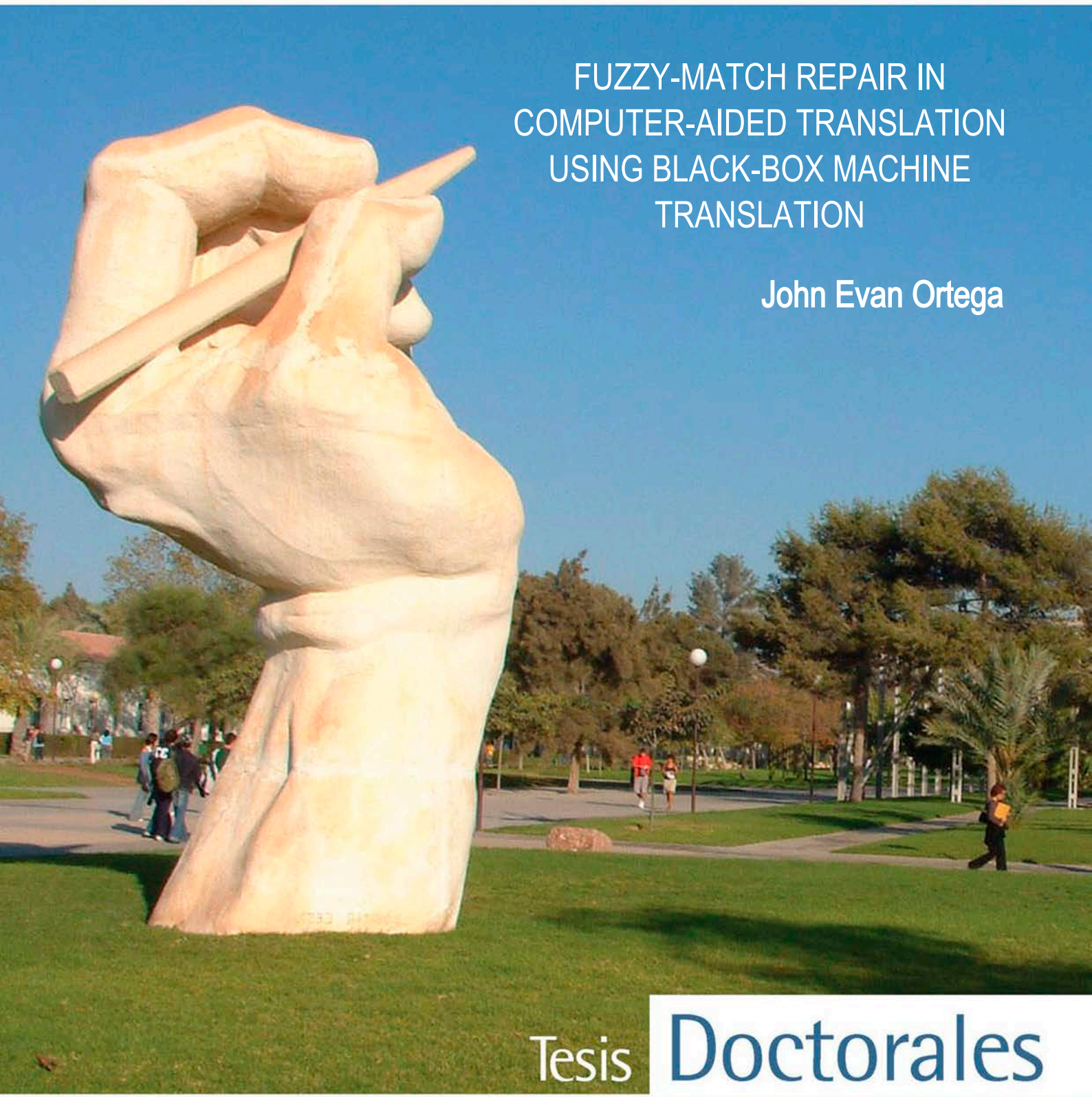
Universitat d'Alacant
Universidad de Alicante

FUZZY-MATCH REPAIR IN COMPUTER-AIDED TRANSLATION USING BLACK-BOX MACHINE TRANSLATION

John Evan Ortega



Tesis Doctorales
UNIVERSIDAD de ALICANTE

Unitat de Digitalització UA
Unidad de Digitalización UA

Department de Llenguatges i Sistemes Informátics
Departamento de Lenguajes y Sistemas Informáticos

Fuzzy-Match Repair in Computer-Aided Translation Using Black-Box Machine Translation

John Evan Ortega

Tesis presentada para aspirar al grado de
DOCTOR POR LA UNIVERSIDAD DE ALICANTE

MENCIÓN DE DOCTOR INTERNACIONAL

Doctorado en Informática

Dirigida por:
Felipe Sánchez Martínez

Marzo 2021

A la memoria de mi padre,
Juan Evangelista Ortega Morote
John Evan Ortega Sr.

Acknowledgments

First and foremost, I would like to thank my thesis director: Felipe Sánchez-Martínez. Felipe has dedicated the time and effort to make this thesis a success. The thesis would have been impossible to complete without Felipe's countless hours of help. I have learned so much about my writing style from Felipe and he has really helped me improve the way I approach scientific writing. I would also like to thank Mikel L. Forcada for helping out with the fuzzy-match repair method and other logistics during the initial development, his assistance will be always remembered.

An enormous token of gratitude goes to the thesis board and committee who have decided to take the time out of their busy schedules to review my thesis and listen to my defense. We are in the middle of a worldwide pandemic (COVID) and I cannot thank the committee and board enough for their overall support and dedication to their field during these turbulent times.

A huge acknowledgment is due to my friends from the *Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Alicante*, specifically Miquel Esplá-Gomis and Daniel Torregrosa. I cannot repeat enough on how their kind words and overall understanding kept me aligned with my thesis goals. Juan Antonio Pérez-Ortiz, as a member of the department and the thesis board, has been a silent partner who has helped me from the beginning when I needed an objective person to help out with the review of the fuzzy-match repair implementation.

I cannot thank enough my colleagues from Nuance Communications. They were a great source of inspiration and allowed me to work on my thesis while attempting to help patients in a clinical setting. Specifically, I would like to thank part of the original team: Brian Delaney, Girija Yegnanarayanan, Neil Barrett, Ravi Kondadadi, and Óscar Ferrández-Escámez. I am in such great debt to Óscar for his recommendation to complete a doctorate degree and his amazing work ethic which helped guide me through my first steps in the natural language processing sector.

My high acknowledgments are due to the collaborating university in the United States, New York University (NYU). Adam Meyers, as a lead researcher and professor at NYU, has taken the time out to review my research several times and provided life-changing opportunities of which I am in great debt. Adam is more than a colleague and cannot be thanked enough. Kyunghyun Cho, a professor in the NYU data science

department, deserves more than a thank you. He has co-authored several papers with me and much more. I would like to thank him again for his participation on the panel at the workshop I hosted: *Post-Editing in Modern-Day Translation at the Fourteenth Conference of the Association for Machine Translation in the Americas* (PEMDT at AMTA). Kyunghyun has provided me with the background needed to apply advanced statistics and neural networks to solve several problems. I would also like to thank Kyunghyun's student at NYU, Katharina Kann. She has helped on several occasions with German evaluation and other natural language processing tasks.

I must acknowledge other collaborators from the Machine Translation (MT) industry. First, I would like to thank Rebecca Knowles and Philipp Koehn for their help in testing the fuzzy-match repair method on multiple MT paradigms. Philipp was one of the few researchers that responded to my email communications from the first day I became interested in MT, before I began doctorate studies. I would also like to thank Marco Turchi and Matteo Negri from *Fondazione Bruno Kessler* because you cannot mention one of their names without mentioning the other. Marco has been a great source of inspiration and assistance since the first time I met him at AMTA 2014 in Vancouver, Canada. I should also thank Kevin Knight, who needs no introduction. Kevin, you made the MT sector seem like something viable to do. I must thank you for constantly offering your help and keeping me in mind over the past eight years since I first met you with my "Spanish" shoes. Additionally, I hugely appreciate the help from Shabnam Tafreshi, an associate at Columbia University. She has been a great source of assistance for the thesis paperwork and has always been available to discuss natural language processing ideas.

I will not forget the enormous support that I have received from family and friends. They have all endured and supported my thesis studies; they make me proud to be the first person to receive a doctorate degree in my family. This is another beginning for the Ortega family, I truly believe that our family was destined to achieve great things in life despite hardships handed to us by society during the earlier years. My father, Don Juan Evangelista Ortega Morote (dad), I wish you were here to see this moment. The unabated words of wisdom from an immigrant father while growing up in the United States kept me focused. I would probably not be alive or even writing this article without my father's advice, he is, and has always been, my hero. I must acknowledge that Rosa Losada-Moreiras has provided support ranging from translations to review. Jim Freyre has been more than a brother to me and provided an unsurpassed vision on the prize. Ani Mccown must also be recognized for the numerous amount of times for her help and support. I would like to thank Miriam (Ortega) Waag for her support, especially in those hard times dealing with the passing of our fathers. Javier Pablos-Arroyo has been the best friend that I could ask to have. Our relationship as friends for nearly twenty years has not died and I am sure that Javier will be around for the next twenty years as I will be for him. Javier's constant ideas and support during every life event since I have known him will not go unnoticed. Also, José Cegrí-Fiérrez and Sandra Fabiola-Ravello deserve a huge thanks for all of their hospitality and help. I

must thank Norvey Fernández-Maldonado and family, friends until the end, the amount of patronage and service they have provided since I met them cannot be counted on twenty hands.

To finalize the acknowledgments, I would like to thank everyone else that I have been lucky enough to interact with in the past seven years while working on my thesis. This includes the University of Alicante and its associates that helped make the thesis possible and all other acquaintances since I began to work on the thesis. Although I have not mentioned you by name here, I am greatly appreciative for knowing you during these times and hope that this serves as a small token of appreciation.

John E. Ortega
Alicante, January 15, 2021

Resumen en español

Introducción

La traducción asistida por ordenador (TAO) podría definirse como el uso de un programa informático que ayuda a un traductor en su tarea de traducir documentos en un idioma denominado "idioma origen" a otro idioma denominado "idioma destino". Las herramientas TAO se denominaban originalmente "herramientas de traducción humana asistida por ordenador" (Bowker, 2002) debido a que el proceso de traducción se dejaba al usuario. Las herramientas TAO proporcionan diferentes recursos para ayudar con la traducción, pero el acto de traducir un segmento origen queda totalmente en manos del usuario. Algunos de los recursos disponibles para los usuarios en las herramientas TAO son: correctores ortográficos y gramaticales, herramientas de búsqueda, memorias de traducción y motores de traducción automática (Federico et al., 2012). La mayoría de las herramientas TAO utilizan memorias de traducción (MT) como recurso principal para ayudar en el proceso de traducción; de hecho, cuando en la literatura reciente se habla de herramientas TAO se sobreentiende que son herramientas TAO basadas en MT.

Una MT es un repositorio que contiene unidades de traducción (UT), esto es, pares de segmentos paralelos (s,t) en lengua origen junto con su traducción a la lengua meta. Generalmente, los segmentos constan de frases simples; pero también podrían constar de otras unidades de texto (por ejemplo, párrafos).

Las herramientas TAO desempeñan un papel fundamental en el flujo de trabajo de los traductores profesionales tal y como se describe con más detalle en la Sección 1.1.2. La responsabilidad subyacente de la herramienta TAO basada en MT es comparar un nuevo segmento en lengua origen a traducir s' con los segmentos en lengua origen existentes en la MT. Si en la comparación se encuentra un segmento en lengua origen *similar* al que se va a traducir, la UT correspondiente (s,t) se muestra al traductor para la posedición de t o para su aceptación inmediata cuando haya una concordancia exacta (ambos segmentos origen sean idénticos). Si se produce la posedición, el segmento origen a traducir s', junto con su traducción t' se guardan para su uso futuro. Es responsabilidad de la herramienta TAO segmentar el documento a traducir e implementar el método de búsqueda en la MT de las UT cuyo segmento origen sea similar al que se va a traducir. El algoritmo que calcula la similitud de dos segmentos

es de particular interés para nuestro método de reparación de concordancias parciales (RCP) y se describe con más detalle en la Sección 1.1.1.

Las herramientas TAO que pueden ayudar a un traductor durante su flujo de trabajo normal no solo se basan en MT, ya que otro de los principales componentes disponibles para el traductor es la *traducción automática* (TA), la cual se define como la traducción que se realiza por medio de un programa informático de textos de un idioma a otro. Aunque la TA sea una manera rápida de traducir no llega a tener el grado de complejidad necesario para traducir los lenguajes naturales hablados por el ser humano, siendo recurrente el echo de que palabras de un idioma no puedan traducirse directamente a otro idioma. La principal diferencia entre un sistema TAO y un sistema de TA radica en que en un sistema TAO el traductor produce la traducción; mientras que en la TA, el usuario no juega ningún papel activo en la traducción aparte de corregir la salida.

El flujo de trabajo típico para traducir un documento origen mediante una herramienta TAO basada en MT consta de cuatro pasos: (1) segmentación del documento origen, (2) búsqueda en la MT para cada segmento origen que se vaya a traducir de UT cuyo segmento origen sea similar al que se traducirá, (3) aceptación o modificación (posedición) de propuestas de la MT, y (4) generación del documento en lengua meta. El flujo de trabajo básico se describe en la Sección 1.1.2.

El flujo de trabajo básico puede ser modificado para incluir un sistema de TA y traducir los segmentos (s') cuando no haya ninguna concordancia aproximada disponible con un porcentaje de coincidencia por encima de un umbral previamente establecido. Este enfoque puede considerarse práctico si el sistema de TA generara traducciones para cuya posedición se requiera menos tiempo que para traducir esos segmentos desde cero o para corregir la coincidencia encontrada en la MT por su bajo porcentaje de coincidencia.

Aparte del uso de TA ya comentado, existen también formas más sofisticadas de integrar MT y TA. En la literatura, pueden encontrarse dos tipos de enfoques que integran MT y TA: (1) enfoques que integran sub-segmentos de la MT en el proceso de decodificación de un sistema de TA y (2) enfoques que utilizan el segmento destino t de una UT (s, t) como plantilla o esqueleto de la traducción a realizar. Los dos enfoques suelen estar respaldados por un sistema de TA que podrá ser de cualquier tipo: basado en reglas, estadístico, o neuronal, por citar algunos. Los enfoques en el segundo grupo de integración MT-TA, es decir, aquellos que usan el segmento destino t en una UT (s, t) como esqueleto de la traducción que se va a producir, están más cerca del enfoque introducido en esta tesis, el cual hace uso de TA para reparar las coincidencias parciales encontradas en la MT, modificando únicamente los sub-segmentos no coincidentes.

Reparación de concordancias parciales

En el Capitulo 2, se presenta nuestra aproximación al problema de reparación de concordancias parciales (RCP). Nuestro enfoque consiste en reemplazar solo los sub-segmentos no concordantes en t de la MT y, al mismo tiempo, guarda los sub-segmentos en t que se puedan reutilizar, ya que han sido traducidos profesionalmente. Primero, el método alinea las palabras en el segmento origen a traducir s' con la UT (s, t). Tras esto, se identifican las palabras no concordantes en s es decir, los sub-segmentos que no tienen en común. A continuación, el método utiliza un sistema de TA, aunque podría usar cualquier otra fuente de información bilingüe, para identificar los sub-segmentos en t que son traducciones de los sub-segmentos no concordantes en s, de una manera similar a la de Esplá-Gomis et al. (2011), y construye un conjunto de *operadores de reparación* traduciendo los sub-segmentos no concordantes en s'. Cada operador de reparación especifica el sub-segmento en lengua meta τ en t que necesita ser reparado y el sub-segmento en lengua meta τ' que se utilizará para la reparación. A continuación, se aplican combinaciones de operadores de reparación compatibles para obtener un conjunto de segmentos reparados, de los cuales se selecciona el que se vaya a utilizar finalmente tras estimar su calidad.

Ya se ha explicado cómo se construyen los operadores de reparación; para ilustrar mejor los pasos de alto nivel, profundizaremos en la implementación del algoritmo de reparación de concordancias parciales. La implementación del algoritmo RCP (Sección 2.2) se divide en dos partes: (1) la generación de operadores de reparación de concordancias parciales y (2) la exploración de posibles combinaciones de operadores de reparación para generar el conjunto de segmentos reparados denominados concordancias parciales reparadas.

El método para generar los operadores mencionados está explicado en detalle en la Sección 2.2.1. Para obtener el conjunto de operadores de reparación que será utilizado, primero, se obtiene el alineamiento entre las palabras de s' y las de s como un subproducto del cálculo de la distancia de edición a nivel de palabra (Wagner and Fischer, 1974) entre s' y s. Los pares de sub-segmentos (σ, σ'), que contienen palabras no alineadas (no concordantes) y sus posiciones correspondientes en s y s', se obtienen utilizando el algoritmo de extracción de pares de sub-segmentos utilizado en TA estadística basada en sub-segmentos bilingües (Koehn, 2010, sección 5.2.3). Estos pares de sub-segmentos se traducen luego a la lengua meta para obtener los conjuntos M y M' de traducciones de sub-segmentos μ y μ', respectivamente, utilizando un sistema de TA como caja negra. Finalmente, estas traducciones se utilizan para construir operadores de reparación buscando todas las ocurrencias en t de cada sub-segmento meta μ para obtener los sub-segmentos meta τ posicionados en t y luego asociando a cada τ el sub-segmento meta μ' para obtener τ', el sub-segmento que se utilizan para la reparación. Si μ no se encuentra en t, no se podrá construir ningún operador de reparación. Esto actúa como un control de calidad que evita que el algoritmo genere operadores de reparación de baja calidad. El ejemplo de la Figura 2.1 ilustra cómo

se construye la lista de operadores de reparación. Cabe destacar que solo en aquellos casos en los que μ, la traducción de σ, se encuentra como un sub-segmento contiguo de palabras en el segmento meta t de la UT que se este reparando, se podrá construir un operador de reparación; esto se indica en la quinta columna de la tabla en la Figura 2.1.

El método para la exploración de posibles combinaciones de operadores de reparación para generar el conjunto de segmentos reparados se presenta en la Sección 2.2.2. Los segmentos reparados se crean utilizando la lista de operadores de reparación (P en el Algoritmo 2 de la Sección 2.2.2). Primero, se combinan los operadores con una búsqueda exhaustiva en profundidad del árbol de recursividad y luego se construyen concordancias parciales reparadas $t^{\simeq}$.

Para que un operador de reparación sea aplicable, deberá ser compatible con el conjunto de operadores de reparación O aplicado hasta el momento para construir $t^{\simeq}$ (en la Sección 2.3). Si es compatible con el resto de operadores de reparación en O, el operador de reparación se agregará a O y se aplica; de lo contrario, la rama del árbol de recursividad se corta. Cuando se alcanza una hoja del árbol de recursividad, la concordancia parcial reparada correspondiente $t^{\simeq}$ se agrega a la lista T de concordancias parciales reparadas.

Para evaluar el potencial de nuestro método de reparación, realizamos una evaluación de tipo oráculo (explicada en la Sección 2.4.3) con tres pares de idiomas: inglés–español (en–es), español–portugués (es–pt) y español–francés (es–fr). Elegimos estos idiomas para estudiar cómo se comporta nuestro método con idiomas estrechamente relacionados (español–portugués y español–francés) y no tan relacionados (inglés–español).

Nuestros experimentos utilizan varios sistemas de TA. Antes de comprobar el rendimiento de nuestro método, evaluamos los sistemas TA de forma aislada. Para esta evaluación se usaron tres sistemas de TA distintos: Apertium (basado en reglas; (Forcada et al., 2011)), Moses (estadístico basado en segmentos bilingües; (Koehn et al., 2007)) y Nematus (neuronal, (Sennrich et al., 2017)). Los resultados de esta evaluación se pueden encontrar en la Sección 2.4.2.

Después de evaluar los sistemas de TA de forma aislada, evaluamos nuestro método RCP con los pares de idiomas mencionados y distintos umbrales de coincidencia: 60 %, 70 % y 80 % , en consonancia con los hallazgos de Bowker (2002), quien demuestra que los umbrales de coincidencia usados suelen ser superiores al 60 %. En estos experimentos, probamos con varios métodos de traducción: MT, TA y RCP. Los resultados de las Tablas 2.5 y 2.6, muestran que con nuestro método de RCP se puede lograr una tasa de error por debajo de la de usar un sistema de TA de forma aislada. Además, concluimos que para el par de idiomas no relacionados (en–es), un sistema RCP respaldado por un sistema TA estadística o neuronal supera al sistema respaldado por un sistema de TA basado en reglas para todos los umbrales de coincidencia.

La diferencia entre un segmento reparado por nuestro método (eligiendo el oráculo) y las traducciones producidas por un sistema TA revela que nuestro método es bastante robusto a los errores típicos de TA. Esto se debe a que para que un operador de reparación se construya con éxito, τ, la traducción del sub-segmento σ de s, debe aparecer como un sub-segmento de texto contiguo en t, la propuesta de traducción a reparar. Esto actúa como un filtro de calidad que hace que nuestro método no se utilice para reparar sub-segmentos para los cuales la fuente de información bilingüe no coincide con la MT. Obviamente, esta verificación de calidad no se puede realizar en τ', la traducción del sub-segmento σ' en s' alineada con σ. Sin embargo, parece que tener un control de calidad en τ ayuda a garantizar que la mayoría de los operadores de reparación construidos sean de buena calidad.

Estimación de la calidad de los segmentos reparados

En el contexto de la TA, los métodos de estimación de la calidad (EC) a nivel frase (Blatz et al., 2004; Specia et al., 2009) han sido desarrollados durante las últimas dos décadas para evitar traducciones de baja calidad y para elegir entre un conjunto de traducciones diferentes producidas por distintos sistemas de TA para un segmento origen determinado, o para estimar el esfuerzo de poseditar la salida de un sistema TA determinado. La calidad generalmente se mide en términos de tiempo de posedición, en términos de cantidad de operaciones de edición necesarias para convertir la traducción en una traducción adecuada, o usando otras métricas relacionadas (Specia, 2011; Bojar et al., 2014).

Las técnicas de EC en TA se pueden adaptar fácilmente para estimar la calidad de las diferentes concordancias parciales reparadas que se obtienen como resultado de aplicar el método descrito en el Capítulo 2. Existen principalmente dos enfoques diferentes para lograr esto: el uso de clasificadores binarios y por otro lado, el uso de un regresor. El primero se puede utilizar para seleccionar la mejor concordancia parcial reparada mediante una comparación por pares (Avramidis, 2013); el segundo se puede utilizar para ordenar el conjunto de segmentos reparados en función de su calidad estimada. Nuestro método, descrito en el Cápitulo 3 sigue este segundo enfoque; específicamente, utilizamos, después de experimentos preliminares con regresores lineales y vectores de soporte (Basak et al., 2007), árboles extremadamente aleatorios (Geurts et al., 2006) para la regresión (*extremely-randomized trees* en inglés).

Nuestro método de estimación de la calidad utiliza en primer lugar el enfoque de reparación descrito en el Capítulo 2 con el que genera un conjunto de concordancias parciales reparadas. Después, estima la calidad de cada concordancia parcial reparada para seleccionar la mejor. El método para estimar la calidad de estos segmentos, inspirado en el trabajo sobre estimación de calidad (EC) en TA a nivel frase (Blatz et al., 2004; Specia et al., 2009), utiliza regresores entrenados usando dos conjuntos de características: uno que usa información disponible para las herramientas TAO (características de *caja negra*, Sección 3.2.1), y otro que explota la información del funcionamiento interno

del algoritmo de reparación descrito en el Capítulo 2 (características de *caja blanca*, Sección 3.2.2).

Para demostrar que nuestro método de EC da buenos resultados, tomamos como punto de partida la evaluación realizada en el Capítulo 2 originalmente basada en el uso de un oráculo usando varios umbrales de coincidencia y con tres pares de idiomas diferentes. Con el fin de determinar la mejor configuración para entrenar el regresor para estimar la calidad de las concordancias parciales reparadas, hemos probado con las siguientes configuraciones: (1) entrenar diferentes regresores para diferentes umbrales de coincidencia y pares de idiomas, (2) entrenar un regresor para cada par de idiomas independientemente del umbral de coincidencia, y (3) entrenar un solo regresor para todos los pares de idiomas y umbrales de coincidencia. Los resultados obtenidos se discuten en la Sección 3.4 (Tablas 3.4 a 3.6) donde se proporcionan resultados con corpus filtrados siguiendo el método de Esplà-Gomis et al. (2015) para eliminar UT con ruido y con corpus sin filtrar. Los resultados de la Tabla 3.6 son bastante similares a los de la Tabla 3.5 donde se usa un regresor diferente por par de idiomas. Para algunos pares de idiomas y umbrales de coincidencia, los resultados mejoran ligeramente, mientras que para otros empeoran. Esto nos permite concluir que, dada la pequeña diferencia en las tasas de éxito informadas en ambas tablas, es aconsejable utilizar un solo regresor entrenado usando un umbral de coincidencia del 60 % para todos los idiomas juntos, al menos para los idiomas que están relacionados en algún sentido (como las cuatro lenguas indoeuropeas occidentales de nuestros experimentos).

Los resultados muestran además que el conjunto de características que proponemos es lo suficientemente informativo como para obtener regresores que permitan seleccionar concordancias parciales reparadas cercanas a la mejor (oráculo) de entre las producidas por nuestro algoritmo de reparación y funciona mejor en los corpus filtrados donde se eliminan las UT *ruidosas*. La diferencia de rendimiento es notable para los umbrales de coincidencia por debajo del 90 %, especialmente en el caso del inglés–español. Además, la tasa de éxito, es decir, la proporción de operaciones de edición evitadas al editar la concordancia parcial reparada seleccionada sobre el número de operaciones de ediciones evitadas al editar la mejor concordancia parcial reparada posible, aumenta a medida que crece el umbral de coincidencia; para el umbral de coincidencia del 60 % las tasas de éxito en los corpus filtrados rondan el 0,68, mientras que para el 90 % alcanzan hasta 0,78; superando el 0,80 para inglés–español y español–portugués. Esto significa que el uso del segmento reparado seleccionado permitirá evitar de media, más del 80 % de las operaciones de edición que se habrían evitado si se hubiera elegido la mejor concordancia parcial reparada posible (oráculo). Si las concordancias parciales reparadas a usar se seleccionaran al azar, el ahorro de operaciones de edición estarían alrededor del 44 %. Nuestro método de EC es mejor a la hora de clasificar las concordancias parciales reparadas cuando la cantidad de sub-segmentos no similares es pequeña; el cual es el escenario típico en el que se usan programas TAO basados en MT. [1]

[1]Las concordancias parciales rara vez se usan para un umbrales de coincidencia por debajo del 70 %.

