newell, A., and Moran, T. P. (2000). *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

Carl, M., Dragsted, B., and Jakobsen, A. L. (2011). On the systematicity of human translation processes. In *Proceedings of Translation Careers and Technologies: Convergence Points for the Future (Tralogy)*.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of the Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Cubel, E., González, J., Lagarda, A., Casacuberta, F., Juan, A., and Vidal, E. (2003). Adapting finite-state translation to the TransType2 project. In *Proceedings of the 8th International Workshop of the European Association for Machine Translation and the 4th Controlled Language Applications Workshop Dublin City University Joint Conference, Ireland*.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314.

DGT annual activity report (2015). Annual activity report. `https://ec.europa.eu/info/sites/info/files/activity-report-2015-dg-t_april2016_en.pdf`. Accessed: 30/05/2017.

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification*. John Wiley and Sons Inc., second edition.

Esplà-Gomis, M., Sánchez-Martínez, F., and Forcada, M. L. (2012). A simple approach to use bilingual information sources for word alignment. *Procesamiento del Lenguaje Natural*, 49:93–100.

Espla-Gomis, M., Sánchez-Martínez, F., and Forcada, M. L. (2015). Using machine translation to provide target-language edit hints in computer aided translation based on translation memories. *Journal of Artificial Intelligence Research*, 53:169–222.

Forcada, M. L., Ginestí-Rosell, M., Nordfalk, J., O'Regan, J., Ortiz-Rojas, S., Pérez-Ortiz, J. A., Sánchez-Martínez, F., Ramírez-Sánchez, G., and Tyers, F. M. (2011). Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144.

Foster, G. (2000). A maximum entropy/minimum divergence translation model. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 45–52.

Foster, G., Isabelle, P., and Plamondon, P. (1997). Target-text mediated interactive machine translation. *Machine Translation*, 12(1-2):175–194.

Foster, G., Langlais, P., and Lapalme, G. (2002). *Text prediction with fuzzy alignments*. Springer.

González-Rubio, J., Ortiz-Martínez, D., and Casacuberta, F. (2012). Active learning for interactive machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 245–254.

Gori, M. and Tesi, A. (1992). On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76–86.

Green, S., Chuang, J., Heer, J., and Manning, C. D. (2014). Predictive translation memory: A mixed-initiative system for human language translation. In *Proceedings of the 27th annual ACM Symposium on User Interface Software and Technology*, pages 177–187.

Haddow, B. and Koehn, P. (2012). Analysing the effect of out-of-domain data on SMT systems. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 422–432, Montreal, Canada.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Hall, M. A. (1998). *Correlation-based Feature Subset Selection for Machine Learning*. PhD thesis, University of Waikato, Hamilton, New Zealand. `www.cs.waikato.ac.nz/~mhall/thesis.pdf`.

Hang, L. (2011). A short introduction to learning to rank. *IEICE Transactions on Information and Systems*, 94(10):1854–1862.

Heafield, K. (2011). Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 187–197, Stroudsburg, PA, USA.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Johnson, H., Martin, J. D., Foster, G. F., Kuhn, R., et al. (2007). Improving translation quality by discarding most of the phrasetable. In *EMNLP-CoNLL*, volume 2007, pages 967–975.

Junczys-Dowmunt, M. (2012). Phrasal rank-encoding: Exploiting phrase redundancy and translational relations for phrase table compression. *The Prague Bulletin of Mathematical Linguistics*, 98:63–74.

Kay, M. and Martins, G. R. (1970). *The MIND system*. RAND Corporation.

Knowles, R. and Koehn, P. (2016). Neural interactive translation prediction. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*, volume 1: MT researchers track, pages 107–120.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In *Proceedings of the conference on Empirical Methods on Natural Language Processing (EMNLP 2004)*, pages 388–395.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the X Machine Translation summit*, volume 5, pages 79–86.

Koehn, P. (2009a). A process study of computer-aided translation. *Machine Translation*, 23(4):241–263.

Koehn, P. (2009b). A web-based interactive computer aided translation tool. In *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, pages 17–20.

Koehn, P. (2010a). Enabling monolingual translators: Post-editing vs. options. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 537–545.

Koehn, P. (2010b). *Statistical Machine Translation*. Cambridge University Press.

Koehn, P. (2016). Recent trends in computer-aided translation. *Retrieved from* `https://ufal.mff.cuni.cz/mtm16/files/17-computer-aided-translation-philipp-koehn.pdf`.

Koehn, P. and Haddow, B. (2009). Interactive assistance to human translators using statistical machine translation methods. *Machine Translation Summit XII*.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180.

Langlais, P. and Lapalme, G. (2002). Trans type: Development-evaluation cycles to boost translator's productivity. *Machine Translation*, 17(2):77–98.

Langlais, P., Loranger, M., and Lapalme, G. (2002). Translators at work with TRANSTYPE: Resource and evaluation. In *Proceedings of the Third International Conferene on Language Resources and Evaluation (LREC)*, pages 2128–2134.

Langlais, P., Sauvé, S., Foster, G., Macklovitch, E., and Lapalme, G. (2000). Evaluation of TransType, a computer-aided translation typing system: a comparison of a theoretical-and a user-oriented evaluation procedures. In *Conference on Language Resources and Evaluation (LREC)*, pages 641–648.

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, 140:1–55.

Macklovitch, E. (2006). TransType2: The last word. In *Proceedings of the 5th International Conference on Languages Resources and Evaluation (LREC 06)*, pages 167–172.