En cuanto a las características utilizadas por los regresores, hemos propuesto un conjunto de características compuestos por características de caja negra (independientes del sistema) y de caja blanca (dependientes del sistema). Las características de caja negra son más fáciles de calcular ya que no explotan ninguna información sobre los operadores de reparación utilizados para generar los segmentos reparados. Las características de caja negra resultan más informativas que las de caja blanca, aunque todas son útiles hasta cierto punto. Este resultado sugiere que el método de EC utilizado para seleccionar el mejor segmento reparado podría usarse para clasificar los segmentos reparados producidos por otros enfoques de RCP como los descritos en la Sección 1.4.

Los concordancias parciales reparadas seleccionadas por nuestro método están consistentemente más cerca de las traducciones de referencia que los segmentos de destino no reparados (t). También, son mejores que las traducciones obtenidas al traducir segmentos de origen completos (s') utilizando el sistema de TA. Además, los mejores regresores se obtienen en la mayoría de las veces cuando se entrenan con todos los pares de idiomas juntos, lo que significa que los valores de las características propuestas y los regresores aprendidos son independientes del idioma.

Selección del traductor automático a emplear para la reparación de concordancias parciales

Para aumentar aún más el rendimiento de nuestro sistema, intentamos ahorrar recursos y tiempo al traductor mediante la implementación de un sistema que selecciona, para cada sub-segmento, el sistema de TA a utilizar para la reparación de concordancias parciales, sin ejecutar realmente el sistema de TA. El Capítulo 4, se centra específicamente en un problema que podría ocurrir al usar varios sistemas de TA: saber seleccionar el sistema de TA más adecuado dado solo el segmento en lengua origen a traducir.

Nuestra investigación se centra primero en la TA para demostrar la importancia de tener un método que seleccione un sistema de TA antes de traducir un segmento en lengua origen. Tras demostrar que se puede seleccionar el sistema de TA con una precisión relativamente alta (casi el 70 %), se ilustra cómo la selección de un sistema de TA de entre un conjunto de varios sistemas de TA mejora los resultados presentados en el Capítulo 2. La motivación para la investigación se basa en trabajos recientes (Bojar et al., 2017) que demuestran que un sistema de TA neuronal logra mejores resultados en términos generales; pero, un sistema de TA estadística o basado en reglas pueden obtener mejores resultados para segmentos individuales dependiendo del dominio y tipo de cada segmento. Existen también investigaciones que demuestran que los traductores profesionales y los usuarios de herramientas TAO prefieren un sistema de TA estadística sobre un sistema de TA basado en reglas o un sistema de TA neuronal.(Arenas, 2013) Un trabajo anterior de Bentivogli et al. (2016) demostró cómo funcionan los diferentes

paradigmas de TA ante ciertos tipos de entrada. Más específicamente, se ha encontrado que un sistema de TA neuronal tiene un rendimiento peor que un sistema de TA estadística debido a anomalías específicas en el texto como signos de puntuación o nombres de entidades (Koehn and Knowles, 2017). Teniendo esto en cuenta, podría considerarse prematuro abandonar por completo los métodos "antiguos" en favor de los "nuevos". Las alternativas que combinan sistemas de TA podrían lograr mejores resultados; particularmente, si el método de combinación de sistemas puede aprovechar las diferentes fortalezas de cada paradigma.

En el Capítulo 4 se presenta el sistema predictivo, llamado SelecT, que tiene como objetivo mejorar las traducciones obtenidas eligiendo el sistema de TA para cada segmento origen antes de intentar traducirlo. También se presentan los resultados de dos experimentos para nuestro sistema de selección: el primer experimento mide la precisión de los clasificadores de SelecT usando BLEU y la tasa de error por palabra (WER) como métricas de evaluación de un sistema TA; el segundo experimento utiliza SelecT como predictor para elegir un sistema TA para RCP. En concreto, hemos probado con tres tipos distintos de clasificadores: uno basado en redes neuronales recurrentes bidireccionales (Schuster and Paliwal, 1997), otro basado en FastText (Joulin et al., 2017), y, por último, otro basado en regresión logística (Cramer, 2002; Fan et al., 2008).

Los resultados cuando se usa SelecT para elegir un sistema de TA (sin evaluar el método RCP) se encuentran en la Sección 4.6.1. Los resultados del primer experimento se muestran en la Tabla 4.1. A cada segmento se le asigna una etiqueta que se corresponde con el sistema de TA con la puntuación BLEU más alta. El trabajo del clasificador durante las pruebas es predecir el sistema de TA con mejor rendimiento para cada segmento. El mejor clasificador es FastText. Tiene una precisión del 68,12 % y supera ligeramente (menos del 1 %) al clasificador que usa redes neuronales recurrentes bidireccionales. En la Tabla 4.2, se ofrece más información del primer experimento que nos muestra: (1) las cotas superior e inferior proporcionadas por el mejor y el peor sistema de TA, respectivamente; (2) el rendimiento usando cada tipo de clasificador; y, (3) el rendimiento de cada sistema cuando se usa de forma aislada. La media entre las cotas superior e inferior ofrece un valor de referencia que nuestro sistema mejora y viene a demostrar que nuestro sistema tiene éxito a la hora de predecir el mejor sistema en la mayoría de los casos. La selección aleatoria también podría utilizarse como referencia, aunque, en nuestros experimentos, la selección aleatoria no proporcionó resultados muchos mejores que la cota inferior (alrededor de 2 puntos de BLEU).

La Tabla 4.4 muestra el rendimiento de nuestro método RCP para tres umbrales diferentes de coincidencia — 60 %, 70 % y 80 % — en términos de tasa de error por palabra (WER en la tabla) para los tres sistemas de TA (Apertium, Moses y Nematus) y para SelecT cuando usa el clasificador basado en FastText (el clasificador con mayor precisión). La última columna (SelecT) muestra que SelecT ofrece mejores resultados cuando se usa para seleccionar el sistema de TA de forma aislada (la sub-columna MT en la tabla), seguido de cuando se usa para RCP (la sub-columna FMR en la tabla). El sistema SelecT supera los resultados del Capítulo 2 para todos los sistemas en

ambas situaciones de concordancias parciales (concordantes o no). Esto es más evidente cuando el umbral de coincidencia es menor. Por ejemplo, para el umbral de coincidencia del 60 %, SelecT es aproximadamente 1 punto mejor que el sistema de TA con mejor puntuación (Moses) y 1,3 puntos mejor que el mejor sistema de TA para RCP (Moses también). Por último, SelecT parece funcionar mejor cuando se traducen segmentos completos que cuando se traducen sub-segmentos para la construcción de operadores de reparación. Este comportamiento se atribuye al entrenamiento del clasificador sobre segmentos en lugar de sub-segmentos.

Nuestros experimentos muestran que SelecT cubre más casos que cualquier sistema de TA probado y, por lo tanto, podría ser útil para un traductor o usuario de herramientas TAO. SelecT es agnóstico con respecto a los sistemas de TA que se utilizan y no requiere cambios en proceso subyacentes.

Combinación con un sistema de posedición automática

En las secciones anteriores nos hemos centrado en el rendimiento de nuestro método de reparación de concordancias parciales (RCP) de forma aislada como un elemento modular del entorno TAO. Los usuarios de herramientas TAO pueden utilizar RCP con una MT y un sistema de TA, pero, todavía existe otra opción disponible en el entorno TAO para mejorar su productividad – la posedición automática (PA). En el Capítulo 5 mostramos como se ha combinado nuestro método con PA: primero, se produce una concordancia parcial reparada a partir de la unidad de traducción y el segmento origen; luego, el segmento reparado se mejora aún más mediante un sistema de PA ajustado específicamente para dicha tarea. Los experimentos realizados sobre la traducción de textos del inglés al alemán muestran que, al combinar las dos tecnologías, la calidad de las traducciones mejora hasta un 23 % en comparación con un sistema de TA usado de forma aislada. Además, la mejora, con respecto a un sistema RCP aislado, es del 16 %. Esta mejora demuestra la eficacia de nuestra solución conjunta. En la Sección 5.2 revisamos el sistema de PA de última generación utilizado en nuestros experimentos y después, en la Sección 5.3, mostramos cómo las dos tecnologías se combinan para formar un nuevo sistema que se agrega de forma modular a un sistema TAO. Los resultados de la combinación de RCP y PA se presentan en la Sección 5.5 junto con una evaluación humana para confirmar los resultados obtenidos usando medidas automáticas de evaluación de la calidad de las traducciones.

Los métodos de PA se han introducido en las herramientas TAO como una técnica de posedición para corregir segmentos traducidos automáticamente. Las investigaciones han demostrado que los traductores pueden ser más productivos cuando utilizan técnicas de posedición de última generación (Isabel, 2017). Según lo motivado por Parton et al. (2012), un sistema de PA puede ayudar a mejorar un sistema de TA de dos maneras: (1) explotando información no disponible durante la traducción o (2)

realizando un análisis del texto más profundo que el que pueda hacer el decodificador típico de un sistema de TA. Además, el sistema de PA puede adaptar la salida de un sistema de TA de propósito general al estilo de escritura de un dominio específico. La PA puede proporcionar a los traductores una traducción automática mejorada para reducir así el esfuerzo (humano) posterior de posedición. Los sistemas de PA no se basan en memorias de traducción y son eficaces sin la intervención inicial del traductor. No obstante, en la Sección 5.3, nuestros experimentos muestran que un sistema de PA puede integrarse sin problemas en el flujo de traducción típico para mejorar las concordancias parciales reparadas. Primero, se utiliza RCP para producir una propuesta de traducción reparada y luego se usa PA como una herramienta para mejorar la calidad de dicha propuesta. El sistema combinado está ilustrado en la Figura 5.1.

Los resultados de nuestros experimentos se presentan en la Sección 5.5 y muestran que la combinación de RCP y PA mejora los resultados obtenidos en el Capítulo 2. Usamos la mejor concordancia parcial reparada (oráculo) en lugar de una obtenida usando la estimación de la calidad como se hizo en el Capítulo 3. Para demostrar el rendimiento de nuestra combinación, presentamos los resultados en la Tabla 5.2, la cual muestra los resultados de tres tipos de experimentos usando como medidas de evaluación BLEU y WER: (1) TA y MT; (2) TA con PA; y, (3) RCP y RCP con PA. Aunque los resultados de la MT son mejores que los experimentos del Capítulo 2, RCP y RCP con PA logran superarlo. La combinación de RCP con PA mejora la calidad de traducción del sistema RCP aislado por un amplio margen (+12 puntos de BLEU). En todos los experimentos, la adición del sistema de PA ayuda a lograr mejores resultados. En un sistema en producción, lo ideal sería que los sistemas de TA y PA estuvieran debidamente entrenados sobre corpus del dominio deseado para obtener el máximo beneficio de la combinación de los dos métodos. Nuestros resultados excluyen otras combinaciones de sistemas, como el uso de corpus del dominio deseado o técnicas de estimación de calidad, con el fin de mostrar únicamente la combinación de RCP y PA bajo configuraciones genéricas (listas para usar).

Las dos métricas de evaluación (BLEU y WER) muestran como nuestra combinación de sistemas mejora los resultados obtenidos hasta ahora y probablemente sería suficientes para demostrar que merece la pena combinar RCP con PA. Sin embargo, como un control cualitativo adicional, verificamos las traducciones de nuestros sistemas con mejor rendimiento con un evaluador nativo de alemán. Se midió la fluidez y coherencia de las traducciones del sistema dadas las frases de origen donde se preguntó al evaluador nativo: "¿Es la traducción comprensible y una traducción válida dado la frase en lengua origen?". La Tabla 5.3 muestra una descripción de cómo se puntuaron nuestros mejores sistemas en una escala Likert (Likert, 1932): la puntuación de la evaluación humana está en línea con las métricas automáticas ya discutidas.

Discusión

El enfoque principal de esta tesis es la introducción de un algoritmo de reparación de concordancias parciales para su uso en herramientas de traducción asistida por ordenador, capaz de usar cualquier fuente disponible de información bilingüe (usamos sistemas de traducción automática en nuestros experimentos) como una caja negra; es decir, sin acceso a su funcionamiento interno. Con la esperanza de que mediante la aplicación de las diferentes técnicas descritas en esta tesis, la intervención humana, desde el punto de vista de la posedición, pueda mantenerse al mínimo y al mismo tiempo preservar la calidad de la traducción. No solo presentamos un algoritmo de última generación para la reparación de concordancias parciales si no que presentamos varias mejoras que pueden considerarse contribuciones importantes al estado de la cuestión. Para ser más específicos, a continuación indicamos los componentes principales en los que se centra esta tesis:

- un novedoso algoritmo de reparación de concordancias parciales capaz de utilizar cualquier fuente de información bilingüe como una caja negra para proponer segmentos reparados basados en memorias de traducción y que modela todas las operaciones de edición (inserciones, borrados y sustituciones) de la misma manera;

- el diseño de un conjunto de características independientes del idioma para la estimación de la calidad que se puede utilizar para seleccionar el mejor segmento reparado por nuestro método en un entorno multilingüe independiente del sistema de reparación de concordancias parciales o fuente de información bilingüe utilizada;

- la combinación de reparación de concordancias parciales con posedición automática para reducir aún más la posedición necesaria para las propuestas de una manera fluida;

- el diseño de características de caja negra, es decir, sin acceso al funcionamiento interno del algoritmo de reparación de concordancias parciales, que permite que nuestro enfoque de estimación de la calidad se pueda utilizar para evaluar la calidad de segmentos reparados producidos utilizando otros enfoques;

- un clasificador que es capaz de seleccionar, a priori, el sistema de TA que se utilizará para la reparación de concordancias parciales para cada segmento.

Con respecto al algoritmo de reparación de concordancias parciales, esta tesis presenta y prueba empíricamente un nuevo enfoque que está diseñado para requerir solo la presencia de una memoria de traducción y una fuente de información bilingüe y, al mismo tiempo, es altamente efectivo en un entorno de herramienta TAO. Otras ventajas principales incluyen: independencia del idioma de origen y de destino y fácil integración

en cualquier entorno de herramientas TAO. En el Capítulo 2, los experimentos se presentan con tres pares de idiomas (inglés–español, español–portugués y español–francés) que muestran el correcto funcionamiento de nuestro algoritmo. Se utilizan tres paradigmas principales de traducción automática (basado en reglas, estadístico y neuronal) para probar el potencial del algoritmo. Los hallazgos muestran que nuestro enfoque es más efectivo que usar un sistema de traducción automática solo y puede usarse con software moderno basado en memorias de traducción como OmegaT.[2]

Otra técnica innovadora que saca a la luz esta tesis es una técnica de estimación de la calidad para seleccionar segmentos reparados por nuestro método. Demostramos que un regresor no lineal puede estimar con precisión, dadas varias características a nivel de segmento, la mejor propuesta reparada de entre muchas. Este enfoque novedoso es una contribución importante, además del algoritmo para la reparación de concordancias parciales, porque también se puede utilizar con otros métodos de reparación. Además, la técnica de estimación de calidad, que funciona mejor con árboles extremadamente aleatorios, es muy eficaz como regresor para varios pares de idiomas.

La modularidad de nuestra solución de reparación es una característica importante. No se trata únicamente de que nuestro sistema se pueda utilizar con cualquier fuente de información bilingüe; también, se puede utilizar en combinación con otros sistemas que normalmente existen en un entorno de herramientas TAO. En esta tesis mostramos lo fácil que es combinar nuestro sistema con un sistema de posedición automática de última generación. Se dejan otras combinaciones para trabajos futuros; pero la facilidad de integración con otras herramientas TAO que muestra nuestro enfoque es una gran ventaja.

En lo que respecta al sistema predictivo la adición de SelecT a esta tesis proporciona evidencia de que la naturaleza de caja negra de nuestro enfoque de reparación de concordancias parciales puede beneficiarse de otros sistemas de manera agnóstica. El clasificador se puede utilizar junto con el sistema de TA que usa el algoritmo de reparación de concordancias parciales para mejorar aún más todo el proceso.

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

Preface

When translating a source text into a target text, translators may use what is known as a computer-aided translation (CAT) tool to increase their productivity. A CAT tool has several functions; one of those functions is the use of what is known as a translation memory (TM). A TM stores parallel sentences, or *segments*, for two languages. Each pair (source, target) of parallel segments is called a translation unit (TU). For a new segment to be translated, CAT tools are responsible for finding TUs from the TM where the TU's source segment is similar to the source segment to translate. When a CAT tool does not find an exact match, the translator has to choose the TU whose source segment is most similar to the segment to be translated and use its target segment as a starting proposal for the translation. Translators use different techniques to assist in the editing of these target segments. In this dissertation, we explore the use of a technique called *fuzzy-match repair* (FMR) which repairs translation proposals from a TM by using any available machine translation (MT) system, or any external source of bilingual information, regardless of its internals. FMR aids CAT-tool users through a process that "repairs" the target segment of a TU by creating a new proposal with an improved translation. The results presented in this dissertation show how FMR improves the translation proposals and reduces the amount of editions needed for translating those source segments for which there is not an exact match in the TM.

My work with FMR spans over several years dating back to October 2013 when I first began my doctorate studies at the Departament de Llenguatges i Sistemes Informatics at Universitat d'Alacant in Alicante, Spain. The initial proposal was to solve a problem that had not been researched at the time – how to repair translation proposals in a CAT-tool environment using a black-box MT system, i.e. without access to its internal workings. I conducted an initial pilot study which lasted about four months. The first pilot study implementation resulted in an article that was presented at the 2014 conference of the Association for Machine Translation in the Americas (AMTA). It consisted of generating all possible fuzzy-match repaired segments and conducting an oracle evaluation to see if the algorithm was capable of producing fuzzy-match repair segments good enough to reduce their post-editing needs. The initial empirical proof was provided through word-error rate measurements that the FMR approach, called *patching* originally, worked for repairing TM proposals in the English–Spanish language pair. The pilot study was the ground work for what resulted in deeper experimentation of how FMR could improve performance in several situations. In the

two years following (2014 and 2015), several approaches were considered to improve FMR which led way to a re-implementation that decreased the number of overlapping repair proposals through the introduction of compatibility checks between the repair operators to avoid generating meaningless proposals.

Apart from the introduction of compatibility checks, new language pairs were introduced including Spanish–French (es–fr) and Spanish–Portuguese (es–pt). The introduction of compatibility checks, new language pairs, and a new implementation led to another article in 2016 at AMTA which described the FMR algorithm in more detail and provided a more complete baseline for FMR, able to be integrated with other systems for further improving translator productivity.

After establishing the baseline in 2016, I began to discuss my work with others from the academic world like Philip Koehn and Rebecca Knowles from John Hopkins University. With the resurgence of neural machine translation (NMT), I began to try out FMR with other MT systems since my method was designed to work as a black box – the original MT system used was Apertium,[3] a free/open-source rule-based MT system whose development is headed by Mikel Forcada at the Departament de Llenguatges i Sistemes Informatics at Universitat d'Alacant. The thought of bringing NMT and statistical machine translation (SMT) systems to test in FMR was intriguing as I had already used Moses, an SMT system, for low-resource language translation of a Peruvian language called Quechua during my Master's thesis in Computer Science at Hofstra University. Armed with the gumption of working with NMT and SMT on FMR, I discussed the idea with Phillip Koehn and Rebecca Knowles. Several tests were done, similar to the 2014 paper at AMTA, in English to Spanish using different types of NMT and SMT systems. The experiments showed that the quality of the MT system and the domain of the text used to train the MT system were important. The article where those experiments can be found was presented in 2018 at AMTA. At that same conference, work from my Master's thesis on low-resource languages was accepted into the Workshop on Technologies for MT of Low Resource Languages (LoResMT 2018) hosted at AMTA 2018. Having two papers accepted at the same location was a new and gratifying experience for me.

During the past three years of my dissertation, I dedicated myself to expanding the FMR system in different ways. I did a research stay from January to April, 2018, at New York University (NYU) hosted by Adam L. Meyers, clinical professor of Natural Language Processing. During my stay at NYU, I worked with several other researchers, one of which became a researcher mentor on neural networks and statistical approaches, Kyunghyun Cho, known for publishing work on the gated recurrent unit for recurrent neural networks. Kyunghyun Cho, Weiyi Lu, and I created a system, known as SelecT, which served as part of this dissertation (Chapter 4). My stay at NYU also resulted in a deeper, academic, dive into natural language processing (NLP) and its approaches. In one project, I worked with two other researchers, Sam Bowman and Katharina

[3] `https://apertium.org`

Kann, on measuring the fluency of MT using a logarithmic approach instead of the most-common MT measurement, BLEU. In another project, with Samir Undavia and Adam L. Meyers, I worked on a method for classifying supreme court cases using NLP. Lastly, at NYU, I also helped with the implementation of a terminology extraction software, called the Termolator,[4] in collaboration with Adam L. Meyers and others.

The last three years of my dissertation provided the extra insight needed for producing what can be considered one of the most important components of our FMR approach, the use of quality estimation to determine which proposals were the best to return to the user. Our quality estimation approach, based on extremely randomized trees, is able to estimate the post-editing effort of each fuzzy-match repaired segment produced. Its introduction in this dissertation was well received and I enjoyed the implementation. For our work, we were rewarded with an article in one of the highest-indexed, most important, journals in the Computer Science sector, namely *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Other collaborations with research groups includes a collaboration with Marco Turchi and Matteo Negri at the Fondazione Brunno Kessler in Trento, Italy. Marco and Matteo were responsible for publishing several works on automatic post-editing (APE) previous to our collaboration. Since they were obtaining results in CAT tools similar to those of FMR, it made sense to attempt to combine the two approaches. The combination of FMR with APE is presented in this dissertation, Chapter 5.

One should note that the time to complete this dissertation was nearly seven years. This was mostly due to the holding of a full-time position in the private sector. For the first 5.5 years (October 2013 - March 2019) of my PhD work, I worked in the health care sector at Nuance Communications where I worked together with several highly-skilled researchers from IBM and others in a project which was dedicated to using NLP to classify health-care records in several ways. During the last 1.5 year (March 2019 - Present), I have also worked as the head of a small NLP group for Blackboard Insurance (an AIG company) solving problems related to underwriter activity and insurance claim fraud. The combination of having worked in private industry along with the research of this dissertation have proven to be a truly valuable experience, providing insight into how NLP products are implemented and working at the global level in the health care and insurance industries.

As an aside, and not related to FMR, my previous work, and background research passion has been working on low-resource MT. I have collaborated with several others to work on indigenous South American languages in the past, including during my Master's thesis which was on a technique for improving MT for Quechua. Since then, I have worked on other approaches and languages including an even lower resource language from Peru and Brazil called Asháninka. Techniques have been introduced on sub-segmentation for MT along with neural machine translation by learning the morphology of a low-resource language through the use of a high-resource language.

[4] `https://nlp.cs.nyu.edu/termolator/`

My plan is to continue to work with both FMR and low-resource MT with the hope to one day combine FMR with low-resource MT as a future work.

Lastly, my advisor and co-author of several works, Felipe Sánchez-Martínez, has been an important resource for understanding several main points. He has helped me focus on the approaches presented and given unparalleled and deep research-associated advice. His constant supervision has given this dissertation a fresh outlook on a problem that included machine learning, a topic of which I thoroughly enjoyed. Mikel L. Forcada from the Departament de Llenguatges i Sistemes Informatics at Universitat d'Alacant helped greatly to establish the FMR idea as a pertinent and valid idea for research. He has also helped direct my research in the right direction. Both have greatly contributed to the success of FMR.

Structure of this thesis

This thesis is structured in 6 chapters. The following is a brief synopsis of each one.

Chapter 1 begins by giving an introduction to computer-aided translation (CAT) tools based on translation memories (TM). It presents the most common fuzzy-matching algorithm used in CAT tools to match a source segment to be translated with a source segment in the TM. A baseline translator's workflow is outlined which explains how a CAT tool is used. After that, an introduction to machine translation (MT) is presented which leads way to a mixed translation workflow that combines TMs with MT and provides the necessary information to better understand the final section which is dedicated to fuzzy-match repair.

Chapter 2 provides details on the fuzzy-match repair (FMR) approach, the centerpiece of this dissertation. The FMR approach is described in detail along with the two core algorithms it uses. First, an algorithm that uses MT for generating repair operators. Second, an algorithm that uses the repair operators to generate as many fuzzy-match repaired segments as possible. Experiments are done that illustrate how the three main MT paradigms (rule-based, statistical, and neural) performed when used as sources of bilingual information for FMR. The results presented in this chapter serve as a first step for the FMR approach presented and use an oracle to measure FMR's performance, a technique that is later replaced by a quality estimation method.

Chapter 3 dives deeper into the FMR algorithm in order to show how FMR segments can be chosen by using a tree-based regressor for estimating their quality. An evaluation is presented on the three language pairs from Chapter 2 to provide proof that the estimation of the "best" fuzzy-match repaired segment is consistently nearer to the oracle than using a TM or MT system alone.

Chapter 4 focuses on how to decide which black-box MT system to use for FMR in real time. A mechanism is presented that automatically selects an MT system of

those available to the CAT tool. It is a state-of-the-art selector which is able to determine the best MT system to use at the segment level *a-prior* without having to translate them. The chapter includes several experiments based on BLEU and shows that the overall aim of the selection method, to save the translator time by selecting at the segment-level which MT system to use, is accomplished.

Chapter 5 explores the integration of FMR and automatic post-editing (APE). In particular, the use of APE is presented as an orthogonal mechanism to work with FMR. Background is given on what APE is and how it works within a CAT tool. Then, the combination of FMR with APE is presented. Finally, experiments are done with the combination system for translations from English to German that show how well FMR with APE perform when compared to a stand-alone MT system in FMR.

Chapter 6 recapitulates the work performed in this dissertation. It summarizes the approaches from each of the chapters by giving a high-level overview of the contributions of this dissertation to the state of the art. It also provides possible future research lines on the FMR approach presented in this dissertation.

Publications

Some parts of this thesis have been published in journals, conference and workshop proceedings. Here is a list of papers in chronological order (in brackets, the chapter to which each paper is related):

- John E. Ortega, Felipe Sánchez-Martınez, and Mikel L. Forcada. (2014). Using any machine translation source for fuzzy-match repair in a computer-aided translation setting. In *Proceedings of the 11th Biennial Conference of the Association for Machine Translation in the Americas*, Volume 1, pp. 42–53, October 2014, Vancouver, Canada. [**Chapter 2**]

- John E. Ortega, Felipe Sánchez-Martınez, and Mikel L. Forcada. (2016). Fuzzy-match repair using black-box machine translation systems: what can be expected. In *Proceedings of the 12th Biennial Conference of the Association for Machine Translation in the Americas*, Volume 1, pp. 27–39, October 2016, Austin, Texas, USA. [**Chapter 2**]

- Rebecca Knowles, John E. Ortega, and Philipp Koehn. (2018). A comparison of machine translation paradigms for use in black-box fuzzy-match repair. In *Proceedings of the 13th Biennial Conference of the Association for Machine Translation in the Americas*, Volume 1, pp. 249–255, March 2018, Massachusetts, Boston, USA. [**Chapter 2**]

- John E. Ortega, Weiyi Lu, Adam Meyers, and Kyunghyun Cho. (2018). Letting a neural network decide which machine translation system to use for black-box

fuzzy-match repair. In *Proceedings of the 21st Annual Conference of the European Association for Machine Translation*, Volume 1, pp. 209-218, 28-30 May 2018, Alicante, Spain. [**Chapter 4**]

- John E. Ortega, Felipe Sánchez-Martınez, Marco Turchi, and Matteo Negri. (2019) Improving translations by combining fuzzy-match repair with automatic post-editing. In *Proceedings of Machine Translation Summit XVII*, Volume 1, Research Track, pp. 256-266, 19-23 August 2019, Dublin, Ireland. [**Chapter 5**]

- John E. Ortega, Felipe Sánchez-Martınez, and Mikel L. Forcada. (2020) Fuzzy-match repair guided by quality estimation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, doi: 10.1109/TPAMI.2020.3021361, 2 September 2020. [**Chapter 3**]

Other papers and patents that I have published, while not directly related to FMR, have to do with translation or natural language processing. Below, is a list of papers in chronological order completed in parallel to this dissertation (in brackets, the field to which the publication pertains).

- Ravi Kondadadi, Girija Yegnanarayanan, Brian William Delaney, and John E. Ortega. (2017) Medical report coding with acronym/abbreviation disambiguation. United States Patent Application 15/045,167, 13 July 2017. [**Natural Language Processing**]

- John E. Ortega, Krishnan Pillaipakkamnatt. (2018). Using morphemes from agglutinative languages like Quechua and Finnish to aid in low-resource translation. In *Proceedings of the 13th Biennial Conference of the Association for Machine Translation in the Americas Workshop on Technologies for MT of Low Resource Languages (LoResMT 2018)*, Volume 1, pp. 1–11, 21 March 2018, Massachusetts, Boston, USA. [**Machine Translation**]

- Adam L. Meyers, Yifan He, Zachary Glass, John E. Ortega, Shasha Liao, Angus Grieve-Smith, Ralph Grishman, and Olga Babko-Malaya. (2018). The termolator: terminology recognition based on chunking, statistical and search-based scores. Frontiers in Research Metrics and Analytics 3, Number 3, p 19, 15 June 2018. [**Natural Language Processing**]

- Samir Undavia, Adam Meyers, John E. Ortega. (2018). A comparative study of classifying legal documents with neural networks. In *Proceedings of the 2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, IEEE, pp. 515–512, 9 September 2018, Poznan, Poland. [**Natural Language Processing**]

- Oscar Ferrandez-Escamez, Neil D. Barrett, Ravi Kondadadi, Girija Yegnanarayanan, Brian William Delaney, and John E. Ortega. (2018) Computer assisted coding

systems and methods. United States Patent Application 15/632,152, 27 December 2018. [**Natural Language Processing**]

- Ravi Kondadadi, Oscar Ferrandez-Escamez, John E. Ortega, Neil D. Barrett, and Brian William Delaney. (2019). Automated analysis system and method. United States Patent Application 16/026,641, US20190027235A1, 24 January 2019. [**Natural Language Processing**]

- John E. Ortega, Richard Alexander Castro-Mamani, Jaime Rafael Montoya Samame. (2020). Overcoming resistance: the normalization of an Amazonian tribal language. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing Workshop on Technologies for MT of Low Resource Languages (LoResMT 2020)*, Volume 1, pp. 1–11, 4 December 2020, China. [**Machine Translation**]

- John E. Ortega, Richard Alexander Castro-Mamani, Kyunghyun Cho. (2020). Neural machine translation with a polysynthetic low resource language. In *Machine Translation*, Machine Translation for Low-Resource Languages, Volume 34, no. 4, pp. 1–23, doi: 10.1007/s10590-020-09255-9, 20 December 2020. [**Machine Translation**]

Contents

Chapter 1

Introduction

Computer-aided translation (CAT) tools based on translation memories (Bowker, 2002; Somers, 2003) are widely used to assist professional translators. A translation memory (TM) consists of a set of translation units (TU) made up of source- and target-language segment pairs. For the translation of a new source segment s', these tools search the TM and retrieve the TUs (s, t) whose source segments are more similar to s'. The translator then chooses a TU and edits the target segment t to turn it into an adequate translation of s'. This dissertation tackles the problem of automatically editing those parts of the translation proposal t that need to be corrected by the user with a *fuzzy-match repair* (FMR) method capable of using any source of bilingual information without access to its inner workings. In this chapter, an introduction to the different components of a CAT tool is presented. First, CAT tool solutions, which are primarily based on TMs, and their history are covered in Section 1.1. Section 1.2 then provides a brief introduction to machine translation (MT) whereas Section 1.3 reports the state of the art on the integration of MT and TM. The chapter ends with an introduction to FMR and a description of the state-of-the-art FMR approaches.

1.1 Computer-aided translation tools

Computer-aided translation (CAT) tools were originally called machine-assisted human translation tools (Bowker, 2002) because the translation process was duly left to the user – the CAT-tool system provides several resources to help with translation but the act of translating a source segment is entirely left up to the user. The main difference between a CAT tool and a machine translation (MT) engine is that in a CAT tool the translator is the one that produces the translation; in MT, on the other hand, the user plays no active role in the translation apart from correcting the output. A few of the resources available to CAT-tool users are spell checking, text search, translation memories, and MT engines (Federico et al., 2012). Most CAT tools use translation memories (TM) as the primary resource to aid in the translation process; in fact, when modern literature talks about CAT tools, they are assumed to be TM-based CAT

tools. A TM is a collection of translation units (TU) that consists of segment pairs (s, t). Each pair contains segments in two languages which are mutual translations. Typically, segments consist of single sentences; but, they may consist of other text units (e.g. phrases or paragraphs).

TM-based CAT tools are a subset of other systems known as terminology-management systems (Bowker, 2002). They play a critical role in the overall CAT-tool translation workflow as described in more detail in Section 1.1.2. The TM-based CAT tool's underlying responsibility is to compare a new source segment to be translated s' with the previously-translated source segments in the TM. If the comparison finds a source segment *similar* to the one to be translated, the corresponding TU (s, t) is returned to the translator for further editing or immediate acceptance of t, as is the case when there is an exact match (both source segments are identical). If post-editing occurs, the source segment to be translated s', along with its translation t' is saved for future use. It is the CAT tool's responsibility to segment the document to be translated and to implement the method for searching the TM for TUs whose source segment is *similar* to the one to be translated. The algorithm that computes the similarity of two segments is of particular interest to our fuzzy-match repair (FMR) method and is described in further detail in Section 1.1.1.

Figure 1.1 illustrates how a TM-based CAT tool works. The main (retrieval) operation of a TM[1] is illustrated for the source segment: *This will also be 100% match.*. The TM retrieves two closely matched TUs (ID 90 and ID 91 from the top window which correspond to ID 1 and ID 2 in the lower window): (1) (*This will also be 100% match.*, *Dies wird auch zu 100% ubereinstimmen.*) and (2) (*This will also be a fuzzy match.*, *Dies wird eine Fuzzy-Match werden.*). The first is a one-hundred percent match with the source segment; and, is probably the segment that the translator would use for the final translation. The second one, nonetheless, could be modified by replacing *be a fuzzy* with *also be 100%* to create a translation. The decision to choose between one or the other is up to the CAT-tool user. If a TM proposal is modified, the newly created TU would be saved to the TM with a new ID (95 for example). The next section describes the algorithm for computing the similarity between two segments in detail.

[1]Example from the TM tool in Wordfast Pro 3 (`https://wordfast.com/WFP3/`).

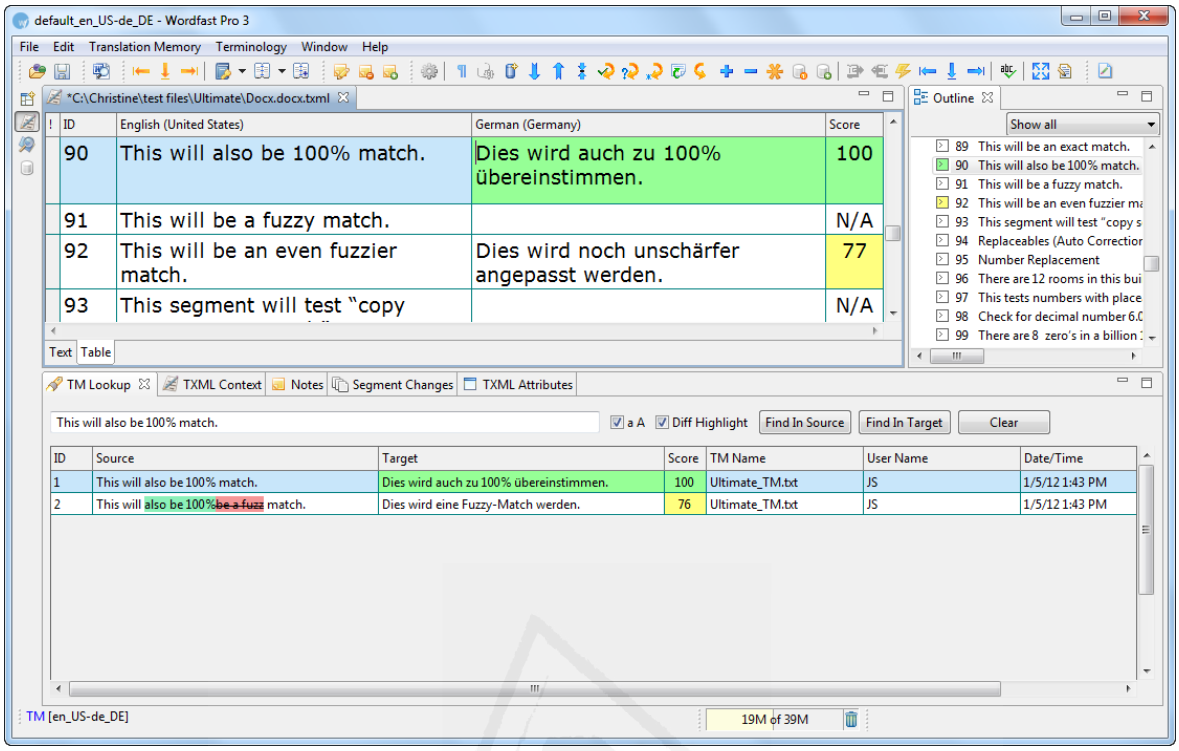


Figure 1.1: A translation memory example using WordFast Pro3 software.

1.1.1 Fuzzy matching

The fuzzy-match algorithm is one of the key methods of how the TU retrieval process works with a TM in a CAT-tool setting. While the similarity distance of two segments can be computed using different types of algorithms like Bitap (Baeza-Yates and Gonnet, 1992), Hamming (Bookstein et al., 2002), or Damerau-Levenshtein (Damerau, 1964), to name a few, the most common method is the word-based edit-distance (Wagner and Fischer, 1974). The edit-distance algorithm may be used to compare the source segment to be translated s' to all the source segments s in the TM by using Equation 1.1 to calculate what is known as the fuzzy-match score (FMS):[2]

$$\mathrm{FMS(s', s)} = \frac{\mathrm{ed}(s', s)}{\max(\mathrm{len(s')}, \mathrm{len(s)})} \tag{1.1}$$

where $\mathrm{ed}(x, y)$ stands for the word-based edit distance between x and y, and len(x) stands for the number of tokens of segment x.

Before the computation of the FMS between two segments, they may be pre-processed in different ways to obtain different FMSs. This pre-processing may include (1) *tokenization*, for separating words and other symbols like punctuation marks into smaller, stand-alone, units; (2) *stemming*, for converting words into their base form or

[2]In OmegaT (`https://omegat.org`), the fuzzy-match score is calculated in this way.

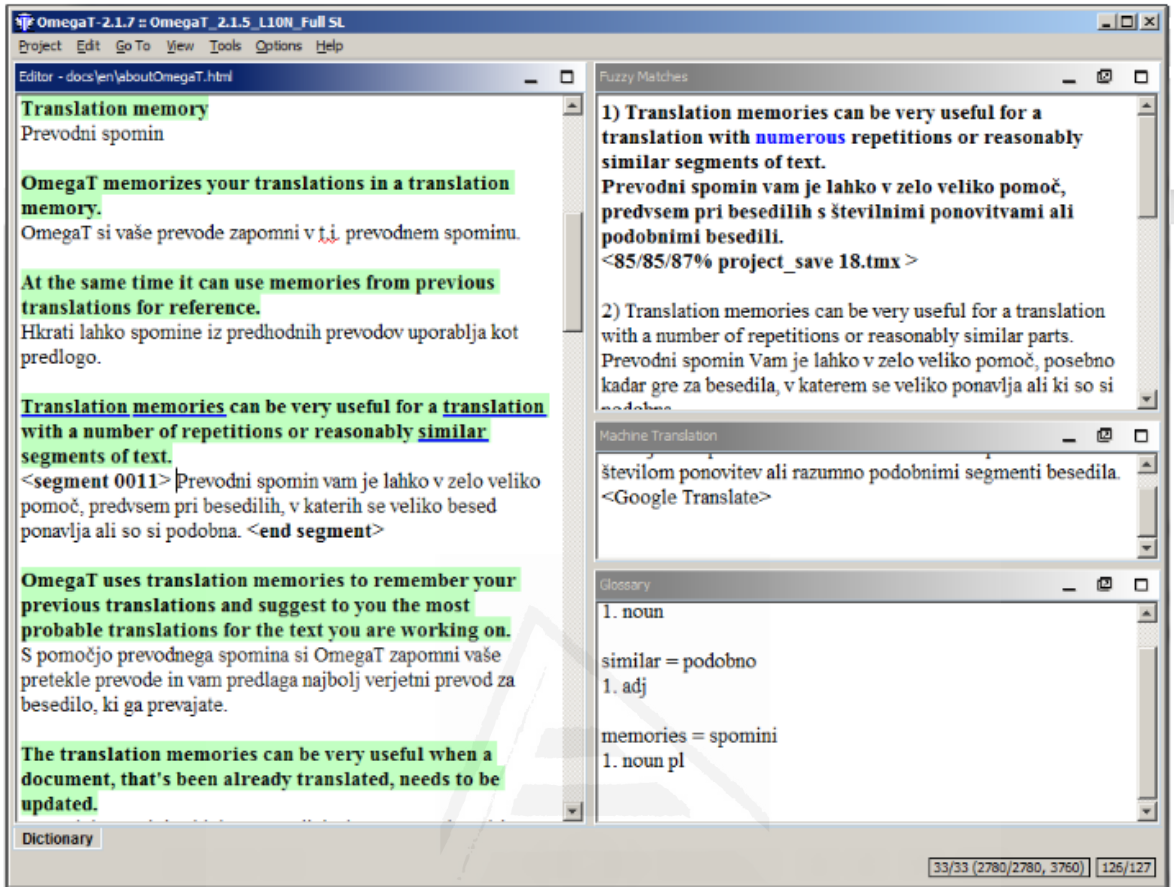

OmegaT-2.1.7 :: OmegaT_2.1.5_L10N_Full SL
Project Edit Go To View Tools Options Help
Editor - docs\en\aboutOmegaT.html
Translation memory
Prevodni spomin
OmegaT memorizes your translations in a translation memory.
OmegaT si vaše prevode zapomni v t.i. prevodnem spominu.
At the same time it can use memories from previous translations for reference.
Hkrati lahko spomine iz predhodnih prevodov uporablja kot predlogo.
Translation memories can be very useful for a translation with a number of repetitions or reasonably similar segments of text.
<segment 0011> Prevodni spomin vam je lahko v zelo veliko pomoč, predvsem pri besedilih, v katerih se veliko besed ponavlja ali so si podobna. <end segment>
OmegaT uses translation memories to remember your previous translations and suggest to you the most probable translations for the text you are working on.
S pomočjo prevodnega spomina si OmegaT zapomni vaše pretekle prevode in vam predlaga najbolj verjetni prevod za besedilo, ki ga prevajate.
The translation memories can be very useful when a document, that's been already translated, needs to be updated.
Fuzzy Matches
1) Translation memories can be very useful for a translation with numerous repetitions or reasonably similar segments of text.
Prevodni spomin vam je lahko v zelo veliko pomoč, predvsem pri besedilih s številnimi ponovitvami ali podobnimi besedili.
<85/85/87% project_save 18.tmx >
2) Translation memories can be very useful for a translation with a number of repetitions or reasonably similar parts.
Prevodni spomin Vam je lahko v zelo veliko pomoč, posebno kadar gre za besedila, v katerem se veliko ponavlja ali ki so si
Machine Translation
številom ponovitev ali razumno podobnimi segmenti besedila.
<Google Translate>
Glossary
1. noun
similar = podobno
1. adj
memories = spomini
1. noun pl
Dictionary
33/33 (2780/2780, 3760)
126/127

Figure 1.2: Fuzzy-match proposal from a TM in OmegaT.

stem (e.g. the verbs *ran*, *run*, and *runs* are converted to the infinitive form *run*) and; (3) *number replacement*, for substituting numbers for a common token. For example, OmegaT,[3] an open-source CAT tool, computes three different FMSs (see Figure 1.2). Assuming that the tokenizer option is turned on in the CAT tool, the first percentage is calculated by using tokenization and stemming on the words only and ignoring formatting tags and numbers. The second one uses tokenization, it does not use stemming on the words and ignores formatting tags and numbers. The third one also only uses tokenization but it includes formatting tags and numbers.

In order to prevent the TM-based CAT tool from proposing matches that have little in common with the segment to be translated, the user can configure a fuzzy-match threshold (FMT). An all-encompassing FMT that maintains productivity is typically around 70% or above according to previous studies (Escartín and Arcedillo, 2015a,b). However, like O'Brien et al. (2017) have shown, often times translators adjust their FMT to the task at hand. Figure 1.1 illustrates the use of a 70% FMT for which two

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

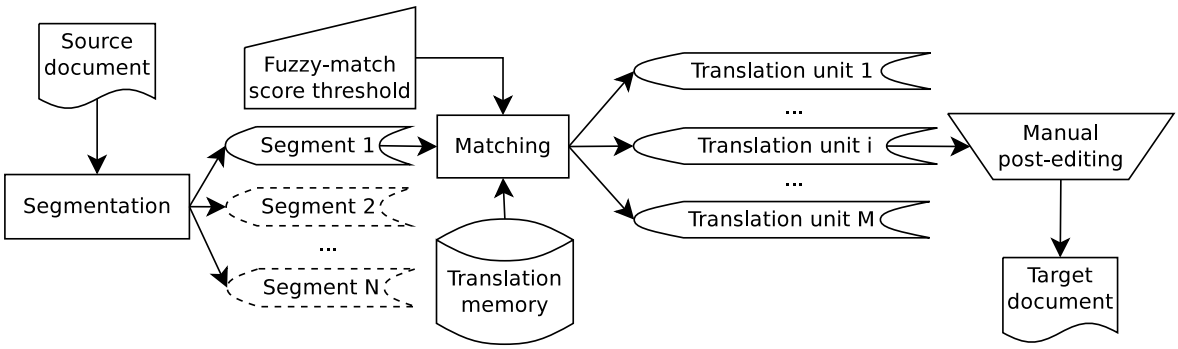


Figure 1.3: A typical baseline workflow for CAT tool users as depicted by Esplà-Gomis et al. (2015).

matches have been found with FMS of 77% and 100%; notice how the CAT tool always provides the FMS for the segment to be translated and the source segment in the TU.

1.1.2 Translation workflow

The typical workflow for translating a source document by means of a TM-based CAT tool consists of: (1) segmentation of the source document, (2) TM lookup for each source segment to be translated, (3) acceptance or modification (post-editing) of the TM proposals, and (4) generation of the target document. This workflow has been well illustrated in previous work (Esplà-Gomis et al., 2015) and is shown in Figure 1.3. FMR is directly related to and fits in between the lookup of TUs similar to the segment to be translated and the proposal of a translation.

The four-step (baseline) process for translating a document with a TM-based CAT tool begins with the division of the source document into several, sentence-level, segments. In order to separate the source document into segments, boundary markers such as punctuation marks can be used. In most cases, a lists of boundaries are used for boundary definitions since segments can vary quite a bit. However, depending on the language and the text, sentence segmentation can be more difficult and are typically broken down into two main approaches: a rule-based approach and statistical based approach (Miłkowski and Lipski, 2009). Rule-based approaches may contain heuristics that determine the beginning and ending boundaries to break up a document into segments. The most common rule-based approach uses a file that contains regular expressions or segment-boundary limits in accordance with an industry standard know as Segmentation Rules eXchange (SRX) (originally maintained by the Localization Industry Standards Association but now maintained by GALA[4]). SRX rules are defined in an XML file that can be transferred from one CAT-tool system to another and are the most widely used segmentation rule types. On the other hand, statistical-based approaches (Mikheev, 2003; Xu et al., 2005; Beeferman et al., 1999; Matusov et al.,

[4] `https://www.gala-global.org/lisa-oscar-standards`

2006; Deng et al., 2007) are most often associated with classification that uses some type of machine learning to determine the beginning and end of each sentence.

The secondary step is the comparison of each source segment to be translated s' to the source segment s in each TU from the TM. All of the s' segments are matched with their corresponding TU as discussed in Section 1.1.1. Then, each s' is presented to the user along with any TM matches that were discovered during the retrieval process and whose bounds meet the FMT. As seen in Figure 1.2, TUs are typically ranked in a list from top to bottom with the most similar segment at the top. The right-hand side pane displays several matches of which the top match is highlighted.

After reviewing the ranked segments in the list, the third step consists of the CAT-tool user either accepting one of the matching segments (typically 100% fuzzy-matches are accepted) or modifying it which, in turn, saves it to the TM. After iterating through steps 2 and 3 with the different segments in the document to be translated, the final document translation is generated. This workflow is the most typical workflow in a CAT-tool setting and depends on the existence of a TM.

The baseline translation workflow described above can be modified to include an MT engine to translate all of those segments for which there is no fuzzy match available. This approach can be considered practical if the MT engine produces translations whose post-editing requires less time than translating those segments from scratch. For those segments for which there is a fuzzy-match above the FMT, it may also be possible to obtain further productivity gains using the MT-translated segments. While the MT translation may contain errors and lack quality, as shown in a previous study (O'Brien et al., 2017), the total amount of editions needed to correct it may be less than the number of editions to convert a translation proposal into the desired translation. Those cases (the post-editing of the MT engine output instead of the TM target segment) have been shown to be accepted as closer to human translations in previous research by Carl et al. (2015) where 57% of translators preferred the MT engine output over the TM output alone. In those cases, fuzzy matches and MT-translated segments need to be ranked together, this can be done if MT quality estimation (QE) methods, either at the segment- or word-level (as performed by Esplà-Gomis et al. (2019)), are used.

1.2 Machine translation

As aforementioned, TM-based CAT tools may also integrate machine translation (MT). MT may be defined as the translation by means of a computer program of texts in one language into another language. Using MT to achieve satisfactory, near-human, results can be considered difficult because natural languages spoken by humans are complex and ambiguous (Costa-jussà, 2012). Words and grammatical constructs that exist in one language may have different meanings in other languages (Jakobson, 1959). Often times, words in one language may not be directly translatable to another language.

There has been a need for MT for many years, recorded as far back as the early 1930's (Hutchins, 2007). The early MT systems were, for the most part, based on a bilingual dictionary and merely substituted words in the source language for those parallel words in the the target language. Later, in 1947, Warren Weaver, director of Natural Sciences Division of the Rockefeller Foundation, recognized that there was a "communication problem" (Hutchins, 1997) in the world and set out to solve it by first defining the problem as a "translation problem". That is when the world began to see MT and the creation of MT systems as an attempt to disambiguate natural language. Since then, numerous MT systems have been created in an attempt to solve the problem. However, we can group the most recent MT paradigms into two main categories: (1) *knowledge-based* and (2)*corpus-based* MT systems.

Knowledge-based MT (KBMT) systems can be defined as those systems that incorporate linguistic knowledge explicitly recorded by human experts in the form of rules (Nirenburg et al., 1994). KBMT is also referred in the literature as rule-based MT (RBMT) since, for the most part, they are based on rules. In general, rules have to be created for every language pair. Today, one of the most used RBMT system is an open-source system called Apertium (Forcada et al., 2011). Apertium allows users to create the resources needed for translating between any pair of languages (mainly morphological dictionaries, disambiguation rules, bilingual dictionaries, and structural-transfer rules). Under-resourced language pairs that cannot benefit from corpus-based approaches often times use RBMT systems like Apertium for their first steps of introducing human knowledge into an MT system.

Corpus-based MT (CBMT) systems are based on statistics and probability. Sometimes called context-driven (Carbonell et al., 2006) approaches, they use features from the source and target languages to attempt to derive a correlating function between the two languages. This allows them to automatically learn to translate from parallel corpora i.e. collections of texts in one language together with their translations into another language. Since the approximation of a correlating function is necessary, mathematical concepts are key features in CBMT systems and have given way to two main paradigms today: (1) statistical machine translation (SMT) and (2) neural machine translation (NMT) . Both SMT and NMT systems are better generic approaches than RBMT for translation when there is an abundance of training data to train a model that will approximate translation well. At the time of this writing, the most used SMT system is Moses (Koehn et al., 2007) and one of the most used NMT systems is Nematus (Sennrich et al., 2017). When there is a plethora of data, NMT systems have generally been found to outperform other MT paradigms (Koehn and Knowles, 2017; Knowles et al., 2018). In some cases, even with low amounts of data, NMT systems have been found to outperform other MT paradigms through hyperparameter tuning. (Sennrich and Zhang, 2019) NMT systems are based on neural networks of which we focus on two main types of architectures, recurrent neural networks (RNN) and transformer-based neural networks. While both are neural networks, they differ in the approach used for translation. RNNs were initially used to translate using

sequence-to-sequence (Sutskever et al., 2014; Cho et al., 2014b), or encoder-decoder, approaches which later evolved into more effective approaches that used attention and could jointly align and translate (Bahdanau et al., 2015; Luong et al., 2015). Not long after, the attention idea was expanded to process segments in a non-sequential order using transformer-based models (Vaswani et al., 2017) that contain several (multi-headed) attention components. In this thesis, we use Apertium, Moses, and Nematus (with the RNN encoder-decoder approach) in our experiments to show the power of the three paradigms.

1.3 Translation memory and machine translation integration

In addition to the use of MT to translate those segments for which there is not a fuzzy match found in the TM, there are more sophisticated ways of integrating TM and MT. In the literature, one can find two main types of approaches integrating TM and MT: (1) approaches that integrate sub-segments from the TM into the decoding process of an MT system and (2) approaches that use the target segment t from a TU (s,t) as the backbone of the translation to be produced.

Of the two types of approaches introduced above, the majority of approaches combine the TM and MT engine by using the first approach, i.e. by introducing sub-segments from the TUs (s,t) in the TM into the decoding process of the MT system. Biçici and Dymetman (2008), for example, use a phrase-based SMT system trained on a bilingual corpus in the same domain as the TM and combine it with the TM's fuzzy match by extracting a phrase table that is dynamically added to the usual set of bi-phrases used for decoding the source. Their implementation augments the internal translation table in the SMT system with bilingual non-contiguous sub-segments (phrases) that have source sub-segments in common with the source segment to be translated s'. Alignments in the system created by Biçici and Dymetman (2008) are detected using word alignments directly obtained from the SMT system training process and are used to find the parts of t that need to be edited (mismatches).