Marcos Iglesias, E., Pellegrino, M., Carl, M., García Martínez, M., Mesa-Lao, B., and Underwood, N. (2013). D6.2: Analysis of the second field trial. Technical report, CASMACAT. `http://www.casmacat.eu/uploads/Deliverables/d6.1.pdf` Accessed: 30/05/2017.

Mesa-Lao, B. and Carl, M. (2012). D6.1: Analysis of the first field trial. Technical report, CASMACAT. `http://www.casmacat.eu/uploads/Deliverables/d6.1.pdf` Accessed: 30/05/2017.

Neco, R. P. and Forcada, M. L. (1997). Asynchronous translations with recurrent neural nets. In *Neural Networks, 1997., International Conference on*, volume 4, pages 2535–2540. IEEE.

Nissen, S. (2003). Implementation of a fast artificial neural network library (FANN). Technical report, Department of Computer Science University of Copenhagen (DIKU). `http://fann.sf.net`.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Och, F. J., Zens, R., and Ney, H. (2003). Efficient search for interactive statistical machine translation. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 387–393.

Ortiz-Martínez, D. (2011). *Advances in Fully-Automatic and Interactive Phrase-Based Statistical Machine Translation*. PhD thesis, Universitat Politècnica de València.

Ortiz-Martínez, D. and Casacuberta, F. (2014). The new Thot toolkit for fully automatic and interactive statistical machine translation. In *Proc. of the European Association for Computational Linguistics (EACL): System Demonstrations*, pages 45–48, Gothenburg, Sweden.

Ortiz-Martínez, D., Sanchís, G., Casacuberta, F., Alabau, V., Vidal, E., Benedí, J.-M., González-Rubio, J., Sanchís, A., and González, J. (2012). The CASMACAT project: The next generation translator's workbench.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting at Association for Computational Linguistics (ACL)*, pages 311–318.

Pérez-Ortiz, J. A., Calera-Rubio, J., and Forcada, M. L. (2001). Online text prediction with recurrent neural networks. *Neural Processing Letters*, 14(2):127–140.

Pérez-Ortiz, J. A., Torregrosa, D., and Forcada, M. L. (2014). Black-box integration of heterogeneous bilingual resources into an interactive translation system. *EACL 2014 Workshop on Humans and Computer-assisted Translation*, pages 57–65.

Ranta, A. (2004). Grammatical framework. *Journal of Functional Programming*, 14(2):145–189.

Rehm, G. and Uszkoreit, H. (2013). *META-NET strategic research agenda for multilingual Europe 2020*. Springer Publishing Company, Incorporated.

Richard, M. D. and Lippmann, R. P. (1991). Neural network classifiers estimate bayesian a posteriori probabilities. *Neural computation*, 3(4):461–483.

Rousseau, A. (2013). Xenc: An open-source tool for data selection in natural language processing. *The Prague Bulletin of Mathematical Linguistics*, 100:73–82.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive Modeling*, 5(3):1.

Sánchez-Martínez, F. (2011). Choosing the best machine translation system to translate a sentence by using only source-language information. In Forcada, M. L., Depraetere, H., and Vandeghinste, V., editors, *Proceedings of the 15th Annual Conference of the European Association for Machine Translation*, pages 97–104, Leuven, Belgium. European Association for Machine Translation.

Sanchis-Trilles, G., Alabau, V., Buck, C., Carl, M., Casacuberta, F., García-Martínez, M., Germann, U., González-Rubio, J., Hill, R. L., Koehn, P., et al. (2014). Interactive translation prediction versus conventional post-editing in practice: a study with the CasMaCat workbench. *Machine Translation*, 28(3-4):217–235.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(4):623–656.

Steinberger, R., Eisele, A., Klocek, S., Pilos, S., and Schlüter, P. (2012). Dgt-tm: A freely available translation memory in 22 languages. In *Proceedings of the 8th international conference on Language Resources and Evaluation (LREC)*, pages 454–459.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Toma, P. (1977). Systran as a multilingual machine translation system. In *Proceedings of the Third European Congress on Information Systems and Networks, Overcoming the language barrier*, pages 569–581.

Torregrosa, D., Forcada, M. L., and Pérez-Ortiz, J. A. (2014). An open-source web-based tool for resource-agnostic interactive translation prediction. *The Prague Bulletin of Mathematical Linguistics*, 102(1):69–80.

Torregrosa, D., Forcada, M. L., and Pérez-Ortiz, J. A. (2016). Ranking suggestions for black-box interactive translation prediction systems with multilayer perceptrons. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*, volume 1: MT researchers track, pages 65–78.

Torregrosa, D., Pérez-Ortiz, J. A., and Forcada, M. L. (2017). Comparative human and automatic evaluation of glass-box and black-box approaches to interactive translation prediction. *The Prague Bulletin of Mathematical Linguistics*, 108(1):97–108.

Vauquois, B. (1968). Ca survey of formal grammars and algorithms for recognition and transformation in mechanical translation. *International Federation for Information Processing, 1968*, pages 1114–1122.

Žabokrtský, Z., Ptáček, J., and Pajas, P. (2008). Tectomt: Highly modular mt system with tectogrammatics used as transfer layer. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 167–170.

Ziemski, M., Junczys-Dowmunt, M., and Pouliquen, B. (2016). The United Nations parallel corpus v1.0. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 3530–3534, Paris, France.