Similarly, Simard and Isabelle (2009) use a phrase-based SMT system by adding phrase pairs (sub-segment pairs) of any length (obtained using a statistical aligner on the TM) to the SMT system's phrase table and introduce a feature to flag those phrase-pairs that come from their TM. After that, they optimize the weighting of the TM-based phrase table in a regular SMT decoder. By means of optimization and phrase table inclusion they are able to make their SMT system produce a translation close to the desired translation t'.

Additional work done by Zhechev and Genabith (2010) makes use of a phrase-based SMT system along with an alignment method that, like Simard and Isabelle (2009), connects source and target sub-segments from the TUs (s,t) in the TM. The alignment method Zhechev and Genabith (2010) use takes advantage of a tree-based structural

alignment created from a bilingual dictionary after training their SMT system with phrase pairs. After aligning the words in s with those in t, Zhechev and Genabith (2010) are able to identify words that should appear in the final translation t'.

Koehn and Senellart (2010) take a similar approach to Biçici and Dymetman (2008). They first align words in s' and s to find mismatches. Then, they align the words in s and t to identify target matches and remove the words in t that are aligned to the mismatched words in s. Target mismatches are sent to the SMT decoder for translation. Mismatched words in Koehn and Senellart (2010)'s system are treated separately; that is, context around a mismatch, while indirectly taken into account by the language model, is not directly taken into account when applying phrase pairs.

Ma et al. (2011), on the other hand, decided to research the shortcomings of using a fuzzy-match score as a threshold for determining translation unit matches that serve as translations for other segments. Ma et al. (2011)'s approach uses discriminate learning and support vector machines to salvage translations of matched words from translation units that would have been otherwise thrown away due to the fuzzy-match score being used as a threshold. Their work, unlike Koehn and Senellart (2010), takes matched parts in s and replaces them with their counterparts in t. The main drawback of the approaches from Ma et al. (2011), Koehn and Senellart (2010), Zhechev and Genabith (2010), and Biçici and Dymetman (2008) is that they are all based on MT and either have access to the internals of an MT system trained on the user's or related data or modify its behavior in some way.

The approaches in the second group of TM-MT integration approaches, i.e. those that use the target segment t in a TU (s,t) as the backbone of the translation to be produced are closer to the approach introduced in this dissertation. In the next section, we introduce the concept of *fuzzy-match repair*, which belongs to the second group of TM-MT integration approaches, and highlight the main differences between our approach and those in the literature.

1.4 Fuzzy-match repair

Fuzzy-match repair approaches differ from the approaches described above in that they use the target segment t in the TU (s, t) as a backbone for the translation to be produced. Its integration into a CAT tool is illustrated in Figure 1.4. FMR's main goal is to replace the sub-segments in the target segment t that are the translation of the sub-segments in the source segment s that do not appear in the source segment s' to be translated with the translation of the corresponding sub-segment in s'.

The aim of this dissertation's FMR approach is to replace only the mismatched sub-segments in t while at the same time saving the sub-segments in t that can be reused, since they have already been professionally translated. The approach first aligns the words in the source segment s of the TU being repaired (s, t) with the words in the source segment to be translated s' and identifies the mismatched words in s

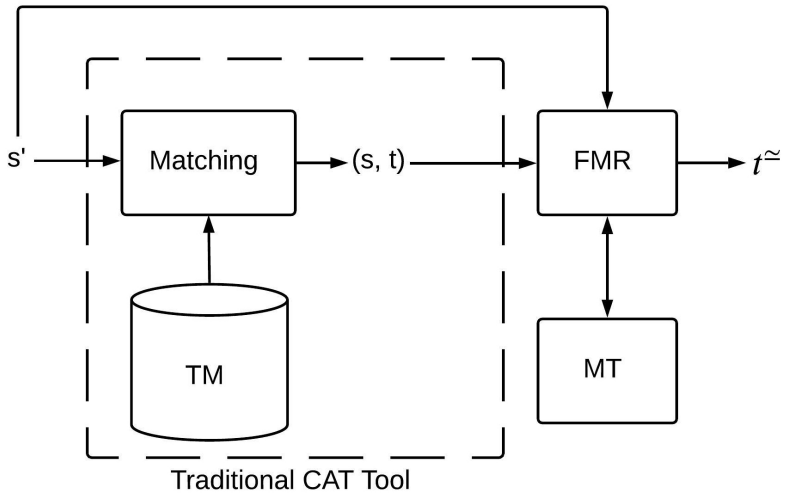


Figure 1.4: The addition of fuzzy-match repair (FMR) in a traditional computer-aided translation (CAT) pipeline. A source segment s' is first *fuzzy-matched* with a translation unit (s, t) from the translation memory (TM). The target segment t is then fuzzy-match repaired to get $t^{\simeq}$, a new *repaired* proposal presented to the CAT tool user.

and s', that is, the sub-segments they do not have in common.[5] It then uses MT, although it could use any other source of bilingual information (SBI) , to identify the sub-segments in t that are the translations of the mismatched sub-segments in s, in a way similar to that of Esplá-Gomis et al. (2011), and builds a set of *repair operators* by translating the mismatched sub-segments in s'. Each repair operator specifies the TL sub-segment τ in t that needs to be repaired and the TL sub-segment τ' to be used for repairing. Combinations of compatible repair operators are then applied to obtain a set of candidate fuzzy-match repaired segments from which the one to be finally used is selected after estimating their quality.

There are several other repair methods similar to ours that do not use quality estimation techniques. Kranias and Samiotou (Kranias and Samiotou, 2004) use several linguistic resources —such as bilingual dictionaries and lists of suffixes and closed-class words— to align the words in s to those in t and use these alignments to identify the words in t to be repaired. The words to be repaired are then replaced (edited, inserted or deleted) by the translation of the corresponding mismatch in s' obtained using MT. This method is similar to the one we focus on in this dissertation, but differs in that it uses context around the mismatches only when the new segment s' contains words not found in s that need to be inserted. In contrast, we use context around all mismatches, when available, and this allows us to treat insertions, deletions and substitutions in the same way. It also allow us to mitigate the incomplete reordering and agreement errors that may occur because of not using context. In addition, Kranias and Samiotou (2004) only produce a single fuzzy-match repaired segment, whereas we produce as

[5]This is usually obtained as a by-product of fuzzy matching.

many fuzzy-match repaired segments as possible and then select the best one using quality estimation (QE) techniques.

Hewavitharana et al. (2005) first use a modified IBM model 1 to align the mismatched words in s to sequences of one or more words in t and then directly map the sequence of source-side one-word edit operations (substitutions, deletions and insertions) needed to convert s into s' into an identical sequence of edit operations on the corresponding word sequences in t to generate the repaired translation. An important strength of the method by Hewavitharana et al. (2005) is that multiple alternative target-side edits are possible for each source-side insertion or substitution, and that they are scored using a probabilistic model. An important limitation, as compared to ours, is the lack of source context around source-side one-word edits.

The method described by Dandapat et al. (2011) first aligns, in a way similar to ours, the words in s and s' using the (word-based) edit distance (Wagner and Fischer, 1974) and marks the mismatched sub-segments in s and s' for translation. The mismatched sub-segments in s are then aligned with their counterparts in t by using a sub-segmental TM built on the user's TM by means of the method used to obtain phrase tables in statistical MT (Koehn, 2010). Finally, the sub-segments in t aligned to mismatched sub-segments in s are replaced by the translations of the corresponding sub-segments in s' as they are found in the sub-segmental TM. There are three main differences with our approach. First, context around mismatches is not taken into account, which may lead to incorrect translations due to *boundary friction* problems (Carl and Way, 2003, p. 341), such as incorrect agreement or incomplete word reorderings. Second, the methods relies on the user's TM (which may be small) rather than on an external SBI. And third, a single fuzzy-match repaired segment is produced, even when the sub-segmental TM contains several translation alternatives per sub-segment, whereas we generate as many fuzzy-match repaired segments as possible and then use QE to select the best one.

A method for FMR to be applied on close matches is described by Bulté et al. (2018). Their method first performs a simple punctuation repair and then applies a set of edit operations —deletions, insertions and substitutions— to the target segment t of the TU (s, t) being repaired. To detect the target sub-segments to be repaired it uses statistical word alignment models. For deletions, it just removes from t the translation of the mismatched sub-segment in s, and the word to the left of the sub-segment to be removed if it is not aligned to any word in s. For insertions, it inserts in t the new sub-segment between the two target words aligned with the matched words in s surrounding the sub-segment in s' not appearing in s. For substitutions, it translates *anchored* mismatched sub-segments in s', that is, mismatched sub-segments surrounded by words common to s, one word at each end. The mismatched sub-segments are translated in-context by translating the whole segment s' using statistical MT and constraining the output of the MT system to use the sub-segments of t aligned to sub-segments of s common to s' in a way similar to that used in Simard and Isabelle (2009); Zhechev and Genabith (2010); Koehn and Senellart (2010). The main differences between this

FMR approach and ours is that it uses statistical word alignments and treats in a different way substitutions, deletions and insertions, whereas our approach does not explicitly compute the alignment between the words in s and t and simply performs string substitutions in t. In addition, this approach is not able to repair a target segment t if the translation of the mismatched source sub-segment does not consist of contiguous words in t; our approach does not have this limitation because it does not impose any constraint on the amount of anchored words surrounding a mismatch sub-segment and their location. Lastly, it can only use MT systems, such as Moses (Koehn et al., 2007), that allow part of the translation to be fixed beforehand; in contrast, our approach can benefit from any available SBI.

Finally, it is worth noting that some commercial CAT tools implement FMR methods. For example, MemoQ[6] implements a feature called MatchPatch that uses term bases and other resources for FMR, while Déjà Vu implements a feature called DeepMiner[7] that extracts sub-segments from the very same TM being used for their use for FMR. Unfortunately, details about how these methods work are not available. All of these approaches can be considered a form of FMR similar to ours which is formally introduced in the next chapter.

[6] https://www.memoq.com/whats-new-in-memoq-2015
[7] https://atril.com/key-features/

Chapter 2

Fuzzy-match repair using black-box machine translation

This chapter goes over our fuzzy-match repair approach and the details of the algorithm used for generating candidate fuzzy-match repaired segments. Experiments and results are provided using a well-known DGT-TM translation memory and, initially, the rule-based machine translation system, Apertium (Forcada et al., 2011). Additionally, in order to compare other types of MT systems, experimentation is conducted using Moses (Koehn et al., 2007) and Nematus (Sennrich et al., 2017). We show that FMR can successfully edit target segments from the TM and propose fuzzy-match repaired segments that are closer to the desired translation.

2.1 Overview

In the following sub-sections, we illustrate the five steps followed by our FMR approach at a high level by showing how repair operators are generated from a source English segment s' to be translated into Spanish and a TU (s,t). We then cover how candidate fuzzy-match repaired segments in Spanish are generated.

2.1.1 Step 1: find mismatches and align

We first find the mismatched sub-segments between the source segment to be translated s' and the source segment s in the TU (s,t). Imagine we are translating from English to Spanish and the new segment to be translated is:

s' = "The blue dog barks loud when it rains at night"

The system shows a translation unit (s, t) from the translation memory and marks mismatched words (in bold below):

s' = "The **blue** dog barks loud when it rains at night"
s = "The **red** dog barks loud **sometimes** when it rains at night"
t = "El perro rojo ladra fuerte a veces cuando llueve por la noche"

According to Eq. (1.1) the fuzzy-match score is $\text{FMS}(s', s) = 81.8\%$, and as a side result of the computation of the edit distance (Wagner and Fischer, 1974) between s' and s, the alignment between the words in s' and those in s is produced. The edit operations needed to convert s into s' are:

- one substitution: replace the word **red** in s by the word **blue** from s', and
- one deletion: delete the word **sometimes** in s between the words *loud* and *when*.

2.1.2 Step 2: translate source-side sub-segments covering mismatches

Each one of the sub-segments σ from s covering a mismatch is sent to the external machine translation system to obtain their corresponding target language translations τ. The hope is to get a translation τ that exists in t so that the corresponding sub-segment can be later replaced by the translation of the mismatched sub-segment σ' from s' to produce the desired translation t'.

If we choose Apertium (Forcada et al., 2011) as the MT system to translate the sub-segments σ covering a mismatch, we get the following (σ, τ) pairs:[1]

- (the red dog, el perro rojo)
- (the red, el rojo)
- (red dog, perro rojo)
- (red, rojo)
- (loud sometimes when, fuerte a veces cuando)
- (loud sometimes, fuerte a veces)
- (sometimes when, a veces cuando)
- (sometimes, a veces)

2.1.3 Step 3: match the source-side translations to t

In step 3, we identify the (σ, τ) pairs for which τ appears in t and can, therefore, be used to build repair operators. We keep the (σ, τ) pairs whose τ appears in t as a contiguous sub-segment; see Table 2.1.

[1] For this example, only sub-segments of up to length three are considered.

σ	position in s	τ	position in t	outcome
the red dog	1–3	el perro rojo	1–3	kept
the red	1–2	el rojo	–	discarded
red dog	2–3	perro rojo	2–3	kept
red	2–2	rojo	3–3	kept
loud sometimes when	5–7	fuerte a veces cuando	5–8	kept
loud sometimes	5–6	fuerte a veces	5–7	kept
sometimes when	6–7	a veces cuando	6–8	kept
sometimes	6–6	a veces	6–7	kept

Table 2.1: The (σ, τ) pairs where the translation τ of mismatched sub-segment σ appears in the target segment t. Since the τ sub-segment, *el rojo*, is not found in t, it is discarded.

Sub-segments that do not have a match in t (e.g. the second one in Table 2.1) are discarded and not used further in the repair process. Word positions in t not covered by any translation τ of any sub-segment σ in s contain words for which there is no evidence to modify them. In the absence of information, they are not changed.

2.1.4 Step 4: pair τ and τ' together to form repair operators

After the initial matching occurs from the translations of sub-segments in s to form (σ, τ) pairs, (σ', τ') pairs are created by translating mismatched sub-segments from the source segment to be translated s'. The alignment found between words in s and words in s' during fuzzy matching is used to extract phrase pairs that project mismatched sub-segments σ in s to their corresponding mismatched sub-segments σ' in s'.[2] The σ' sub-segments are sent to the MT system to get their translations τ'.

Once the translation of the σ' sub-segments is obtained, fuzzy-match repair operators are created. A repair operator is a tuple of string-positioned sub-segments (σ,σ', τ,τ') where τ' is used to replace (repair) the target sub-segment τ in t. Table 2.2 shows the σ and σ' sub-segments and their τ and τ' translations computed using Apertium. There were a total of eight original (σ, τ) pairs in Table 2.1. The second (σ, τ) pair ("the red", "el rojo") is discarded due to the lack of a τ match in t and, thus, does not show up as a repair operator in Table 2.2 where there are 7 total operators.

In the running example, most of the τ' sub-segments are aligned word by word to their corresponding τ sub-segments in part because their source sub-segments (σ' and σ) are also aligned word by word. Notice, however, that the fourth τ' (*fuerte cuando* in Table 2.2) does not align word by word to its corresponding τ sub-segment (*fuerte a veces cuando*) because it is a deletion case where the sub-segment (*a veces*) should be deleted.

.

[2]For more details on alignment and phrase extraction see section 2.2.

	σ	σ'	τ	τ'
#1	*the red dog* [1–3]	*the blue dog* [1–3]	*el perro rojo* [1–3]	*el perro azul*
#2	*red dog* [2–3]	*blue dog* [2–3]	*perro rojo* [2–3]	*perro azul*
#3	*red* [2–2]	*blue* [2–2]	*rojo* [3–3]	*azul*
#4	*loud sometimes when* [5–7]	*loud when* [5–6]	*fuerte a veces cuando* [5–8]	*fuerte cuando*
#5	*loud sometimes* [5–6]	*loud* [5–5]	*fuerte a veces* [5–7]	*fuerte*
#6	*sometimes when* [6–7]	*when* [6–6]	*a veces cuando* [6–8]	*cuando*
#7	*sometimes* [6–6]		*a veces* [6–7]	ϵ

Table 2.2: A set of repair operators for the source segment to be translated "The blue dog barks loud when it rains at night" and the translation unit ("The red dog barks loud sometimes when it rains at night","El perro rojo ladra fuerte a veces cuando llueve por la noche").

2.1.5 Step 5: apply the repair operators

Once the set of repair operators has been built, the final step is to apply them to t to create fuzzy-match repaired segments. It is entirely possible to have multiple combinations of repair operators that form multiple repaired segments $t^{\simeq}$. Below, are some possible results of applying repair operators to t from the running repair example after comparing them to the reference translation t' (*el perro azul ladra fuerte cuando llueve por la noche*):

- $t_1^{\simeq}$ = *el perro azul ladra fuerte cuando llueve por la noche* - (**correct**, produced by #1 and #4)
- $t_2^{\simeq}$ = *el perro azul ladra fuerte* ***a veces*** *cuando llueve por la noche* - (**incorrect**, produced by #2)
- $t_3^{\simeq}$ = *el perro* ***rojo*** *ladra fuerte* ***a veces*** *cuando llueve por la noche* - (**incorrect**, produced by #4)
- $t_4^{\simeq}$ = *el perro azul ladra fuerte cuando llueve por la noche* - (**correct**, produced by #2 and #4)

Repair operators can deal with all three types of edit operations: substitution, insertion, and deletion. The repair examples shown above depict a substitution and a deletion example. Nonetheless, insertions can be handled using repair also. Insertions occur when τ' contains new words not in τ (most likely because its corresponding σ' contained words that were not in the corresponding σ).

Note that deletions are typically produced with some overlap, that is, in context. The example above, for instance, creates a repair operator (#4) that deletes the word *sometimes*. We are able to use the context of the surrounding words *loud* and *when* to determine deletion. Context used for repair in this manner is different from other research: Hewavitharana et al. (2005), for example, apply all possible context-free deletions according to statistical word alignment, and later score the resulting segments to determine the best translation.

It is worthwhile to note that repair operators, as seen above, may not always be applicable because positions in t that are modified by a repair operator may not be available for another repair operator when τ does not match the partially-repaired sentence. That would make the two repair operators incompatible. The incompatibility issue for repair operators and its implementation is covered in detail in section 2.3.

The FMR method generates all possible fuzzy-match repair segments $t^{\simeq}$ obtained by applying all possible sets of mutually compatible repair operators (see Section 2.2.1). For the experiments in this chapter, candidate fuzzy-match repaired segments are determined to be correct (or not) by comparing them to previously-translated "gold" segments in a test set. The results of this oracle evaluation shows that our approach is capable of generating fuzzy-match repaired segments close to the desired translation. Chapter 3 discusses a quality estimation method to automatically select the fuzzy-match repaired segment to show to the CAT-tool user.

2.2 Fuzzy-match repair algorithm

A high-level description of how repair operators are built has been given. In order to better illustrate the high-level steps, we dive deeper into the fuzzy-match repair algorithm's implementation. This section covers a *two-part* algorithm that entails: 1) the generation of fuzzy-match-repair operators and 2) the exploration of potential repair operator combinations to generate the set of candidate fuzzy-match repaired segments.

2.2.1 Generation of repair operators

Algorithm 1 describes the procedure for building the list of repair operators to be applied for the generation of candidate fuzzy-match repaired segments.

To obtain the set of repair operators to be used, first, the alignment between the words in s' and those in s is obtained as a by-product of the computation of the (word-level) edit-distance (Wagner and Fischer, 1974) between s' and s, a component of the fuzzy-match score. The string-positioned sub-segment pairs (σ, σ'), containing unaligned (unmatched) words and their corresponding positions in s and s', are then obtained by using the phrase-pair extraction algorithm used in phrase-based statistical MT to obtain bilingual phrase pairs (Koehn, 2010, section 5.2.3). These sub-segment pairs are then translated into the target language to obtain the sets M and M' of sub-segment translations μ and μ' respectively using the SBI available. Finally, these translations are used to build repair operators by looking for all the occurrences in t of each target sub-segment μ to get the corresponding string-positioned target sub-segments τ, and then associating to each τ the target sub-segment μ' to get τ', the sub-segment to be used for repairing. If μ is not found in t, no repair operator can be built. This acts as a quality check that prevents the algorithm from building low-quality repair operators.

Algorithm 1 BuildRepairOp$(s', (s,t))$ generates the list of repair operators to use.

Input: SL segment to be translated s'; TU (s,t) to be repaired
Output: A list of repair operators P

```
1: P ← ()   ▷ Initially P is an empty list
2: A ← EditDistanceAligner(s', s)
3: for (σ, σ') ∈ ExtractPhrasePairs(s', s, A) do
4:    M ← Translate(σ)   ▷ Set with translations of σ
5:    M' ← Translate(σ')   ▷ Set with translations of σ'
6:    for μ ∈ M do
7:       for μ' ∈ M' do
8:          for τ ∈ FindInSegment(μ, t) do
9:             τ' ← AttachTranslationToString(τ, μ')
10:            append (σ, σ', τ, τ') to P
11:         end for
12:      end for
13:   end for
14: end for
15: return P
```

The example in Figure 2.1 illustrates how the list of repair operators is built. It is worth noting that only in those cases in which μ, the translation of σ, is found as a contiguous segment of words in the target segment t of the TU being repaired a repair operator can be built; this is indicated by the fifth column in the table. This acts as a quality check to avoid creating repair operators for sub-segments with insufficient context that lead to translations that do not appear in t.

2.2.2 Generation of candidate fuzzy-match repair segments

Candidate fuzzy-match repaired segments are built from the list of repair operators P by combining them in all possible ways. This is done through a backtracking depth-first exhaustive search, depicted in Algorithm 2, that incrementally builds fuzzy-match repaired segments $t^{\simeq}$.

The algorithm is initialized with two calls Repair$(P, \emptyset, 1, (s,t), t, \textbf{false}, ())$ and Repair$(P, \emptyset, 1, (s,t), t, \textbf{true}, ())$, where () stands for an empty list. At each level of the recursion tree a new repair operator is considered and tested for applicability ($D = \textbf{true}$) or discarded ($D = \textbf{false}$). For a repair operator to be applicable it needs to be compatible with the set of repair operators O applied so far to build $t^{\simeq}$ (see Section 2.3). If it is compatible with the rest of repair operators in O, the repair operator is added to O and applied (lines 3–4); otherwise the branch of the recursion tree is cut. When a leaf of the recursion tree is reached (i.e. $n = \text{length}(P)$) the corresponding fuzzy-match repaired segment $t^{\simeq}$ is added to the list T of candidate fuzzy-match repaired segments. The algorithm ApplyRepairOp$(o, t^{\simeq})$ replaces in $t^{\simeq}$

σ	σ'	μ	μ'	μ in t?
Gina found [1-2]	*found* [2]	*Gina encontró*	*encontró*	no
Gina found [1-2]	*Bill found* [1-2]	*Gina encontró*	*Bill encontró*	no
Gina found out [1-3]	*found out* [2-3]	*Gina se enteró*	*se enteró*	yes
Gina found out [1-3]	*Bill found out* [1-3]	*Gina se enteró*	*Bill se enteró*	yes
found [2]	*Bill found* [1-2]	*encontró*	*Bill encontró*	no
found out [2-3]	*Bill found out* [1-3]	*se enteró*	*Bill se enteró*	yes
about the [4-5]	*about the fraud* [4-6]	*sobre el*	*de la estafa*	no
about the news [4-6]	*about the* [4-5]	*de noticias*	*sobre el*	no
about the news [4-6]	*about the fraud* [4-6]	*de las noticias*	*de la estafa*	yes
the [5]	*the fraud* [5-6]	*el*	*la estafa*	no
the news [5-6]	*the* [5]	*las noticias*	*el*	yes
the news [5-6]	*the fraud* [5-6]	*las noticias*	*la estafa*	yes

Figure 2.1: Example illustrating how the list of repair operators is built. The segment s' = *Bill found out about the fraud* is to be translated into Spanish with the help of the TU (s,t)=(*Gina found out about the news*, *Gina se enteró de las noticias*). Unmatched (unaligned) words in s' are *Bill* and *fraud*; unmatched (unaligned) words in s are *Gina* and *news*. The string-positioned sub-segment pairs (σ,σ') shown are those up to length 3 that contain at least an unmatched word. Their translations (μ,μ') into Spanish are also provided. In this example, we assume that every σ and σ' has a single translation, that is, that M and M' are singletons.

the sub-segment τ by τ'; this can be safely done if repair operator P_n is compatible with the other repair operators applied so far.

This algorithm takes advantage of the fact that repair operators that are compatible can be applied in any order because the repaired segment to be generated would be the same. Thanks to this assumption, the worst-case complexity of the algorithm is $O(2^n)$, with $n = \text{length}(P)$, in which case 2^n repaired segments would be produced.[3] However, as many repair operators are incompatible, the actual complexity of the algorithm is well below the worst case (see Section 2.6).

For the example introduced in section 2.2.1, Algorithm 1 would produce $2^6 = 64$ repaired segments if all 6 repair operators were compatible. However, most of them are not compatible because they edit the same words in t and the algorithm ends up producing only 25 repaired segments. Some of these 25 fuzzy-matched repaired segments are identical but are produced by applying a different set of repair operators. For instance, the repaired segment *Bill se enteró de la estafa* is produced by applying the repair operator (*Gina found out*, *Bill found out*, *Gina se enteró*, *Bill se enteró*) and either the repair operator (*about the news*, *about the fraud*, *de las noticias*, *de la estafa*) or the repair operator (*the news*, *the fraud*, *las noticias*, *la estafa*).

[3]If the algorithm had to explore the application of all the repair operators in P and in all possible orders its worst-case complexity would be super-exponential.

Algorithm 2 Repair$(P, O, n, (s,t), t^{\simeq}, D, T)$ generates all possible fuzzy-match repaired segments by backtracking.

Input: List of repair operators P; set of repair operators O applied so far; position in P of the repair operator being considered, n; TU to be repaired (s,t); fuzzy-match repaired segment being built $t^{\simeq}$; boolean D indicating whether the n-th repair operator in P will be attempted to apply (true) or not (false), list T containing fuzzy-match-repaired segments

1: **if** D **then**
2: **if** Compatible$(P_n, O, (s,t))$ **then**
3: ApplyRepairOp$(P_n, t^{\simeq})$
4: $O \leftarrow O \cup \{P_n\}$ ▷ Add compatible repair operator
5: **else**
6: **return** ▷ Prune this branch of the recursion tree
7: **end if**
8: **end if**
9: **if** $n = \text{length}(P)$ **then**
10: **append** $t^{\simeq}$ **to** T ▷ Add candidate fuzzy-match repaired segment to list T
11: **return** ▷ All the repair operators have been considered
12: **else**
13: Repair$(P, O, n+1, (s,t), t^{\simeq}, \textbf{true}, T)$ ▷ Continue by applying operator $n+1$
14: Repair$(P, O, n+1, (s,t), t^{\simeq}, \textbf{false}, T)$ ▷ Continue by not applying operator $n+1$
15: **end if**

2.3 Compatibility between repair operators

Two repair operators are deemed incompatible, and therefore cannot be applied to build the same candidate fuzzy-match repaired segment, if they edit the same word in t or if they work on the same source-side mismatch, that is, if they take care of the same change in s. Note that there may be repair operators that do not edit any word in t but introduce missing ones. In those cases, if they were applied to build the same candidate fuzzy-match repaired segment, they could end up producing candidate fuzzy-match repaired segments $t^{\simeq}$ with repeated words. The following example illustrates this situation. Suppose the segment s' = *the size does not exceed 100 cm* to be translated with the help of the translation unit (s,t) = (*the size does not exceed 100*, *el tamaño no supera los 100*) whose target segment can be repaired with the two repair operators $(\sigma_1, \sigma_1', \tau_1, \tau_1')$ = (*exceed 100*, *exceed 100 cm*, *supera los 100*, *supera los 100 cm*) and $(\sigma_2, \sigma_2', \tau_2, \tau_2')$ = (*100*, *100 cm*, *los 100*, *los 100 cm*). Both repair operators do not edit (change) any word in t but if they are applied one after the other the result would be the fuzzy-match repaired segment $t^{\simeq}$ = *el tamaño no supera los 100 cm cm*, which contains duplicated words due to the fact that the word *cm* is to be inserted by both operators.

To avoid this problem we need to identify when two repair operators work on the same source-side mismatch, and to do so, one needs to check the mismatches both in s and s' because there may be words in s not appearing in s' (the mismatch only shows up in s), or words that do not appear in s but are introduced in s' (the mismatch only shows up in s', as in the example above). Hence two repair operators $o_i = (\sigma_i, \sigma'_i, \tau_i, \tau'_i)$ and $o_j = (\sigma_j, \sigma'_j, \tau_j, \tau'_j)$ will be marked as incompatible if they edit the same word in t or they meet the following condition:

$$\begin{gathered}(\text{mismatch}(\sigma_i, s) \cap \text{mismatch}(\sigma_j, s) \neq \emptyset) \vee \\ (\text{mismatch}(\sigma'_i, s') \cap \text{mismatch}(\sigma'_j, s') \neq \emptyset)\end{gathered}$$

where $\text{mismatch}(x, y)$ returns the set of mismatched words covered by sub-segment x in segment y.

It is worth nothing that this last restriction may mark as incompatible two repair operators that, even though they work on the same mismatch, do not edit the same words in t. In those cases it is still advisable to forbid the application of the two repair operators since it is very likely that they work on the same region in t and their application interfere with one another. The following example illustrates this situation. Suppose the segment s' = *the size is around 100 cm* to be translated with the help of the translation unit (s, t) = (*the size is about 50 cm*, *el tamaño es de unos 50 cm*) whose target segment can be repaired with the two repair operators $o_1 = (\sigma_1, \sigma'_1, \tau_1, \tau'_1)$ = (*is about*, *is around*, *es de unos*, *está alrededor de*) and $o_2 = (\sigma_2, \sigma'_2, \tau_2, \tau'_2)$ = (*about 50*, *around 100*, *de unos 50*, *de unos 100*). Both operators share a mismatch (*about*) but do not edit the same words in t: o_1 edits the word *es* (which is replaced by *está*), introduces the word *alrededor* and removes (edits) the word *unos*; o_2 edits the word *50* and replaces it by *100*. The two operators can be applied at the same time if operator o_2 is applied first —the repaired target segment being $t^{\simeq}$ = *el tamaño está alrededor de 100 cm*— but not the other way around. Recall that the algorithm described in Section 2.2 assumes that repair operators can be applied independently of each other and the order in which they are applied does not affect the final result.

2.4 Experimental settings

To evaluate the potential of the fuzzy-match repair algorithm described in Section 2.2, we perform an oracle evaluation (Section 2.4.3) on three different language pairs: English–Spanish (en–es), Spanish–Portuguese (es–pt) and Spanish–French (es–fr). These language pairs are chosen to study how the method behaves when translating between closely-related languages (e.g. Spanish–Portuguese and Spanish–French) and when the languages involved in the translation are not so closely related (English–Spanish). In addition, of the two closely-related language pairs we use, Spanish and Portuguese are more alike than Spanish and French: Spanish and Portuguese are both

		en–es	es–pt	es–fr
TM	# TUs	196,294	150,567	149,479
	Avg. SL segment length	9.61	27.24	27.35
Test set	# SL segments	1993	1983	1983
	# SL words	40238	45334	46350
	Avg. SL segment length	20.19	22.67	21.73

Table 2.3: Data about the TMs and their corresponding source language (SL) segments and words from the test sets used in the experiments.

pro-drop, Ibero-Romance languages —they permit null-subject sentences— whereas French is a non-pro-drop Gallo-Romance language.

Our experiments use various MT systems as the black-box source of bilingual information. In particular, we first compare how well FMR performs using Apertium (Forcada et al., 2011), an RBMT system. Then, as a secondary experiment, we use two MT systems from distinct paradigms: Moses (Koehn et al., 2007), a phrase-based SMT system, and Nematus, an RNN-based sequence-to-sequence with attention NMT system, on the worst performing language pair (en–es) as a way to test if FMR backed by MT systems from distinct paradigms makes a difference on the outcome.

In this chapter, we evaluate the potential of the FMR method with fuzzy-match score thresholds of 60%, 70% and 80% with the aim of studying whether out method is more capable of repairing fuzzy matches above a given threshold. In this regard it is worth noting that professional translators usually set the fuzzy-match score threshold above 60% (Bowker, 2002). In later chapters, we extend our experiments even further to cover thresholds of 90%.

2.4.1 Corpora

As for the corpora used for the experiments, we use three translation memories, one per language pair, extracted from the DGT-TM 2015 multilingual translation memory;[4] each translation memory contains between 145,000 and 200,000 translation units. In addition, we also extract three test sets from the same source. Each test set contains around 2,000 parallel segments with source segments no longer than 100 words. The experiments consist of simulating the translation of each source segment in the test sets when using the TMs and using the corresponding target-language segment as a reference for evaluation. Table 2.3 provides additional information about the TMs and test sets used.

2.4.2 Machine translation systems

In the trade-off between adequacy (translations with the same meaning as the source) and fluency (translations that sound fluid or natural), NMT systems, tend

[4] `https://ec.europa.eu/jrc/en/language-technologies/dgt-translation-memory`

towards greater fluency, while sometimes producing fluent-sounding but semantically inadequate output (Bojar et al., 2016; Koehn and Knowles, 2017; Toral and Sánchez-Cartagena, 2017). In FMR, a full segment from the TM *may* provide the (fluent) backbone for the translation, while only containing a few sub-segment mismatches (such as numbers, names, and noun phrases). This naturally raises the question of how RBMT (which may provide greater adequacy for individual sub-segments) will compare to NMT systems (which may provide greater fluency) or SMT systems (which may fall between the two) for FMR.

The MT systems are used to build repair operators by translating sub-segments σ into the target language and to translate the segments in the test set for which a fuzzy match above the given threshold has not been found. They are used, more formally, to provide a single translation for each source segment;[5] and, they reflect the three paradigms mentioned: RBMT, SMT, and NMT. SMT and NMT systems are used here as a non-methodical way of validating ideas from the latest work in MT that point towards better-scoring translations which would, in turn, improve this work's worst performing FMR system based on English to Spanish translation with Apertium (Forcada et al., 2011). NMT systems submitted to the Second Conference on Machine Translation (WMT17), such as those built using the Nematus toolkit (Sennrich et al., 2017) have produced state-of-the-art performance across a number of language pairs (Bojar et al., 2017). NMT has been applied to other CAT applications, namely interactive translation prediction, (Knowles and Koehn, 2016; Wuebker et al., 2016) and neural approaches have been used for automatic post-editing (Pal et al., 2016; Junczys-Dowmunt and Grundkiewicz, 2016; Hokamp, 2017). Rarely, has NMT been used as the backing MT system for FMR.

Neural MT (Nematus)

We use the RNN-based sequence-to-sequence attention-based encoder-decoder Nematus (Sennrich et al., 2017) and the compatible AmuNMT decoder[6] (Junczys-Dowmunt et al., 2016).

Initial model training is done using Europarl v7 (Koehn, 2005) and News Commentary v10 data[7] (WMT13 training data for English–Spanish), with 2012 News Test data for validation. Following the domain adaptation method described in Luong and Manning (2015) and Freitag and Al-Onaizan (2016), we continue training on DGT-TM 2011–2013, with 3000 parallel sentences from the 2014 release as validation data.[8]

We use these training parameters: vocabulary of size 50,000, word embedding layer size of 500, hidden layer size of 1000, batch size of 80, Adadelta (Zeiler, 2012) as the optimizer, maximum sentence length of 50, and default learning rate of 0.0001. All

[5]That is, sets M and M' in lines 4 and 5 of Algorithm 1 are singletons in this case.

[6]Now part of Marian (`https://github.com/marian-nmt/marian`).

[7]`http://www.casmacat.eu/corpus/news-commentary.html`

[8]As the fuzzy-match repair scenario assumes that no sentences from that test set have been observed in the TM, we remove exact test set matches from DGT-TM training data.

other parameters are set to Nematus defaults. Data is pre-processed with the standard pre-processing scripts: tokenization, truecasing, and byte pair encoding (Sennrich et al., 2016) with 10,000 operations. We report scores with a beam size of 12.

Phrase-Based SMT (Moses)

We use Moses (Koehn et al., 2007) to train our phrase-based statistical SMT system using the same parallel text as the NMT model, with the addition of Common Crawl,[9] for phrase extraction. Europarl v7, News Commentary v10, monolingual News Crawl from 2007–2011, Spanish Gigaword v3 (Mendonça et al., 2011), and the target side of DGT-TM was used to build a 5-gram interpolated language model.

We use an operation sequence model (Durrani et al., 2015) with order 5, Good-Turing discounting (Good, 1953) of phrase translation probabilities, grouped phrase pairs according to counts, pruning of low-probability phrase pairs, and sparse features for target word insertion, deletion, and translation, and phrase length. Tuning is run on the same DGT-TM data used for NMT model validation.

Rule-Based MT (Apertium)

Apertium[10](Forcada et al., 2011), is a RBMT system, which performs translation using a pipeline of components: a morphological analyzer, a part of speech tagger, a lexical transfer module (which uses a bilingual dictionary to translate lexical forms from source language to target language), a structural transfer module (which performs syntactic operations), and a morphological generator.

For the primary multi-language pair (en–es, es–pt, and es–fr) experiments (Table 2.4), we use the language-pair packages `apertium-en-es`,[11] `apertium-es-pt`[12] and `apertium-fr-es`[13], respectively. In the secondary experiments, performed later in time, a more recent version of `apertium-en-es` is used.[14]

Machine translation performance

During development of fuzzy-match repair, it became clear that the original version of Apertium used performed worse than other, more state-of-the-art, MT systems. In order to better understand the difference in quality of the three systems, we present comparative information in Table 2.4 that provides the word error rate (WER) and BLEU scores attained by the initial, Apertium-only,[15] experiments and then the secondary, multi-MT system that uses a newer version of Apertium,[16] Nematus, and

[9]Available at `http://www.statmt.org/wmt13/translation-task.html`
[10]`https://www.apertium.org`
[11]SVN revision 64348.
[12]SVN revision 62539.
[13]SVN revision 62696.
[14]SVN revision 83165.
[15]en–es, SVN version 64348.
[16]en–es, SVN version 83165.

	Primary Experiments			Secondary Experiments		
	Apertium			Apertium	Moses	Nematus
	en–es	es–pt	es–fr	en–es		
WER	65.3%	47.4%	55.2%	60.8%	35.2%	36.8%
BLEU	18.6%	36.4%	24.7%	19.2%	57.2%	52.6%
OOV	2.6%	2.4%	2.4%	2.5%	2.5%	2.5%

Table 2.4: Multiple experiments performed that first focus on three language pairs (en–es, es–pt, and es–fr) with Apertium. Then, secondary experiments with several MT systems are done on the worst-performing language pair (en–es). Primary experiments use Apertium (`apertium-en-es`: SVN revision 64348, `apertium-es-pt`: SVN revision 62539, and `apertium-fr-es`: SVN revision 62696). Secondary experiments focus on the en–es language pair with three MT systems: a more recent version of Apertium (`apertium-en-es`: SVN revision 64348), Moses and Nematus. Additionally, out-of-vocabulary words (OOV) are provided for all experiments.

Moses. Additionally, we show the percentage of out-of-vocabulary words (OOV) for both sets of experiments.

As seen in Table 2.4, Apertium needs less post-editing in the case of two closely-related language pairs (es–pt and es–fr), unlike the case of English–Spanish. Gains, however, for the en–es language pair were achieved by using the SMT (Moses) and NMT (Nematus) systems instead. This is probably due to the fact that the SMT and NMT systems have more resources and are better statistical estimators whereas Apertium, which is a shallow-transfer RBMT system, aims at translating between closely related languages and may not have enough transfer rules to address most of the grammatical divergences between English and Spanish. Our results are favorable, even in the case of the lower-performing MT system from the primary experiment – Apertium. The newer version of Apertium used in the secondary experiment lowered the word-error rate by nearly 5%.

2.4.3 Oracle evaluation

The way to study the potential of our approach for fuzzy-match repair has been to generate, for each source segment s' in the test set, the set of all possible fuzzy-match repaired segments T and then use t', the translation of s', to choose the best one and evaluate its quality. Obviously, in a real setting t' would not be available and the best fuzzy-match repaired segment would need to be chosen using a method similar to those used for estimating the quality of machine translation output (see Chapter 3). What follows is a detailed explanation of the procedure we have followed with each source segment s' in the test set:

1. Retrieve the set of translation units U whose fuzzy-match score $\text{FMS}(s', s)$ is above the desired fuzzy-match threshold θ.

2. If there is no translation unit (s, t) so that $\text{FMS}(s', s) \geq \theta$, i.e. $U = \emptyset$, use machine translation to get a translation for s'. Otherwise use the TU $(s, t) \in U$ with the highest $\text{FMS}(s', s)$ and produce the set T with all possible fuzzy-match repaired segments.

3. Select the fuzzy-match repaired segment $t^{\simeq} \in T$ with the minimum edit distance to t'.

Once all the segments in the test set have been processed, the translations produced ($t^{\square}$) are evaluated by comparing them to the target segments in the test set and computing the error rate over the whole test set as follows:

$$\frac{\sum_{i=0}^{N} \text{ed}(t_i^{\square}, t'_i)}{\sum_{i=0}^{N} \max(|t_i^*|, |t'_i|)} \tag{2.1}$$

where $\text{ed}(x, y)$ returns the word-based edit distance between the segments x and y, N is the number of segments in the test set, and $|x|$ is the number of words of segment x. This way of computing the error rate resembles the way in which the fuzzy-match score is computed and accounts for the amount of operations needed to carry out a *translation job*.[17]

2.5 Results

2.5.1 Multiple language pairs with Apertium

Table 2.5 shows, for the three different language pairs on which we have evaluated our approach and for three different fuzzy-match score thresholds (FMT) —60%, 70% and 80%—, the error rate computed as described in Equation (2.1) when:

TM: the target segment in the translation unit with the highest fuzzy-match score is used as a translation, if available; otherwise, an empty translation is used, and therefore the error reflects the need to type the words in the reference translation.

MT: the same machine translation system used as SBI (Apertium) is used to translate the source segments in the test set.

FMR: the translation to be evaluated is obtained by applying the fuzzy-match repair algorithm described in Section 2.2 with the translation unit with the highest fuzzy-match score, if available, and selecting the fuzzy-match repaired segment that is closer to the reference translation (oracle); otherwise, the translation is produced using machine translation.

[17] For instance, OmegaT (`http://www.omegat.org`) computes the fuzzy-match score between s and s' as $1 - \frac{\text{ed}(s,s')}{\max(|s|,|s'|)}$.

FMT: 60%	**en–es**			**es–pt**			**es–fr**		
	TM	MT	FMR	TM	MT	FMR	TM	MT	FMR
Error (%)	55.0	65.3	36.5	56.5	47.4	31.3	56.4	55.2	34.7
Er. (%) on matches	20.1	65.3	17.9	22.5	47.4	17.0	20.3	55.2	16.5
# matches	1184	1993	1184	1221	1983	1221	1206	1983	1206
Avg. length	22.6	22.1	22.6	21.1	20.6	21.1	22.8	22.4	22.8

FMT: 70%	**en–es**			**es–pt**			**es–fr**		
	TM	MT	FMR	TM	MT	FMR	TM	MT	FMR
Error (%)	61.0	65.3	38.5	62.4	47.4	31.8	62.3	55.2	35.6
Er. (%) on matches	16.3	65.3	14.6	18.0	47.4	13.9	15.8	55.2	12.8
# matches	828	1993	828	777	1983	777	786	1983	786
Avg. length	22.4	22.1	22.5	20.8	20.6	21.1	22.6	22.4	22.6

FMT: 80%	**en–es**			**es–pt**			**es–fr**		
	TM	MT	FMR	TM	MT	FMR	TM	MT	FMR
Error (%)	69.7	65.3	42.6	70.1	47.4	33.8	69.5	55.2	38.2
Er. (%) on matches	13.1	65.3	11.9	15.3	47.4	11.9	12.2	55.2	9.7
# matches	660	1993	660	641	1983	641	649	1983	649
Avg. length	22.3	22.2	22.4	20.8	20.6	21.1	22.5	22.4	22.8

Table 2.5: For the three different language pairs considered in our evaluation and for three different means of translation —translation memory (TM), machine translation (MT) and fuzzy-match repair (FMR)— and fuzzy-match score thresholds (FMT), the table gives the error rate over the whole test set, the error rate over the segments in the test set for which a match above the given threshold is found in the translation memory, the amount of these segments (# matches) and the average length of the target segments produced.

Two error rates are reported, one computed on the whole test set and another computed only on the set of segments for which a TU with a fuzzy-match score above the given threshold is found (error on matches). The former provides and indication of the actual translation effort a translator would made to translate the source segments in the test set. The latter provides an indication of the performance of our method for fuzzy-match repair (FMR) without the interference of whole-segment machine translation, since it focuses only on those segments for which there is a translation unit to repair. This allows to directly compare FMR performance to that of using the target segment in the best TU without any repair (TM). In addition, the number of source segments for which a match is found in the translation memory and the average length of the translations produced are provided.

Error rate is measured at the word level and over the entire test set. The error rate over the test set grows with the fuzzy-match score threshold (FMT). This happens because the greater this threshold is, the less source segments can be translated using

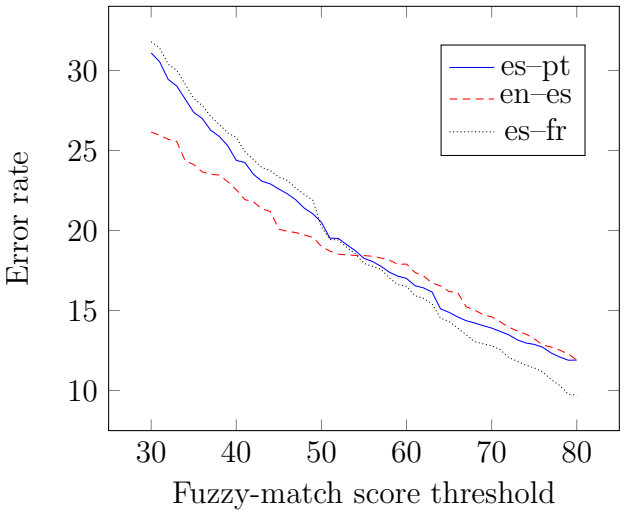


Figure 2.2: Error rate when using Apertium and three language pairs for evaluation over the segments in the test set for which a match above the fuzzy-match score threshold is found in the translation memory.

fuzzy-match repair and, as a consequence, the amount of segments that are translated with Apertium grows. If we focus only on those segments that can be translated by means of FMR, we can see that the error rate decreases as the threshold grows; Figure 2.2 show how the error rate from the initial multi-language experiment with Apertium on matches behaves as a function of the fuzzy-match score threshold. This is the expected behavior because as the threshold grows the amount of words to repair decreases.

2.5.2 Apertium, Moses, and Nematus on the English to Spanish language pair

The initial experiments in Table 2.5 show how well FMR performs using the best-case (oracle) candidate fuzzy-match repaired segment along with a rule-based MT system, Apertium, and three language pairs. This section takes a deeper dive into other MT paradigms with a focus on English to Spanish. Focus is placed on the en–es experiments for two reasons. First, because English-to-Spanish MT systems can benefit more from the plethora of data than other language pairs, resulting in higher quality, in-domain, translations. Secondly, the need to better the results gotten from the initial experiments with the en–es language pair was evident.

Table 2.6 reports the results using three MT paradigms: Apertium (RBMT), Moses (SMT) and Nematus (NMT). Test data is also drawn from the 2015 DGT-TM data set. While the initial experiments are unable to better the translation memory score by more than 3 points, using other MT paradigms: statistical (Moses) and neural (Nematus), with the same data nearly maintains a 3 point spread for all experiments. Thus, a good part of the difference in performance between the three language pairs

FMT: 60%	NMT			SMT			RBMT		
	TM	MT	FMR	TM	MT	FMR	TM	MT	FMR
Error (%)	55.0	36.8	27.1	55.0	35.2	26.7	55.0	60.8	37.5
Er. (%) on matches	20.8	36.8	15.2	20.8	35.2	15.6	20.8	60.8	18.5
# matches	1184	1993	1184	1184	1993	1184	1184	1993	1184
Avg. length	22.6	22.1	22.6	22.6	22.1	22.6	22.6	22.1	22.6

FMT: 70%	NMT			SMT			RBMT		
	TM	MT	FMR	TM	MT	FMR	TM	MT	FMR
Error (%)	61.0	36.8	20.8	61.0	35.2	27.3	61.0	60.8	39.6
Er. (%) on matches	16.7	36.8	12.0	16.7	35.2	12.7	16.7	60.8	15.0
# matches	828	1993	828	828	1993	828	828	1993	828
Avg. length	22.4	22.1	22.5	22.4	22.1	22.5	22.4	22.1	22.5

FMT: 80%	NMT			SMT			RBMT		
	TM	MT	FMR	TM	MT	FMR	TM	MT	FMR
Error (%)	69.7	36.8	28.5	69.7	35.2	27.9	69.7	60.8	43.7
Er. (%) on matches	13.4	36.8	9.4	13.4	35.2	10.4	13.4	60.8	12.2
# matches	660	1993	660	660	1993	660	660	1993	660
Avg. length	22.3	22.2	22.4	22.3	22.2	22.4	22.3	22.2	22.4

Table 2.6: Secondary experiments for three different MT paradigms: neural (NMT), statistical (SMT), and rule-based (RBMT). We evaluate three different means of translating English to Spanish: translation memory (TM), machine translation (MT) and fuzzy-match repair (FMR) using various fuzzy-match score thresholds (FMT). The table gives the error rate over the whole test set, the error rate over the segments in the test set for which a match above the given threshold is found in the translation memory, the amount of these segments (# matches) and the average length of the target segments produced. Apertium, the RBMT system, scores better here than the primary experiments in Table 2.5 due to the use of a more up-to-date version (en–es: SVN 85136); yet, Nematus (NMT) and Moses (SMT) still outperform it.

can be attributed to the performance of the MT system; if we pay attention to the performance of FMR when the evaluation only focuses on those segments for which a match has been found, we can see that the scores reported are quite similar for all language pairs, even though this does not happen in the case of the MT scores reported, i.e. our method for fuzzy-match repair appears to be quite robust to MT errors.

From the results in Table 2.6, it is clear that an error rate below that of using the target segment in the best translation unit (TM) and below that of using machine translation (MT) can be achieved by all MT systems. Additionally, we can conclude that for the worst-performing language pair (en–es), an FMR system backed with an SMT or NMT system outperforms the RBMT system by nearly 20 error-rate points (3 points when using matched segments alone) for all fuzzy-match thresholds.

The difference between the best candidate fuzzy-match repaired segment produced (oracle) by our algorithm and the MT-produced translations reveals that our FMR approach is quite robust to MT errors. This is because for a repair operator to be successfully built, τ, the translation of the sub-segment σ of s, must appear as a contiguous text sub-segment in t, the translation proposal being repaired. This acts as a quality filter that makes our FMR approach not to use for repairing sub-segments for which the SBI does not match the TM. Obviously, this quality check cannot be performed on τ', the translation of the sub-segment σ' in s' aligned with σ. However, it seems that having a quality check on τ helps to ensure that most of the repair operators built are of good quality.

Interestingly, despite having worse error rates on full-sentence translations, the NMT system actually outperforms the SMT system as a source of bilingual information for FMR on the subsets of data for which TM matches were found. The SMT system's high performance on *all* source segments ("Error (%)" row in Table 2.6) is due to the higher quality translations of the entire segment when a fuzzy-match is not found in the TM. Poorer performance of the RBMT system can be attributed to two main factors: 1) Apertium is a shallow transfer system designed for the translation between closely-related languages like Spanish and Portuguese; and, 2) Apertium generally requires wider context for high-quality translations due to its use of transfer rules. All of the MT systems outperform the no-repair TM baseline error rate (in which we simply computed the error rate for the best fuzzy-matches from the TM, without any repairs). Table 2.7 depicts an example from the data.

The NMT system outperforms SMT and RBMT for FMR on matches and is more often successful at repairing segments. While the SMT and NMT display stronger performance over the RBMT system and, thus, require more attention here, we include some details of the RBMT performance as well to better understand how all systems compare. At the 60% FMT level, the SMT system successfully produced repairs for 788 segments, while the NMT system successfully produced repairs for 957 segments,

s':src	promote human resources training;
t':ref	promover la formación de los recursos humanos;
s:TM	promote human resources development;
t:TM	fomentar el desarrollo de los recursos humanos;
RBMT	fomentar el desarrollo de los los recursos humanos que entrenan;
SMT	fomentar la formación de recursos humanos;
NMT	fomentar los recursos humanos;

Table 2.7: Example segments, showing the best fuzzy-match repaired segments for three MT systems.

and the RBMT system for 825 (out of a possible 1184 segments).[18] Since those are two distinct sets of segments, we cannot directly compare the error rate. We first examine the intersection of those sets (the subset of segments for which the three systems successfully performed FMR).

A total of 754 segments were successfully repaired by all systems. There were 34 segments which the SMT system repaired and the NMT system did not, and 203 segments for which the opposite was true. Of the 754 segments repaired by both, 212 were repaired better by the NMT system, 139 were repaired better by the SMT system, and 403 were repaired equally well by the two MT systems (all in terms of error rate). Computing the error rate over this shared 754 segment set, we find that the error rate of the SMT system (14.4%) is quite close to that of the NMT system (14.3%). This suggests that the NMT system's ability to repair more sentences plays a major role in its better FMR results.

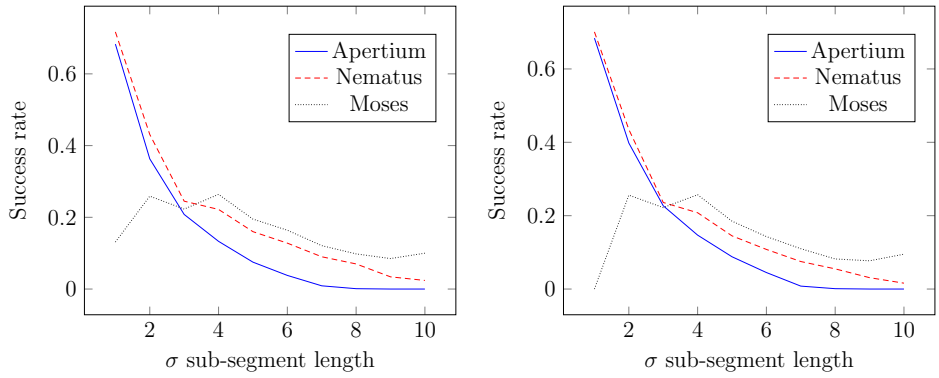
The NMT system produced an average of 1.92 possible repaired segments per source segment (standard deviation: 1.29, maximum: 9). Using the SMT system, an average of 1.68 possible repaired segments were produced per source segment (standard deviation: 0.92, maximum: 7). In a real-world setting, the system would need to choose between more repaired options for the NMT system than the SMT system.

2.5.3 A closer look at repair operator creation and performance

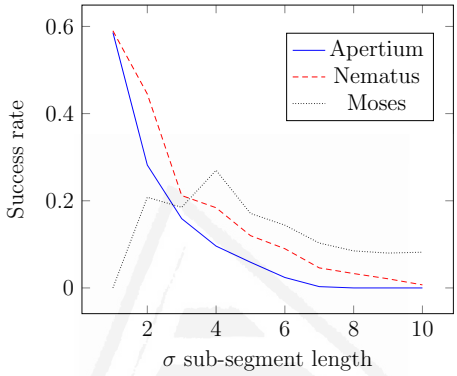
With respect to the process of building repair operators, and since the performance of the machine translation system differs between the language pairs, it is worth studying how *successful* FMR is when building repair operators. Figure 2.3 plots the success rate when building repair operators as a function of the length of the source sub-segments σ for a fuzzy-match score threshold of 60%, 70% and 80%. As can be seen, success rates for different fuzzy-match thresholds behave very similarly. A repair operator is successful when the translation of the sub-segment σ of s is found in t, that is, when the machine translation system and the proposed translation unit exactly agree on the translation of a source sub-segment: this acts as a safety feature, as repair is not attempted when this agreement is absent.

As seen in Figure 2.3, the longer the sub-segments the harder it is that the translation obtained from the MT system is found in t. This behavior is present in all the language pairs and is more pronounced when the translation involves non-related language pairs (en–es); hence, our analysis focuses on those σ sub-segments. Nonetheless, the average length of σ in the repair operators used to build the fuzzy-match repaired segment chosen by the oracle when the fuzzy-match score threshold is set to 80% is around 2.8 words for en–es, 3.7 for es–fr and 4.7 for es–pt.

[18] Professional translators typically use higher fuzzy-match thresholds, but we select 60% in this section to provide the greatest amount of data for direct comparison of repairs.



(a) σ success rate at 60% FMT **(b)** σ success rate at 70% FMT



(c) σ success rate at 80% FMT

Figure 2.3: Success rates when building repair operators using three different MT system: Apertium (RBMT), Nematus (NMT), and Moses (SMT) for English to Spanish and for fuzzy-match score thresholds (FMT) of 60%, 70%, and 80%. Success rates are provided as a function of the number of words in the source sub-segments σ being translated when building repair operators.

We examine the sub-segment translations produced by the RBMT (Apertium), NMT (Nematus) and SMT (Moses) systems to gain insight about what allows the NMT system to repair more segments and produce more candidate fuzzy-match repaired segments per segment in those cases where the length of σ is between 2 and 3. On the other hand, we also examine σ using Moses for lengths of 4 to 10. Notably, Moses performs well over more σ sub-segments than the other MT systems.

First, we look at the lengths of the translations of the sub-segments. Particularly, we focus on the two higher performing systems from Table 2.6: SMT and NMT. For both the SMT and NMT systems, the translations tend to be longer than the source sub-segments (64% of the time for the SMT system and 58% of the time for the NMT system). The NMT system produces translations that are shorter than the source 23% of the time, while the SMT system does so 18% of the time. They also differ in the range of lengths; the NMT system has more extreme values, sometimes producing no translation at all and even occasionally producing translations more than three times

the length of the longest source segment. On average, the SMT translations are 2.37 tokens longer than the source sub-segments (standard deviation of nearly 3.95). The NMT translations average 2.71 tokens longer than the source, with a much greater standard deviation of 10.26. The very long NMT translations may be more likely to be discarded (due to not matching), but the very short translations may be easier to find matches for in the TM target side, contributing to the larger number of sentences the NMT system repairs. As an aside, RBMT generally fairs well for sub-segments where words are to be replaced (e.g. short segments of 1 to 3 words where adjectives are to be replaced), much like the SMT system; however, SMT performs equally well on deletions and insertions.

We also note a qualitative difference: the SMT system often add additional punctuation that was not included in the source, as well as determiners. These spurious tokens could make it harder to find matches in the TM target segments, resulting in fewer opportunities for fuzzy-match repair. This could be caused by the language model providing higher scores to the phrases that include those tokens.

The sub-segments that are most often translated for FMR are not complete segments; rather, they are sub-segments that can be taken from any point in the original segment. This poses a potential challenge for any MT system which is trained on full segments. In the case of the SMT system, the language model may prefer sub-segment translations that include, for example, determiners or additional punctuation, as we observed. NMT systems have been observed to do a poor job of handling data that differs from the original training data, often producing fluent-seeming text that has little to do with the source. While this mismatch does not seem to have had a strong negative impact on the overall results, it is possible that the results could still improve if the sub-segmental input were better matched to the training data.

2.6 Actual complexity of the repair algorithm

In Section 2.2.1, we saw that the worst-case complexity of the repair algorithm is $O(2^n)$, where n is the number of repair operators available for a TU (s, t) and segment to be translated s'. Usually, not all repair operators are compatible, and cannot therefore be applied together to produced a fuzzy-match repaired segment (see Section 2.3); as a result, the actual complexity of the algorithm is drastically reduced. Figure 2.4 shows, for `en-es`, the ratio of fuzzy-match repair segments produced to those that would be produced in the worst case (2^n) as a function of n. As can be seen, this ratio decreases as n increases and is quite close to zero when the amount of repair operators is high. This accounts for the practicality of the approach.

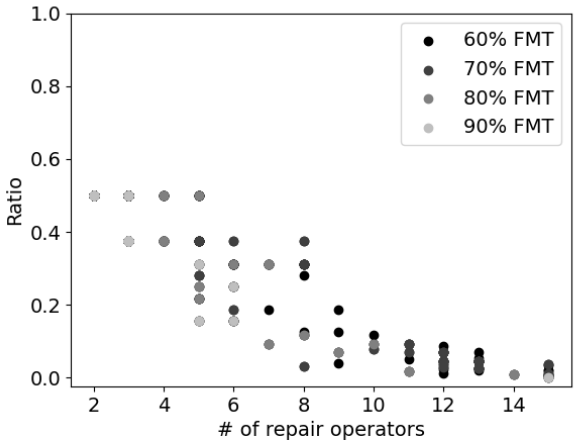


Figure 2.4: For `en-es`, ratio of fuzzy-match repair segments produced to those that would be produced in the worst case.

2.7 Concluding remarks

In this chapter, we have introduced our fuzzy-match repair approach. We first covered the general FMR approach, capable of using any available source of bilingual information, such as an MT system, to generate a set of fuzzy-match repaired segments from the source segment to be translated s' and a TU (s,t) from the TM. In order to show the details of our approach, a review of the five-step repair process is presented. First, our approach locates mismatches between the source segment to be translated s' and the source segment s from the TU to be repaired. We align those mismatches using a well known phrase extraction algorithm (Koehn, 2010, section 5.2.3) and then, in the second step, translate the source-side sub-segments gotten from those mismatches. In the third and fourth steps, we pair sub-segment translations together to form repair operators. Finally, in the fifth step, we apply the mutually compatible repair operators to generate fuzzy-match repaired segments.

After walking through the five steps of our FMR approach, to give the reader a better understanding through the use of a running example, we dive deep into the FMR algorithm in Section 2.2. Particularly, we provide an algorithmic description of how repair operators are generated and how fuzzy-match repaired segments are obtained by applying them. To illustrate its application, we provide an in-depth analysis of repair creation drawn from another running example which shows the robustness of our approach.

Repair compatibility, first described in Section 2.3, is clearly defined to describe how our approach avoids situations where two repair operators are incompatible and cannot be applied together to produce the same fuzzy-match repaired segment. The compatibility of repair operators leads way to a thorough analysis of the actual complexity of our algorithm. Additionally, an in-depth analysis is performed in Section

2.5.3 that shows that our repair method generally performs well on repairs between two and five words long. The actual compatibility analysis combined with the repair operator analysis provide the necessary details to show how the FMR approach would perform.

Our experimentation illustrates the FMR approach's performance with three different language pairs (en–es, es–pt, es–fr) using a well-known TM (DGT-TM). The oracle evaluation we have conducted reveals the potential of our FMR approach. For the three language pairs experimented on, we consistently improve the translation quality produced —both with respect to raw machine translation and unrepaired fuzzy matches— despite the initial MT system (Apertium) performance as compared to the more state-of-the-art systems (Moses and Nematus).

Chapter 3

Quality estimation for fuzzy-match repair

In Chapter 2, we presented and evaluated a fuzzy-match repair algorithm capable of generating a set of fuzzy-match repaired segments from a TU and the segment to be translated. We performed an oracle evaluation to see if the FMR algorithm was able to generate a fuzzy-match repaired translation close to the reference translation. In this chapter, we describe how the best candidate fuzzy-match repaired segment may be chosen by estimating the quality of the fuzzy-match repaired segments produced using a regressor. Our evaluation conducted on the three different language pairs used in Chapter 2 shows that the candidate fuzzy-match repaired segment with the best predicted quality is a good approximation to the best (oracle) candidate produced and is consistently closer to reference translations than either machine-translated segments or unrepaired fuzzy matches. In addition, a single quality estimation model trained on a mix of data from all the languages performs well on any of the languages used in our experiments.

3.1 Introduction

In this chapter, we use the FMR approach described in Chapter 2 to generate a set of candidate fuzzy-match repaired segments and then estimate the quality of each fuzzy-match repaired segment produced to select the best one. The method for estimating the quality of candidate fuzzy-match repaired segments produced, inspired by the work on sentence-level MT quality estimation (QE) (Blatz et al., 2004; Specia et al., 2009), uses regressors trained on a combination of black-box, system-independent features and glass-box, system-dependent features. In our experiments we use *extremely randomized trees* (Geurts et al., 2006) as regressors and evaluate our approach using various fuzzy-match score thresholds on the three different language pairs tested in Chapter 2. The results show that the set of features we propose are informative enough to obtain

regressors that allow us to select candidate fuzzy-match repaired segments close to the best (oracle) one among those produced by our FMR algorithm. The selected candidates are consistently closer to the reference translations in our test sets than both the non-repaired target segments (t) and the translations obtained by translating whole source segments (s') using the MT system used as the SBI. Moreover, the best regressors are most of the times obtained when they are trained on all language pairs together, which signifies that the values of the features proposed and the regressors learned are language-independent.

The rest of the chapter is organized as follows. Section 3.2 describes the QE approach and the features we propose to automatically select the best fuzzy-match repaired segment. Sections 3.3 and 3.4 then discuss, respectively, the experimental settings and the results obtained when evaluating the success of our approach. Finally, Section 3.5 ends with concluding remarks.

3.2 Quality estimation of candidate repaired segments

In the context of MT, sentence-level quality estimation (QE) methods (Blatz et al., 2004; Specia et al., 2009) have been developed during the last two decades to avoid bothering professional translators with low-quality translations, to choose among a set of different translations produced by different MT systems for a given source segment, or to estimate the effort to post-edit a given MT output. Quality is usually measured in terms of post-editing time, in terms of the amount of edit operations needed to turn the translation into an adequate translation, or using other related metrics (Specia, 2011; Bojar et al., 2014).

MT QE techniques can easily be adapted for estimating the quality of the different candidate fuzzy-match repaired segments produced for the source segment to be translated and pick the best one. There are mainly two different approaches to achieve this: the use of a binary classifiers and the use of a regressor. The former can be used to select the best translation on a pairwise comparison basis (Avramidis, 2013); the latter can be used to rank the set of candidate fuzzy-match repaired segments. In this chapter we follow this last approach; in particular, we use, after preliminary experiments with linear and support-vector regressors (Basak et al., 2007), extremely randomized trees (Geurts et al., 2006) for regression.

In the following, we describe the features used by the regressor. In particular, we describe two sets of features: one using information readily available to CAT tools (black-box features, Section 3.2.1), and a second one that exploits information from the inner workings or the FMR algorithm used to generate the set of candidate fuzzy-match repaired segments (glass-box features, Section 3.2.2).

3.2.1 Black-box (system-independent) features

The following features, some of which are borrowed from the set of 17 baseline features used in the shared tasks on MT QE at the WMT MT contests and implemented in QuEst++ (Specia et al., 2015),[1] use only information already present in the source segment to be translated s', in the candidate fuzzy-match repaired segment $t^{\simeq}$ or in the translation unit being repaired (s, t):

BB1 Number of tokens in the source segment s'.

BB2 Number of tokens in the candidate fuzzy-match repaired segment $t^{\simeq}$.

BB3 Ratio of the number of tokens in $t^{\simeq}$ to the number of tokens in s'.

BB4 Number of punctuation marks in s'.

BB5 Number of punctuation marks in $t^{\simeq}$.

BB6 Ratio of the number of punctuation marks in $t^{\simeq}$ to the number of punctuation marks in s'.

BB7 Number of digits in s'.

BB8 Number of digits in $t^{\simeq}$.

BB9 Ratio of the number of digits in $t^{\simeq}$ to the number of digits in s'.

BB10 Source fuzzy-match score: $\mathrm{FMS}(s, s')$.

BB11 Target fuzzy-match score: $\mathrm{FMS}(t, t^{\simeq})$.

BB12 Ratio of the source fuzzy-match score to the target fuzzy-match score:

$$\frac{\mathrm{FMS}(s, s')}{\mathrm{FMS}(t, t^{\simeq})}.$$

BB13 Source sub-segment-level fuzzy-match score: $\mathrm{MMS}_{\mathrm{seg}}(s, s')$. It is computed by using the matching and mismatching sub-segments as the building blocks when computing the edit distance; this implies a monotonic segmentation of (s, s') performed on the basis of the word alignments obtained as a by-product of the edit distance.[2] It measures the ratio of mismatched sub-segments to the total number of sub-segments.

BB14 Target sub-segment-level fuzzy-match score: $\mathrm{MMS}_{\mathrm{seg}}(t, t^{\simeq})$.

[1] `https://github.com/ghpaetzold/questplusplus`

[2] This is similar to the n-gram tuples used in n-gram based statistical MT (Marino et al., 2006, Sec. 2.1)

BB15 Ratio of the source sub-segment-level fuzzy-match score to the target sub-segment-level fuzzy-match score:

$$\frac{\mathrm{MMS}_{\mathrm{seg}}(s, s')}{\mathrm{MMS}_{\mathrm{seg}}(t, t^{\simeq})}.$$

It is worth noting that features BB1, BB4, BB7, BB10 and BB13 do not pay attention to the candidate fuzzy-match repaired segment but to the source segment to be translated (s') and its relation to the source segment s in the TU being repaired. Nevertheless, the learning algorithm can rely on them to specialise the trees to be used for regression, that is, to use different sub-trees for regression depending on their value.

3.2.2 Glass-box (system-dependent) features

The features described next make use of information about the inner workings of the FMR algorithm used to generate candidate fuzzy-match repaired segments; more precisely, they capture information about the repair operators used and their form:

GB1 Ratio of word positions in $t^{\simeq}$ covered by at least one repair operator to the number of words in $t^{\simeq}$:

$$\frac{|\{j : \exists \tau' \wedge t^{\simeq}_j \text{ is part of } \tau'\}|}{|t^{\simeq}|},$$

where $t^{\simeq}_j$ is the j-th word of $t^{\simeq}$.

GB2 Sum of the length of the repair operators used to build $t^{\simeq}$ divided by the length of $t^{\simeq}$:

$$\frac{\sum_{i=1}^{N} |\tau'_i|}{|t^{\simeq}|},$$

where N is the number of repair operators used to get $t^{\simeq}$. Notice that a word in $t^{\simeq}$ may be covered by more than one repair operator and is counted as many times as it is covered.

GB3 Ratio of word positions common to t and $t^{\simeq}$ that are covered by at least one repair operator to the number of words positions common to t and $t^{\simeq}$; i.e. average overlap:

$$\frac{|\{j : \exists \tau \wedge t_j \text{ is part of } \tau \wedge j \in \mathrm{match}(t, t^{\simeq})\}|}{|\mathrm{match}(t, t^{\simeq})|},$$

where function $\mathrm{match}(t, t^{\simeq})$ returns a set with the word positions common to t and $t^{\simeq}$.

GB4 Sum of the length of the overlapping sub-segments of the repair operators used to build $t^{\simeq}$ divided by the length of $t^{\simeq}$:

$$\frac{\sum_{i=1}^{N} |\mathrm{match}(t, \tau'_i)|}{|t^{\simeq}|}.$$

The overlapping sub-segments of a repair operator are those containing words common to t and $t^{\simeq}$. As above, a word common to t and $t^{\simeq}$ may be covered by more than one repair operator and is counted as many times as it is part of an overlapping sub-segment.

GB5 Ratio of words in s' covered by at least one σ' used to build a repair operator to the number of words in s':

$$\frac{|\{j : \exists \sigma' \wedge s'_j \text{ is part of } \sigma'\}|}{|s'|},$$

where s'_j is the j-th word of s'.

GB6 Sum of the length of the σ's used to build a repair operator divided by the length of s':

$$\frac{\sum_{i=1}^{N} |\sigma'_i|}{|s'|}.$$

Similarly to GB2, a word in s' may be covered by more than one σ' and is counted as many times as it is covered by a σ'.

GB7 Ratio of matched source word positions that are covered by at least one σ used to build a repair operator to the number of matched source words positions; i.e. average overlap in the source language:

$$\frac{|\{j : \exists \sigma \wedge s_j \text{ is part of } \sigma \wedge j \in \text{match}(s, s')\}|}{|\text{match}(s, s')|}.$$

GB8 Sum of the length of the overlapping sub-segments of the (σ, σ') used to build the repair operators used to get $t^{\simeq}$ divided by the length of s':

$$\frac{\sum_{i=1}^{N} |\text{match}(s, \sigma'_i)|}{|s'|}.$$

The overlapping sub-segments here are those containing words common to s and s'. As above, a word that is part of the matching between s and s' may be covered by more than one (σ, σ') and is counted as many times as it is part of an overlapping sub-segment.

GB9 Mean target context per repair operator:

$$\frac{\sum_{i=1}^{N} |\text{LCS}(\tau_i, \tau'_i)|}{\sum_{i=1}^{N} \min(|\tau_i|, |\tau'_i|)},$$

where $\text{LCS}(x, y)$ is the longest common sub-sequence to x and y.

GB10 Mean source context per repair operator:

$$\frac{\sum_{i=1}^{N} |\text{LCS}(\sigma_i, \sigma'_i)|}{\sum_{i=1}^{N} \min(|\sigma_i|, |\sigma'_i|)}.$$

GB11 Measure of how evenly distributed are the mismatched target-language sub-segments in the repair operators used to build the fuzzy-match repaired segment $t^{\simeq}$:

$$\frac{\sum_{k=1}^{N} |\tau_k| \prod_{j=1}^{n_k} \frac{|m_j|-1}{|\tau_k|-2n_k+1}}{\sum_{k=1}^{N} |\tau_k|},$$

where n_k is the number of matching sub-segments in τ_k and m_i is the i-th matching sub-string. When computing n_k it must be taken into account that every matched sub-segment must consist of contiguous words both in τ and τ'. The closer the value of the feature is to 1, the more evenly distributed the mismatched sub-segments are.

GB12 Measure of how evenly distributed are the mismatched source-language sub-segments in the repair operators used to build the fuzzy-match repaired segment $t^{\simeq}$:

$$\frac{\sum_{k=1}^{N} |\sigma_k| \prod_{j=1}^{n_k} \frac{|m_j|-1}{|\sigma_k|-2n_k+1}}{\sum_{k=1}^{N} |\sigma_k|},$$

where n_k is the number of matching sub-segments in σ_k and m_i is the i-th matching sub-string. As above, when computing n_k it must be taken into account that every matched sub-segment must consists of contiguous words both in σ and σ', as in GB11. The closer the value of the features is to 1, the more evenly distributed the mismatched sub-segments are.

GB13 Number of repair operators used to get $t^{\simeq}$ divided by the number of mismatched words in s.

GB14 Number of repair operators used to get $t^{\simeq}$ divided by the number of mismatched sub-segments in s (sequences of contiguous word positions).

GB15 Number of repair operators used to get $t^{\simeq}$.

GB16 Ratio of *grounded* repair operators to the number of repair operators used to get $t^{\simeq}$. A repair operator is considered to be grounded if at least one word at each end of σ is matched.

GB17 Binary feature to flag whether all the repair operators used to get $t^{\simeq}$ are grounded or not.

3.3 Experimental settings

The experimental settings are nearly the same as those in Chapter 2. The difference between them lies in the manner that the corpora is used. In this chapter, we split the corpora used in Chapter 2 in a different way to provide training samples for the QE regressor. In what follows, we describe the resources used, the way the training samples used to train the QE component of the method were generated, the regressor

used and how it is trained, and the metrics used to evaluate the performance of our FMR approach with QE.

3.3.1 Resources

We use the DGT-TM 2015 parallel corpus[3] for the three language pairs (en–es, es–pt, and es–fr). For each fuzzy-match-score threshold (FMT) ϕ used in our experiments, we split the DGT TM 2015 corpus into three sets: one for training the regressor used for QE (training), another to select the configuration of the regressor to be finally used (see next section; development) and a third one for performance evaluation (testing). This splitting can be formally described as follows. Let P be the parallel corpus, J_ϕ a randomly drawn subset of P, a *translation job*, for the FMT ϕ, and $M_\phi = P - J_\phi$ the translation memory to be used. J_ϕ is build by randomly selecting parallel segments (s', t') from P so that the final set meets the following condition:

$$\forall (s', t') \in J_\phi, \exists (s, t) \in M_\phi : \mathrm{FMS}(s, s') \geq \phi.$$

After building J_ϕ, we remove those segments for which, with the MT system used as a SBI, it is not possible to build at least one repair operator. The resulting set is divided into three disjoint subsets, $J_{\phi,\mathrm{train}}$, $J_{\phi,\mathrm{dev}}$ and $J_{\phi,\mathrm{test}}$, such that $J_\phi = J_{\phi,\mathrm{train}} \cup J_{\phi,\mathrm{dev}} \cup J_{\phi,\mathrm{test}}$. Table 3.1 shows, for each FMT used in the experiments and for each language pair, the amount of segments to be translated and the average number of candidate repaired segments $t^{\simeq}$ per segment to be translated s'. These sets were obtained as explained above from a parallel corpus P with 196,294 segments for `en-es`, 150,567 for `es-pt` and 149,479 for `es-fr`. For convenience, the TM to be used is the same for all FMTs, $M_{60\%}$, that is the one obtained for $\phi = 60\%$.

Like in Chapter 2, the figures in Table 3.1 show that the average number of candidate fuzzy-match repaired segments per segment to be translated goes down as the FMT grows because the percentage of words to be repaired, and; therefore, the number of repair operators, is reduced.

For each $(s', t') \in J_{\phi,\mathrm{train}}$ a set of training samples is generated as follows. First the repair algorithm described in Section 2.2 is run to generate the set of candidate fuzzy-match repaired segments $\{t^{\simeq}\}$ for the translation of s' using the TU with the highest FMS, that is, the TU $(s, t) \in M_{60\%} : (\nexists (s'', t'') \in M_{60\%} : \mathrm{FMS}(s', s) < \mathrm{FMS}(s', s''))$.[4] Then, from each candidate fuzzy-match repaired segment $t^{\simeq}$ a training sample is generated by computing the features described in Section 3.2 and the error rate to be predicted. This error rate is similar to the one presented in Section 2.4.3 with the exception that this error rate is calculated for FMR proposals at the segment level and the other one is calculated for several segments at the document level. It is depicted as:

$$\xi(t^{\simeq}, t') = \frac{\mathrm{ed}(t^{\simeq}, t')}{\max(|t^{\simeq}|, |t'|)} \tag{3.1}$$

[3] `https://ec.europa.eu/jrc/en/language-technologies/dgt-translation-memory`

[4] If there is more than one TU meeting this condition, one is selected at random.

FMT (ϕ)	Language pair	Training		Development		Test	
		$N_{s'}$	$N_{t\simeq}$	$N_{s'}$	$N_{t\simeq}$	$N_{s'}$	$N_{t\simeq}$
60%	en-es	3,249	17.6	238	14.8	460	32.2
	es-pt	3,194	14.9	220	15.5	1,084	11.3
	es-fr	2,930	14.0	120	14.6	1,070	16.4
70%	en-es	2,289	9.8	157	16.6	355	25.5
	es-pt	1,981	10.8	134	18.5	678	11.7
	es-fr	2,376	12.4	94	17.1	716	17.2
80%	en-es	1,250	6.1	50	8.3	194	5.5
	es-pt	1,267	7.6	83	9.0	554	9.5
	es-fr	1,812	5.7	122	10.1	588	13.2
90%	en-es	239	6.9	18	8.3	56	3.3
	es-pt	483	7.2	33	13.1	229	6.8
	es-fr	762	5.4	50	4.2	236	6.6

Table 3.1: For each fuzzy-match score threshold (FMT; ϕ) and language pair, number of segments to be translated ($N_{s'}$) and average number of candidate fuzzy-match repaired segments per segment to be translated ($N_{t\simeq}$) in the training, development and test sets.

Language pair	FMT (ϕ)			
	60%	70%	80%	90%
en-es	57,816	22,333	7,653	1,661
es-pt	47,675	21,305	9,589	3,492
es-fr	41,124	29,432	10,351	4,140

Table 3.2: For the training set, number of samples for the different fuzzy-match-score thresholds (FMT) used in the experiments.

where $\text{ed}(x, y)$ returns the word-based edit distance (Wagner and Fischer, 1974) between the segments x and y. Table 3.2 shows, for the different fuzzy-match score thresholds used, the amount of training samples used to train the regressor used for QE.

The corpora used to build the training, development and test sets may contain parallel segments that are *free* translations of one another, and this may be introducing noise affecting the regressor's performance. This problem was already detected in the DGT-TM 2015 parallel corpus by Esplà-Gomis et al. (2015), who proposed a simple filtering method to discard this noise. This method removes the candidate repaired segments obtained from a segment to be translated s', reference translation t' and TU to be repaired (s, t) for which $|\text{FMS}(s, s') - \text{FMS}(t, t')| > 0.05$. This filtering is based on the assumption that the number of words that differ in both pairs of segments should be similar for both languages. In Section 3.4, we report results when using both filtered and non-filtered corpora to study the effect of this noise on the regressor's performance. By applying this filtering the amount of segments that are discarded is around 22%.

Regarding the MT system used for FMR, we use the same version of Apertium and language pair packages (`apertium-en-es`, `apertium-es-pt`, and `apertium-fr-es`) from Chapter 2.

3.3.2 Regressor

In Section 3.4, we report the results obtained when using *extremely randomized trees* (ERT) (Geurts et al., 2006) because it is the regressor that performed best when evaluated on the development corpus on a set of preliminary experiments in which we also tried with linear regression and support vector regression (SVR) (Basak et al., 2007). In the case of SVR, we performed a 3-fold cross-validation grid search to find the optimum hyper-parameter values of the Gaussian radial-basis function kernel we used and tried with different feature selection methods, such as recursive feature elimination (Guyon et al., 2002), chi-square, and the Gini importance computed over extremely randomized trees (Louppe et al., 2013).[5]

Extremely randomized trees is a tree-based ensemble method for classification and regression (Geurts et al., 2006). At each internal tree node the best feature is determined from a subset of features selected at random from the whole set of features whereas the cut-off point is selected fully at random. In our implementation the best feature is selected according to the *Gini importance*, also know as *mean decrease in impurity* (Breiman et al., 1984, ch. 4) (see Section 3.4.1 for more information). When ERTs are used for regression, the prediction is computed as the average of the output of all the trees in the ensemble.

The main hyper-parameters controlling the learning process of ERT are the size of the subset of features randomly selected (N) and the amount of trees in the ensemble (M). If $N = 1$, the feature to use in each internal node is selected fully at random and the method builds completely randomized trees. If N equals the total amount of features (F), randomization only happens in the selection of the cut-off point. In our experiments we tried with $N = \sqrt{F}$, $N = \log_2(F)$ and $N = F$; the best results on the development set were obtain with $N = F$, like in Geurts and Louppe (2011). In regards to the amount of trees in the ensemble, we tried with 10, 100, 200, 500, 800 and $1,000$ and; overall, the best results on the development set were obtained for $M = 100$. The difference between using 100 or more trees is negligible. With respect to the rest of parameters, we used the default ones in the ERT implementation of scikit-learn.[6]

Finally, due to the randomization of trees, different training executions resulted in regressors with small differences in the output they provide. The results we report in the following section are those obtained with the ERT performing best on the development set out of ten different training executions.

[5]SVR implementation used: version 0.19 of scikit-learn, `http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html`.

[6]Version 0.19, `http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html`.

3.3.3 Evaluation

We measure the performance of our QE approach for FMR with the following metrics:

- Error rate over the whole test set is as performed in Chapter 2:

$$\frac{\sum_{i=1}^{N} \mathrm{ed}(t_i^{\square}, t'_i)}{\sum_{i=1}^{N} \max(|t_i^{\square}|, |t'_i|)},$$

  where t'_i is the gold standard translation in the test set for the source segment s'_i and $t_i^{\square}$ is the translation being evaluated; $t_i^{\square}$ may be the (unrepaired) target segment t_i, a candidate fuzzy-match repaired segment $t_i^{\simeq}$ produced by our repair algorithm or the translation of s_i produced by the MT system.

- *Mean absolute error* (MAE) of the error rate predicted by the ERT regressor on each candidate fuzzy-match repaired segment in the test set. MAE evaluates the ability of the regressor to predict error rates at the segment level. It is computed as

$$\frac{\sum_{i=1}^{N} \sum_{j=1}^{M_i} |\xi(t_{ij}^{\simeq}, t'_i) - \hat{\xi}(t_{ij}^{\simeq}, t'_i)|}{\sum_{i=1}^{N} M_i},$$

  where $\xi(\cdot)$ and $\hat{\xi}(\cdot)$ are the predicted error rate (see Equation (3.1)) and the error rate to be predicted, respectively, and M_i is the number of candidate fuzzy-match repaired segments produced for source segment s_i.

- *Success rate* (SR) of the ERT regressor used to select the best candidate fuzzy-match repaired segment. It is computed by comparing the amount of edit operations that are saved when editing the candidate fuzzy-match repaired segment $t_i^{\simeq}$ with the lowest predicted error rate, instead of the (unrepaired) target segment t_i of the TU being repaired, to the amount of edit operations saved when editing the best possible (oracle) fuzzy-match repaired segment $t_i^{\star}$:

$$\frac{\sum_{i=1}^{N} \frac{|\mathrm{ed}(t_i, t'_i) - \mathrm{ed}(t_i^{\simeq}, t'_i)|}{|\mathrm{ed}(t_i, t'_i) - \mathrm{ed}(t_i^{\star}, t'_i)|}}{N}$$

  where t'_i is the gold standard translation in the test set for the source segment s'_i. This metric shows how good $t_i^{\simeq}$ is when compared to the best possible repaired segment (oracle) produced by the repair algorithm described in Section 2.2: the numerator is the actual change in the edit distance when replacing t_i with the repaired version $t_i^{\simeq}$ and the denominator is the change in edit distance that would be produced if t_i was replaced with the *oracle* (best possible) repair t_i^{*}.

FMT	Non-filtered corpora			Filtered corpora		
	TM	MT	Oracle	TM	MT	Oracle
English–Spanish						
60%	27.16	57.61	23.24	22.13	56.73	17.91
70%	22.83	56.21	19.32	18.53	56.43	14.73
80%	17.31	61.60	14.31	13.27	56.00	9.71
90%	11.63	66.84	5.85	11.21	58.91	4.78
Spanish–Portuguese						
60%	22.08	40.56	15.86	17.29	36.86	10.07
70%	18.20	40.00	13.41	13.63	36.30	8.03
80%	15.29	39.67	11.13	10.62	35.55	5.52
90%	10.42	44.87	7.69	7.24	40.23	3.97
Spanish–French						
60%	19.78	49.66	15.34	15.27	48.59	10.81
70%	15.97	49.75	12.52	12.17	48.54	8.71
80%	12.48	49.56	9.38	9.84	48.51	6.56
90%	7.71	51.42	5.36	6.34	50.72	3.75

Table 3.3: For the non-filtered and filtered corpora, error rate (%) for the target segment t in the TU (s, t) being repaired (TM), for the translation produced by the MT system for the whole source segment s' (MT) and for the best possible fuzzy-match repaired segment $t^\star$ (Oracle).

3.4 Results and discussion

First, we repeat the oracle evaluation performed in Chapter 2 to show that the FMR algorithm described in Section 2.2 provides similar results with the splitting of the corpora we are using in this chapter. Table 3.3 shows the error rate, computed as described above, for the oracle evaluation. Additionally, we provide error rates computed on corpora filtered using the method by Esplà-Gomis et al. (2015) and described in Section 3.3.1.

The results reported in Table 3.3 show the ability of our approach to produce good candidate fuzzy-match repaired segments. It is worth noting the lower error rates reported for all three approaches across the table when evaluated on the filtered corpora, which is consistent with the way in which the corpora are filtered (see end of Section 3.3.1).

Next, we evaluate how well our QE approach performs when selecting the best candidate fuzzy-match repaired segment among the whole set of candidates produced. In order to determine the best configuration to train the regressor to be used to estimate the quality of fuzzy-match repaired segments, we have tried with the following setups: (1) training different ERT regressors for different FMTs and language pairs, (2) training one regressor per language pair regardless of the FMT, and (3) training a

FMT	Non-filtered corpora					Filtered corpora				
	TM	ERT	Oracle	SR	MAE	TM	ERT	Oracle	SR	MAE
English–Spanish										
60%	27.16	25.71	23.24	0.37	0.06	22.13	19.38	17.91	0.65	0.04
70%	22.83	21.58	19.32	0.36	0.06	18.53	15.71	14.73	0.74	0.04
80%	17.31	16.14	14.31	0.39	0.07	13.27	10.91	9.71	0.66	0.05
90%	11.63	7.55	5.85	0.71	0.07	11.21	5.77	4.78	0.85	0.02
Spanish–Portuguese										
60%	22.08	17.74	15.86	0.70	0.11	17.29	11.84	10.07	0.76	0.06
70%	18.20	15.35	13.41	0.59	0.09	13.63	9.77	8.03	0.69	0.04
80%	15.29	13.54	11.13	0.42	0.08	10.62	6.79	5.52	0.75	0.03
90%	10.42	8.73	7.69	0.62	0.08	7.24	4.56	3.97	0.82	0.03
Spanish–French										
60%	19.78	17.31	15.34	0.56	0.08	15.27	12.45	10.81	0.63	0.05
70%	15.97	14.44	12.52	0.44	0.07	12.17	10.37	8.71	0.52	0.05
80%	12.48	10.99	9.38	0.48	0.05	9.84	7.44	6.56	0.73	0.03
90%	7.71	6.50	5.36	0.52	0.06	6.34	4.57	3.75	0.68	0.03

Table 3.4: For both the non-filtered and filtered corpora, error rate (%) for the target segment in the TU being repaired (TM), for the fuzzy-match repaired segment with the lowest predicted error rate (ERT) and for the best possible one (Oracle). Success rates (SR) and mean absolute errors (MAE) are also provided. A different ERT regressor was trained for each FMT and language pair.

single regressor for all language pairs and FMTs. As above, we provide results when the ERT is trained and evaluated both on filtered and non-filtered corpora.

Table 3.4 shows the error rates obtained for the target segment in the TU being repaired (TM) —the one with the highest fuzzy-match above the FMT—, for the fuzzy-match repaired segment with the lowest predicted error rate (ERT) and for the best possible fuzzy-match repaired segment produced by the FMR algorithm (Oracle). The table also provides the success rate (SR) obtained by comparing the error rate of the oracle and the error rate of the fuzzy-match repaired segment with the lowest predicted error rate, and the mean absolute error (MAE) of the ERT regressor. For each FMT and language pair a different ERT regressor was trained.

The results show that the use of the ERT regressor to select, for a given source segment and TU, the candidate fuzzy-match repaired segment with the lowest predicted error rate performs better on the filtered corpora, where *noisy* translation units have been removed from the training, development and test sets, than on the non-filtered corpora. The difference in performance is noteworthy for FMT below 90%, especially in the case of English–Spanish. Additionally, the success rate, that is the proportion of edit operations saved when editing the selected candidate fuzzy-match repaired segment over the number of edit operations saved when editing the best possible fuzzy-match repaired segment, increases as the FMT grows; for 60% FMT the success rates on the

FMT	Non-filtered corpora					Filtered corpora				
	TM	ERT	Oracle	SR	MAE	TM	ERT	Oracle	SR	MAE
English–Spanish										
60%	27.16	25.71	23.24	0.37	0.06	22.13	19.38	17.91	0.65	0.04
70%	22.83	21.30	19.32	0.44	0.06	18.53	15.68	14.73	0.75	0.05
80%	17.31	16.07	14.31	0.41	0.07	13.27	10.54	9.71	0.77	0.04
90%	11.63	6.29	5.85	0.92	0.07	11.21	5.00	4.78	0.96	0.02
Spanish–Portuguese										
60%	22.08	17.74	15.86	0.70	0.11	17.29	11.84	10.07	0.76	0.06
70%	18.20	15.14	13.41	0.64	0.09	13.63	9.54	8.03	0.73	0.04
80%	15.29	12.75	11.13	0.61	0.09	10.62	6.61	5.52	0.79	0.04
90%	10.42	8.66	7.69	0.64	0.09	7.24	4.49	3.97	0.84	0.03
Spanish–French										
60%	19.78	17.31	15.34	0.56	0.08	15.27	12.45	10.81	0.63	0.05
70%	15.97	14.19	12.52	0.52	0.07	12.17	9.90	8.71	0.66	0.05
80%	12.48	10.82	9.38	0.53	0.06	9.84	7.71	6.56	0.65	0.03
90%	7.71	6.43	5.36	0.55	0.06	6.34	4.46	3.75	0.73	0.02

Table 3.5: Error rate for the target segment in the TU being repaired (MT), for the fuzzy-match repaired segment with the lowest predicted error rate (ERT) and for the best possible one (Oracle). Success rates (SR) and mean absolute errors (MAE) are also reported. A single ERT regressor was trained on 60% FMT for each language pair.

filtered corpora are around 0.68, whereas for 90% FMT they scale up to 0.78, surpassing 0.80 for English–Spanish and Spanish–Portuguese. Our QE method is clearly better at ranking fuzzy-match repaired segments when the amount of mismatched sub-segments is small; which is the typical scenario where TM-based CAT is used.[7]

If we pay attention to the performance of the ERT regressor when evaluated as such, we can see that the MAEs reported are below 0.10 in all cases but one (es–pt, 60% FMT, non-filtered corpora), and that, for the filtered corpora they are around 0.05. This accounts for the high informativeness of the features defined in Section 3.2. We do not report the root mean square error because it shows a similar trend. It is worth noting that MAE is computed at the sample-level and, as a result, all candidate fuzzy-match repaired segments are equally judged, regardless of their length.

The results in Table 3.4 were obtained when using a different ERT regressor per FMT and language pair. In order to study how dependent is the ERT regressor on the FMT used for training, we repeated the experiments reported in Table 3.4 but using a single regressor per language pair. In particular, we used the regressor trained on samples obtained from TUs for which the fuzzy-match is above the 60% FMT; the results are reported in Table 3.5.

[7]Fuzzy-matches are seldom used for FMS below 70%.

FMT	Non-filtered corpora					Filtered corpora				
	TM	**ERT**	**Oracle**	**SR**	**MAE**	**TM**	**ERT**	**Oracle**	**SR**	**MAE**
English–Spanish										
60%	27.16	25.64	23.24	0.39	0.06	22.13	19.72	17.91	0.57	0.04
70%	22.83	21.20	19.32	0.47	0.06	18.53	15.77	14.73	0.73	0.04
80%	17.31	15.97	14.31	0.45	0.08	13.27	10.40	9.71	0.81	0.05
90%	11.63	6.36	5.85	0.91	0.10	11.21	5.00	4.78	0.96	0.02
Spanish–Portuguese										
60%	22.08	17.69	15.86	0.71	0.11	17.29	11.42	10.07	0.81	0.05
70%	18.20	15.23	13.41	0.62	0.09	13.63	9.11	8.03	0.81	0.04
80%	15.29	12.70	11.13	0.62	0.07	10.62	6.34	5.52	0.84	0.04
90%	10.42	8.83	7.69	0.58	0.07	7.24	4.55	3.97	0.82	0.04
Spanish–French										
60%	19.78	17.18	15.34	0.59	0.08	15.27	12.02	10.81	0.73	0.05
70%	15.97	14.34	12.52	0.47	0.07	12.17	9.84	8.71	0.67	0.04
80%	12.48	11.17	9.38	0.42	0.05	9.84	7.60	6.56	0.68	0.03
90%	7.71	6.27	5.36	0.61	0.06	6.34	4.47	3.75	0.72	0.02

Table 3.6: Error rate for the target segment in the TU being repaired (MT), for the fuzzy-match repaired segment with the lowest predicted error rate (ERT) and for the best possible one (Oracle). Success rates (SR) and mean absolute errors (MAE) are also shown. The same ERT regressor trained on 60% FMT is used for all language pairs.

From the comparison of the results in tables 3.4 and 3.5 we can conclude that using a single ERT regressor per language pair is the best option. The results obtained with a single regressor are better than those obtained with a regressor per FMT, especially for the 80% and 90% FMTs. This is probably due to the fact that the amount of training samples for 80% FMT and 90% FMT is an order of magnitude lower as compared to 60% FMT (see Table 3.2), and as a result the ERT regressor was not learning adequately.

None of the features used and listed in Section 3.2 are language-dependent, although the distribution of their values and how informative they are may differ from one language to another. To ascertain whether or not the ERTs informed by these features are also language-independent, we repeated the experiments reported in Table 3.5 but training an ERT regressor on the samples obtained from TUs for which the fuzzy-match is above the 60% FMT for *all language pairs together*. Table 3.6 reports the results obtained; it is worth noting that the same ERT regressor is used regardless of the FMT and language pair, and that the amount of training samples is 146,615.

The results in Table 3.6 are quite similar to those in Table 3.5, where a different ERT is used per language pair. For some language pairs and FMTs results improve slightly, while for others they worsen slightly. This allow us to conclude that, given the small difference in the success rates reported in both tables it is advisable to use a single regressor trained on 60% FMT for all languages together, at least for languages

which are related in some sense (such as the four Western Indo-European languages in these experiments).

Next, we study how fast the learning process converges to see if the amount of training samples used for training is enough to learn the best possible ERT models. Figure 3.1 plots the learning curve computed when training ERT models for regression for the English–Spanish language pair and for all languages together using a 60% FMT, both on filtered and non-filtered corpora. The learning curves show cross-validation (dashed line) and training (solid lines) scores of the ERT models as a function of the number of training samples. For cross validation we performed 10 iterations of *shuffle* split: in each iteration 20% of the training set was randomly selected as a validation set. These scores correspond to the coefficient of determination (Rao, 1973, ch. 4): the best possible score is 1.0 and it can be negative if the model gets arbitrarily worse. Our ERT models for 60% FMT converge to values of the coefficient of determination above 85%, some of them even above 95%, which indicates that our ERT models are learning and that the amount of samples used for training is adequate for the task.

3.4.1 Discussion on the informativeness of features

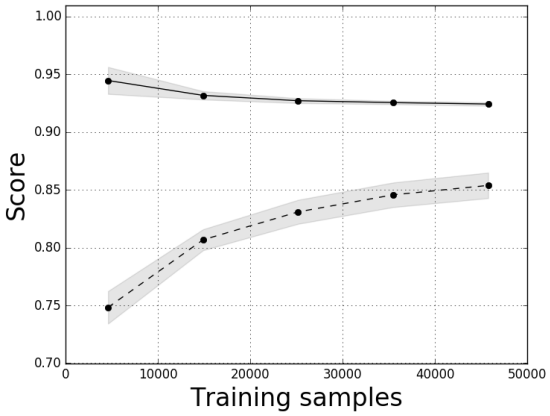
To better understand how our ERT models work, we present an analysis of the informativeness of the features used to create them. As mentioned above, the ERT implementation we have used uses the *Gini importance*, also know as *mean decrease in impurity* (Breiman et al., 1984, ch. 4), to decide the splitting of the nodes while building the trees.[8] The mean decrease in impurity is directly related to the *purity* of a tree node. A node in a tree is considered to be pure if its probability for the training samples reaching it is 1; that is, if all samples reaching the node are in the same class (in the case of classification) or are close to a single target value (in the case of regression).

Our analysis is based on the Gini importance computed as the total decrease in node impurity, weighted by the probability of reaching that node (which is approximated by the proportion of samples reaching that node) and averaged over all trees in the ensemble.[9] We used Gini importance instead of other methods, such as *permutation importance*, which has been shown to have "better statistical properties" (Strobl et al., 2008), because it is less computationally expensive (Breiman and Cutler, 2019) and is the measure used by the scikit-learn implementation of ERT we have used.

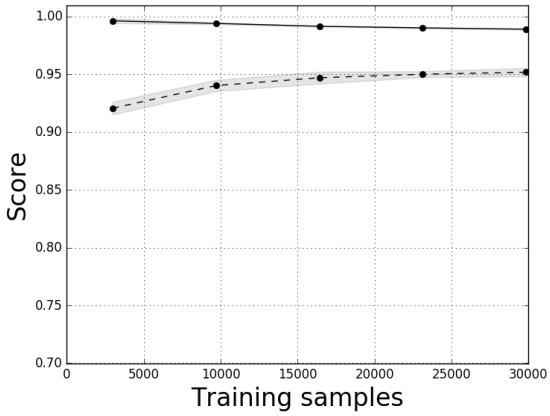
Since the learning curve of the ERT models that use all language pairs (see Figure 3.1) is similar and in some cases better than the ones computed for individual language pairs, we analyse the informativeness of the features obtained from the ERT models for all language pairs trained on filtered and non-filtered corpora with a 60%

[8]This way of deciding on the splitting of nodes differs from that of the original paper describing ERT (Geurts et al., 2006).

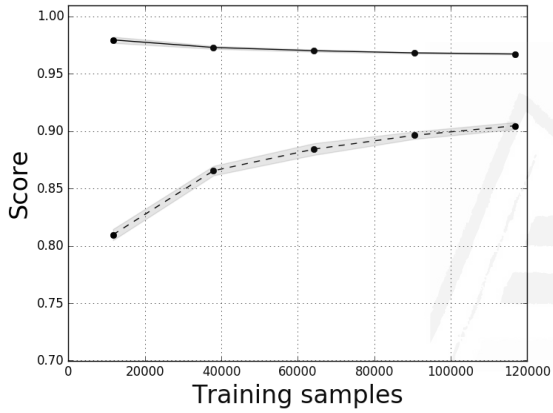
[9]For more information about the computation of the Gini importance we refer the reader to the work by Louppe et al. (2013).



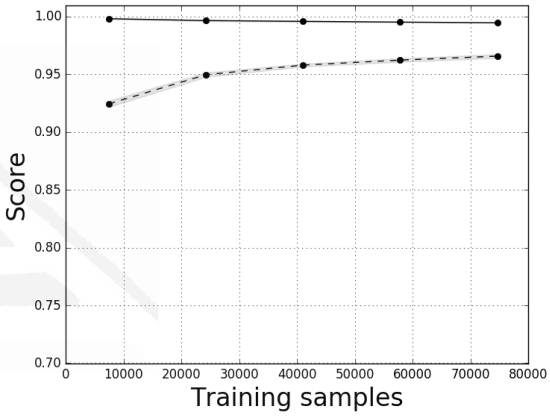
(a) English–Spanish, non-filtered corpora



(b) English–Spanish, filtered corpora



(c) All language pairs, non-filtered corpora



(d) All language pairs, filtered corpora

Figure 3.1: Learning curve for training ERT models for English–Spanish (((a), (b)) and for all language pairs together ((c), (d)) when using non-filtered ((a), (c)) and filtered ((b), (d)) corpora. The solid line is the learning curve computed over the training corpus, whereas the dashed line corresponds to the cross-validation learning curve.

FMT. Figure 3.2 shows the Gini importance of the top 12 features for the ERT models aforementioned. The rest of features, although less relevant, still have a Gini importance above 0.0, which means that they help the ERT regressor; in fact, the removal of any of them affect the performance of the resulting ERT regressor.

As Figure 3.2 shows, the top 12 features are all black-box features that do not take advantage of the information about the inner workings of the repair algorithm. Of these top 12 features, BB10 scores unusually high while BB7 and BB1 are the best performing features which are closer to the median of the Gini importance. The value of BB10, that is, the fuzzy match between s and s', may be working as a means for specializing the trees to be used for regression so that the ones used for small values of BB10 are different from those used for larger ones. In this regard, it is worth noting that the value of BB10 is the same for all training samples obtained from the segment

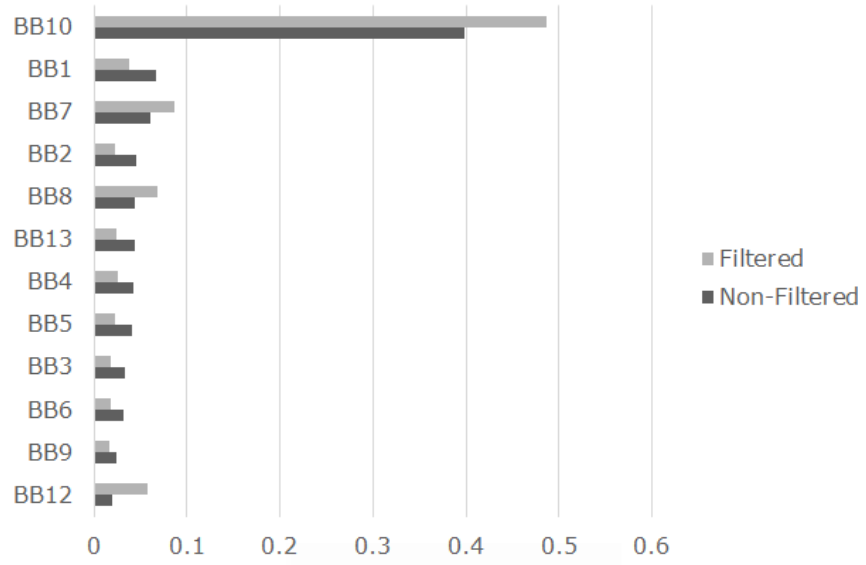


Figure 3.2: Gini importance for the top 12 features computed over ERT regressors trained for all language pairs with a 60% FMT and on filtered and non-filtered corpora. Features are ordered according to the Gini importance on non-filtered corpora.

to be translated s' and the TU (s, t) to be repaired. Other high-scoring features like BB1 and BB7 are in a similar situation because they count tokens and digits from the source segment s' to be translated, and provide a nice check of the validity of the repair operators used (the number of digits should be invariant).

3.5 Concluding remarks

In this chapter we have described a quality estimation technique for fuzzy-match repair that estimates the quality of fuzzy-match repaired segments produced in order to select one for final translation. For selecting the best fuzzy-match repaired segment, we propose a set of features and train a tree-based regressor to predict the amount of edit operations needed to convert each candidate fuzzy-match repaired segment into an adequate translation of s'. The candidate fuzzy-match repaired segment with the lowest predicted amount of edit operations needed is the one selected as best.

We have extensively evaluated the performance of this approach on three different language pairs, namely English–Spanish, Spanish–Portuguese and Spanish–French, with different fuzzy-match score thresholds (FMT), and using raw (non-filtered) corpora and (filtered) corpora from which *noisy* translation units have been removed. The best results are obtained on the filtered corpora and with regressors trained on training samples obtained using a 60% FMT. We have also evaluated the performance when the regressor is trained on a mix of training samples from all language pairs and then tested on each different language pair. The last evaluation shows that not only are the

features that we propose language-independent; but also, the regressor based on them is.

The performance of the quality estimation approach used to select the best candidate fuzzy-match repaired segment depends on the similarity between the segment to be translated s' and the source segment s in the TU (s, t) being repaired. The more similar they are (the greater the FMT), the more successful it is. For a 90% FMT, the QE method selects the best possible candidate fuzzy-match repair segment generated by the repair algorithm most of the times, resulting, on average, in a success rate on filtered corpora above 0.83. This means that the use of the selected candidate fuzzy-match repaired segment allows to save, on average, 83% of the edit operations that would have been saved if the best possible (oracle) candidate segment would have been chosen. If the candidate repaired segments were selected at random the saving of edit operations would be, on average, 44%.

As for the features used by the regressors, we have proposed a set of features made up of black-box (system-independent) and glass-box (system-dependent) features; black-box features are easier to compute as they do not exploit any information about the repair operators used to generate the candidate fuzzy-match repaired segments. Black-box features are found to be more informative that glass-box ones, although all of them are useful to some extent. This result suggests that the QE method used to select the best fuzzy-match repaired segment could be used to rank candidate fuzzy-match repaired segments produced by other FMR approaches like those described in Section 1.4.

Chapter 4

Selecting an MT system for fuzzy-match repair

This chapter deals with a problem that could occur when using various MT systems: how to select the optimum MT system given a source segment. Here, we attempt to save the CAT tool user, or industry-level expert, resources and time by deploying a system that selects, on a segment basis, the MT system to be used for fuzzy-match repair without actually running the MT system.

In this chapter we initially diverge our focus from FMR to MT in order to show the importance of having a method that selects an MT system before translating a source segment. After finding that we are able to select an MT system with relatively high accuracy (nearly 70%), we then illustrate how the selection of an MT system from a set of several MT systems improves the original FMR approach presented in Chapter 2.

4.1 Introduction

While systems using the NMT paradigm achieve the highest scores on recent shared tasks (Bojar et al., 2017), SMT and RBMT systems may provide better results for individual segments due to different internal workings. Additionally, in some cases, research has shown that professional translators and CAT-tool users prefer SMT over NMT, despite NMT's higher performance using conventional scoring standards like BLEU (Arenas, 2013). Previous work by Bentivogli et al. (2016) has shown how different MT paradigms perform on certain types of input. More specifically, NMT has been found to perform worse than SMT due to specific anomalies in the text like punctuation or named entities (Koehn and Knowles, 2017). With that in mind, it may be premature to completely abandon "old" methods in favor of "new" ones. Alternatives that combine MT systems could achieve optimum results; particularly, if the system combination method can take advantage of the different paradigm strengths.

In Chapter 2, we saw how multiple MT systems score differently among various fuzzy-match thresholds and language pairs. In this chapter, we introduce a predictive system called SelecT that aims to derive optimum translations by choosing an MT system for each source segment before attempting to translate it. Relying on a single MT system could be costly because, as shown in a previous research (Rosti et al., 2007), the single system's accuracy may be directly related to its linguistic knowledge. That linguistic knowledge can be better captured by using several MT paradigms that take advantage of word, phrase, or even character-level features. As a way of addressing those differences, professional translators use high-quality systems like Google Translate[1] or Bing Translate[2] that provide state-of-the-art translations based on multiple paradigms like NMT and SMT. The drawback, though, is that professionals may have several MT systems at their disposal and they typically run their systems at the document level which makes sentence-by-sentence processing nearly impossible. Contrarily, SelecT determines the MT system to be used for each source segment based on a predictive mechanism that is described in this chapter.

SelecT is beneficial because it prevents an external system from running several MT systems per segment; thus, saving translators time and money when dealing with high-grade, often times costly, MT systems. This chapter explores SelecT when used as a selection device between the three paradigms introduced in Chapter 2: Nematus (Sennrich et al., 2017), the NMT system; Moses (Koehn et al., 2007), the SMT system; and Apertium (Forcada et al., 2011), the RBMT system.

The SelecT system presented in this chapter uses a FastText[3] classifier (Joulin et al., 2017) that predicts which MT system would perform best given a source segment. The FastText classifier performs better in our experiments when compared to two other classifiers: a recurrent neural network and a logistic regression classifier. SelecT's classifier is trained on a corpus of sub-segments labeled with MT system (Apertium, Moses, or Nematus) producing the best translation according to the BLEU scores computed by comparing each system's output to a reference translation. It is important to note that *any* other scoring mechanism could be used to train SelecT, it does not need to be BLEU.

SelecT works in an agnostic, or black-box, manner; so, although it is typically trained on entire segments, it can provide MT system hypotheses at the sub-segment (or fuzzy-match repair) level as well. Our experiments show that using SelecT with FMR increases performance for both entire segments and sub-segments, despite being trained only on entire segments. The FMR system presented in Chapter 2 behaves as a black-box also and accepts translations from any MT system; thus, it can be easily integrated with SelecT for higher gains even though most (more than 80%) of the requested translations from our FMR system are sub-segments.

[1] `https://translate.google.com`

[2] `https://www.bing.com/translator`

[3] `https://github.com/facebookresearch/fastText/`

The next section explores previous work on system combination and quality assessment of MT systems. We then dive deeper into the motivation and methodology of selecting an MT system before translating. Afterwards, in Section 4.4, we explain the three MT systems we use – making note of the major differences between their translation approaches. Next, in Section 4.5, experimentation is described by first explaining the settings and; secondly, showing how the best classifier is chosen from the three different classifiers we have tested. After establishing the best performing classifier, in Section 4.6 we show how well SelecT performs when integrated with the FMR system described in Chapter 2. Lastly, we provide concluding remarks for the chapter and discuss future work.

4.2 Related work on system combination

There are a plethora of papers about system combination in MT. Most research, like the majority of combination systems from WMT17 (Bojar et al., 2017), chose to combine MT systems following different approaches. Combination systems can be broken down into several main paradigms. Some of the approaches may use a pipeline approach where one MT system's output is used as input to another one, as is done by Niehues et al. (2016). But typically, MT system combinations falls into three main categories: 1) *consensus*, 2) *selection*, and 3) consensus and selection together (i.e. *in-between*).

Consensus approaches are the most common approaches and typically create a unique translation composed of several words or phrases using translations from various MT systems. They are powerful approaches and in previous work (Peter et al., 2017) they have outperformed translations from a single MT system in terms of translation quality. Most of the more recent consensus methods, like Di Gangi et al. (2017)'s work, rely on a standard system combination framework called Jane (Freitag et al., 2014). Jane uses a confusion matrix to find the shortest path of word reordering among parallel translations from distinct MT systems.

Other consensus approaches, like the work by Heafield and Lavie (2010), use a Multi-Engine Machine Translation (MEMT) scheme. Their work combines translations by first aligning them with METEOR (Lavie and Agarwal, 2007) and then using a beam search to reorder sub-segments into a final translation. Additionally, Peter et al. (2016) use a consensus approach that takes advantage of the differences between SMT and NMT by combining sub-segments using METEOR.

Selection methods keep translations intact by choosing what they consider to be the most appropriate translation from several distinct MT system's translation candidates. As seen in previous work (Nomoto, 2004; Zwarts and Dras, 2008), selection methods do not modify the final chosen translation in any way. Nomoto (2004)'s selection method uses a voting system based on a perplexity score. The voting system is considered a confidence model that will assign a perplexity score for each translation according

to how well the language models statistically fare on an input sentence. Their work selects a sentence based on a score, much like SelecT that uses a BLEU score for training its classifier; however, while their work requires translations of the sentence first, our system does not.

Somewhere in between the consensus and selection approaches there are systems, like Macherey and Och (2007)'s system, which also uses a MEMT system, that first select a subset of promising translations and then combine them to produce a single output. In-between approaches are less common and tend to be more complex. For example, Garmash and Monz (2016) attempt to induce parameters for the various MT systems in an ensemble system based on the best-scoring translations. González-Rubio and Casacuberta (2015) use an in-between approach bringing together consensus and selection by using a predictive, risk-like, model for marking and translating sub-segments based on a performance metric.

We focus on system combination approaches that use a selection method because it is the method that SelecT uses. Zwarts and Dras (2008)'s selection approach, for example, is similar to SelecT in some ways. They propose a classifier approach that selects the best translation from a MEMT system. But, unlike Nomoto (2004), attention is focused more on syntactic differences between the source and target sentences. Furthermore, their objective, similar to that of our work, is to produce translations that score well on a quality assessment metric. The main difference between their work and ours is that they build a custom MT system (a modified SMT system) using knowledge gained from the source and target segment. Then, they compare a baseline SMT system to the modified one. Our work uses a black-box MT approach that does not modify the baseline MT systems in any way. Additionally, SelecT is a unique selection system because it focuses on selecting an MT system in a *a-prior* manner without actually using it to translate the source segment until chosen.

Due to its *a-prior* method that uses sentence-level features to train binary maximum entropy classifiers, Sánchez-Martínez (2011)'s work can be considered the closest to ours. His work uses features from a parse tree to train a classifier for each MT system that estimates the probability of that system being the best one to translate a given source. The main differences between his approach and SelecT lie in how the classification is done. SelecT reduces the overall implementation complexity by reducing the amount of features while at the same time increasing classification probability by testing 3 distinct *baseline* classifiers (see Section 4.3) based on the latest prediction architectures such as neural networks. Other classification details also differ. For example, while Sánchez-Martínez (2011) extracts features at the sentence level like SelecT, his method can be considered complex because of requirements on a statistical parser, quality estimation techniques, and other deeply contextual knowledge. The three classifiers in SelecT's system either extract the features in an unsupervised manner, as is the case for most neural networks, or are based on a standard representation such as *bag-of-words*. Apart from feature extraction, differences between SelecT and Sánchez-Martínez (2011)'s work are found in how the classifiers are configured. His work use one

classifier per MT system; SelecT uses one classifier with multiple (MT-engine) labels in an attempt to discover the single best classifier.

To our knowledge, our work is novel because it uses state-of-the art classifiers and MT systems. SelecT determines which MT system to use *before* translating the source segment given. Also, while details from this chapter report that BLEU scores are used to determine the best MT system for each source segment, any score could be used; thus making SelecT an agnostic classifying mechanism for modular translation tasks. The next section covers SelecT's classifier in more detail.

4.3 Classification approach

SelecT is designed to use any number of classifiers in an agnostic manner during development. The idea is to first train and test each classifier on the entire training corpus during the development stage to determine the optimum one; then, use that classifier on the unseen test data. In this section we first describe the classifiers used in our experiments but leave the explanation of experimental settings for each classifier to Section 4.5. After that, we show how SelecT is used as a classifier for fuzzy match repair.

Our work intrinsically compares 3 open-source MT systems using the three predictive models below:

1. **Bi-Directional Recurrent Neural Network**: A recurrent neural network (RNN) is an artificial neural network that processes an input sequence using a directed graph that can be thought of as state "memory". RNNs are different than their n-gram model predecessors because they attempt to induce distributions over sequences using gradient calculation while n-gram models count exact matches between new and old examples. According to Kolen and Kremer (2001), they do not use exact templates from the training data to make predictions, but rather, like other neural networks, use their internal representation to perform a high-dimensional interpolation between training examples. For our experimentation, we use a specific type of RNN, called a bi-directional RNN (BIRNN), that processes text in both left-to-right and right-to-left manners, identical to Schuster and Paliwal (1997). Additionally, we use gated recurrent units (GRU) (Cho et al., 2014a).

   We base our decision to use an RNN on the fact that, as Yin et al. (2017) show, they are good at capturing word sequences, especially at the sentence level. RNNs are also known to learn grammatical structures automatically despite missing word or character information which sometimes leads to degradation when dealing with large sentences (Knowles et al., 2018; Cho et al., 2014a; Steijvers and Grünwald, 1996). Vukotic et al. (2016) say: "sometimes solely the current sentence suffice while in most cases, the knowledge of what has been

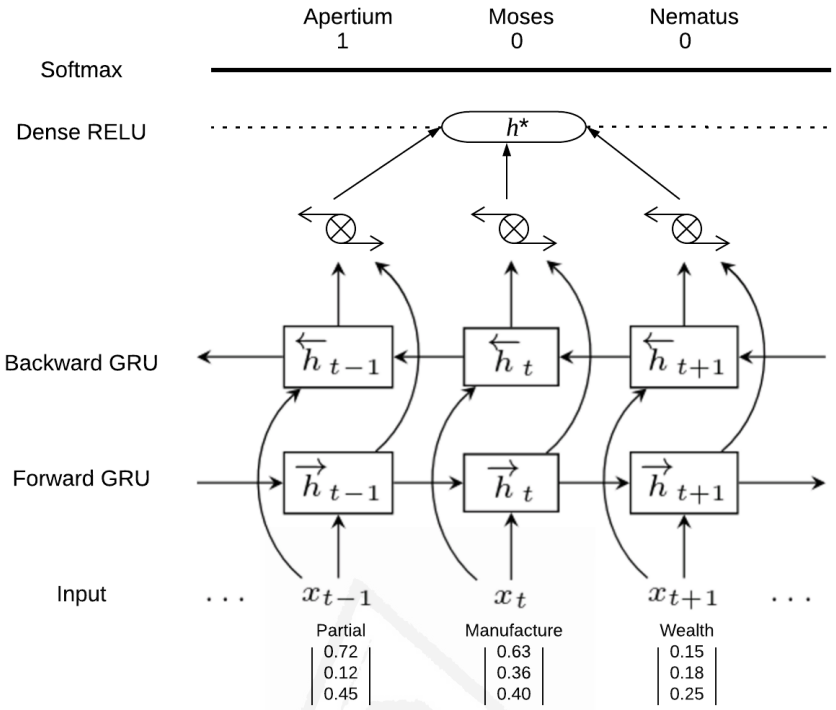


Figure 4.1: A bi-directional recurrent neural network with gated recurrent units to predict MT systems.

previously mentioned improves the understanding of the current sentence". We use a BIRNN classifier with hopes that it has the ability to learn from the context around words and classify MT systems better than the other (non-neural) classifiers.

Our BIRNN architecture consists of an embedding layer, two hidden GRU layers (one in each direction), a dense feed-forward layer using a rectified linear unit (ReLU) (Hinton et al., 2006) as its activation function, and a *softmax* layer, as show in Figure 4.1.

Word-embedding creation. The input sequence (x_1, x_2, ..., x_n) is a sequence of words mapped to vectors of real values known as word embeddings. Our word embeddings are created using Word2Vec (Mikolov et al., 2013) which produces a vector for each word in the input by using a continuous-bag-of-words (CBOW) approach (Mikolov et al., 2013).

BIRNN training and classification. In order to train the BIRNN, we make the pre-trained word embeddings available to the model and for each sentence in the training corpus we extract the embedding that corresponds to each word. The activation function (h) produces a final vector of hidden states where each word's "state" is passed on to the next hidden state. The final output, or state, of each hidden layer consists of a concatenation of words from the backward

states onto the forward state's vector and each final state represents a real number from RELU activation that is the same size in dimension as the input. We use a softmax layer to reduce the sentence representation into one of the three classifications used to map each sentence's input to a target MT system (Apertium, Moses, or Nematus). In order to speed up the training process, sentence batches are used along with fixed-length sentences (see 4.5 for configuration details). Sentences that exceed the established (maximum) length are cropped and shorter sentences are padded with zero-filling masks. This also accounts for variable-length sentences during classification.

2. **FastText Supervised Learner**: The second classifier that we test with in our SelecT combination system is the FastText[4] classifier (Joulin et al., 2017). FastText is know for its speed; however, we mostly use it here because it can achieve prediction performance comparable to RNNs (Joulin et al., 2017), especially at the sentence level. Since it is a linear classifier based on a bag-of-words like approach that does not depend on deep learning, the FastText classifier can be considered less complex than the BIRNN. The architecture, according to Yu et al. (2016), maps each vocabulary to a real-valued vector, with unknown words having a special vocabulary id. In FastText, sentences are represented using a linear representation that takes the average from a lookup table produced by word vectors that are created using an architecture similar to a continuous-bag-of-words (CBOW) model (Mikolov et al., 2013).

   The FastText framework by default offers pre-trained word vectors (Mikolov et al., 2018) that encompass more than 5 million words from crawled websites and Wikipedia news data. FastText has its own word representation (Bojanowski et al., 2017) that can be considered state-of-the-art and comparable to the most common word embeddings like Word2vec (Mikolov et al., 2013) and Glove (Pennington et al., 2014). We use the default pre-trained word embeddings which are based on a CBOW-like model.

   While the supervised classifier from FastText is inspired by the CBOW model, in order to avoid speed complications, it uses a CBOW variant created to use a bag of n-grams (see Section 4.5 for the exact number of n-grams and setup) as additional features to capture some partial information about the local word order, similar to work by Joulin et al. (2017). For comparison purposes, FastText can be considered a neural network with a single hidden layer that uses a bag-of-n-grams representation.

   Bojanowski et al. (2017) and Kovstial and Davrena (2017) have shown that FastText is a powerful source for enriching verbal representation and that it works well with morphology due to its use of sub-word information that works at the n-gram level. One of the MT systems that we use for experiments, Apertium, also works well with morphology due to its use of a morphological dictionary and shallow, structured, transfer rules. It is our hope that FastText will be able to

[4] `https://fasttext.cc/`, `https://github.com/facebookresearch/fastText/`

capture differences at the sub-word level better than the other classifiers in order to highlight Apertium's strengths by classifying those segments where Apertium would typically score better.

FastText is also a good choice for experimentation because it is efficient. Joulin et al. (2017)'s main effort was to improve the typical processing times that would sometimes take days on standard multi-core processors. They were able to train the classifier on more than one billion words in less than ten minutes using a standard multi-core CPU, and classify half a million sentences among 312,000 classes in less than a minute.

3. **Logistic Regression**: For our purposes, the Logistic Regression (LR) classifier can be considered the least complex because it is a linear classifier with a very straightforward approach. Its origin dates back to the early 19th century (Cramer, 2002) which make it a popular algorithm that is implemented in many packages such as the one by Fan et al. (2008).

   The LR classifier models the relationship between a set of variables and their outcome. For our purposes, we use multinomial logistic regression, also known as a maximum-entropy classifier, to select a specific MT system for each sentence. The probability that the LR classifier will choose a particular MT system (Apertium, Moses, or Nematus) can be expressed as:

$$\Pr(\text{y} = \text{c} \mid \text{x, W}) = \frac{\exp(w_c^T x)}{\sum_{c'=1}^{C} \exp(w_c'^T x)} \tag{4.1}$$

   Equation 4.1 can be considered the standard multi-class logistic regression model where the probability of a sentence (y) being classified with the class (c) depends on the weight of each feature vector (w_c and w'_c). The sentence-level features for our experiments are produced using a bag-of-words model and scored using term frequency inverse document frequency (TF-IDF) (Salton and Buckley, 1988) (see Section 4.5 for more details).

The SelecT classifier can be thought of as a model similar to the QE model from Chapter 3 because it attempts to use segment-level features to choose an optimum translation. During training, each source segment is labeled with the MT-system class (Apertium, Moses, or Nematus) by selecting the system with the highest BLEU score for that segment. Afterwards, during testing, the best performing classifier is used for labeling new segments, or sub-segments.

4.4 MT paradigm differences

In this dissertation, we use MT systems from different paradigms. In this chapter, we show how the SelecT classifier allows us to take advantage of each paradigm's

strengths by comparing three specific paradigms: rule-based, statistical, and neural, as used in Section 4.5. This allows us to show how well an *a-prior* selection system can increase the odds for an optimum FMR implementation by taking advantage of each MT system's approach.

Apertium, the representative rule-based MT systems, is used as the MT system for FMR in Chapter 2. We use the same version (SVN 64348) and language-pair package: apertium-en-es in this chapter. Apertium is a shallow-transfer RBMT system and, for our experiments, it may make a difference because it is able to capture grammatical differences, such as part-of-speech, well. Similarly, dictionaries and rules in Apertium are encoded by human experts and can be helpful for infrequent words and phrases not typically captured in by an SMT or NMT system due to their reduced number of occurences in the corpora, if any. Therefore, while Apertium has been shown to perform worse on English to Spanish in Chapter 2 and by Knowles et al. (2018), RBMT systems can still be considered useful for FMR.

Moses is our representative SMT system. Moses combines statistical models with phrase tables that are used for decoding. In Chapter 2, we found that Moses performs well when compared to other MT engines.[5] It is the most widely adopted open-source *statistical* MT system and it generally outperforms other RBMT systems like Systran (Dugast et al., 2007). On the other hand, Moses also peforms better than NMT in some cases (Schwenk et al., 2012). In a recent comparison to NMT, Junczys-Dowmunt and Grundkiewicz (2016) show that performance for en–es is nearly the same according to BLEU (about 1.4 difference). Moses is a complex system that, in our experiments, performs well on word ordering and on the translation of symbols like punctuation and quotation marks. In several cases, Moses is the only MT system that correctly translates rare punctuation mark differences (Knowles et al., 2018).

Nematus is used here as the NMT representative. One major advantage of Nematus over Moses and Apertium is that it uses byte-pair encoding (BPE) which starts from a character-level segmentation and eventually encodes full words as a single symbol (Sennrich et al., 2016). The potential for Nematus to score well on translations that differ at the character-level instead of at the word level is high. In WMT 2016, Nematus outperformed other SMT systems with more complex language models.

Based on the previous work using the three MT systems (Apertium, Moses, and Nematus), we believe that SelecT should outperform any single system. Each MT system is capable of outperforming the others on specific segments due to its internal workings and those differences are what a classifier is able to use as discriminant features. For example, Apertium may produce quality translations in some cases where morphology or part-of-speech linguistic features are absolutely necessary; Moses may perform better than Apertium on segments that have frequent phrases; and, Nematus will probably outperform the other systems for most segments.

[5]We train Moses on Europarl V7 (Koehn, 2005) and tune it on WMT12.

Some types of problems that an MT system may find with the test corpus, DGT-TM 2016,[6] relate to the corpus's parliamentary text. It contains punctuation irregularities and a lot of legal register segments, like article and section numbers, where target (Spanish) words do not change much despite the language difference (English to Spanish). In addition, the text contains several hundreds out-of-vocabulary (OOV) words which can be hard to cover with any MT system. Luong et al. (2014) and Alva-Manchego et al. (2017) show that Moses is conservative with deletions, yet good with punctuation. Apertium is good at making lexical and morphological distinctions. However, both Apertium and Moses are unlikely to do well with lexical complexity (Luong et al., 2014).

4.5 Experimental settings

The experiments in this chapter use the same corpora and MT systems as in Chapter 2. SelecT is first used to show that a single MT system can be selected in an *a-prior* manner for each source segment. Then, the best-performing classifier, used initially to learn from *entire* source segments, is used as a selection device when translating *sub-segments* for fuzzy-match repair.

4.5.1 Classification experiments

The source code for the three classifiers that we use in SelecT is placed on GitHub.[7] All classifiers and their respective MT system's output use the DGT-2016 TM[8] for training. We divide DGT-2016 into an 80%/10%/10% split for train/dev/test, respectively. At this point, to address the weakness found when translating from English to Spanish in Chapter 2, we focus only on the en–es language pair which contains a total of 203,214 parallel segments. We lowercase all segments and tokenize them using the tokenizer in Moses.

In order to address caveats in Chapter 2, such as difficulties with long sub-segments and proper nouns in English, we maintain the Apertium version but slightly change the SMT and NMT systems. The SMT system, Moses, mirrors the typical SMT baseline[9] and is trained on Europarl v7 (Koehn, 2005) and tuned on the development set (newstest2008) distributed for the WMT12[10] translation task. The NMT system, Nematus, is trained on Europarl v7 and News Commentary v10 corpora[11] (WMT13 training data for en–es). While there is some difference between WMT12 and WMT13, for modeling purposes, they can be considered similar corpora because the overall language is the same and they belong to the same domain – news commentary.

[6] https://ec.europa.eu/jrc/en/language-technologies/dgt-translation-memory
[7] https://github.com/AdamMeyers/Web-of-Law/EAMT2018
[8] https://ec.europa.eu/jrc/en/language-technologies/dgt-translation-memory
[9] http://www.statmt.org/moses/?n=Moses.Baseline
[10] http://www.statmt.org/wmt12/dev.tgz
[11] http://www.casmacat.eu/corpus/news-commentary.html

As mentioned in Section 4.3, the training corpus of SelecT consists of a labeled data set where each segment is accompanied by the best scoring translation using BLEU. For our experiments, there are 162,571 segments in the training set. Apertium scores best on 26,426 of the segments; Moses scores best on 54,372 of the segments; and Nematus scores best on 81,773 of the segments. If we were to apply the trained model on our test set of 20,321 segments, the perfect SelecT classification would be: 3,441 Apertium segments, 6,602 Moses segments, and 10,278 Nematus segments.

The three classifiers used for experimentation and their implementation have been covered in section 4.3. Here, we extend the explanation to discuss their configuration:

1. **Bi-Directional Recurrent Neural Network**: Word2Vec (Mikolov et al., 2013) word embeddings are created using Gensim.[12] They are trained using the continuous bag-of-words algorithm on the DGT-TM 2016 corpus. The embeddings are of size 300 and serve as input to the BIRNN.

   As a manner of reducing memory footprint, a limit of 100 words is set on each input segment to the BIRNN. Segments with more than 100 words are reduced by removing extra words from the end of the segment to fit in the input space. The DGT-2016 corpus contains few segments longer than 100 (less than .0004% of the total); therefore, changes in performance using a higher word limit would be negligible.

   The model is implemented using Theano.[13] The RNN layers themselves contain 300 hidden units, a dropout rate of 0.5, and RELU activation. Since the two layers are combined, the final dense layer sent to RELU for activation is of size 600 (300 and 300 from each RNN). The Softmax layer then consists of 3 outputs, one for each MT system.

   After several developmental experiments with various learning rates (0.025, 0.050, and 0.075), we settled on the learning rate parameter of 0.025. Additionally, we train for 20 epochs. Since training the BIRNN takes longer than the other classifiers, we had to reduce our memory footprint. So, in addition to reducing the word-length for input segments, we use 50-segment batches for training.

2. **FastText Supervised Learner**: Our FastText[14] training phase consists of 25 epochs. For word embeddings, we use a vector of 300 dimensions and a n-gram length of 5.

3. **Logistic Regression**: For our Logistic Regression (LR) model we used the popular Python machine learning framework SciKit-Learn v0.19.1[15]. Segment representations are vectors that contain the term frequency inverse document

[12] https://radimrehurek.com/gensim/

[13] http://deeplearning.net/software/theano/

[14] https://github.com/facebookresearch/fastText/

[15] https://sklearn.org/modules/generated/sklearn.linear_model.LogisticRegression.html

frequency (TF-IDF) (Salton and Buckley, 1988) scores for each word and are created using the distinct words from each document.

Model training time differs for the 3 models. FastText and logistic regression (generating a bag-of-words representation and features based on TF-IDF features) can both be trained within several minutes (on 12 cores of an Intel Xeon E-2690v2 3.0GHz CPU), while it takes roughly 16 minutes per epoch to train the bi-directional recurrent neural network (on one NVIDIA P40 GPU). For our purposes during the development stage, the best accuracy on the development set for the RNN was observed at 40 epochs. Clearly, in our experiments, the FastText and logistic regression models train faster than the RNN.

4.5.2 FMR experiments

In order to replicate experiments from Chapter 2, we use exactly the same settings as before. Thus, there are 1,993 test sentences along with a translation memory extracted from DGT-TM 2015. All three systems (Apertium, Moses, and Nematus) make up part of the SelecT system that FMR uses when calling its black-box translate method such that, when a new source-side sub-segment (σ or σ') is proposed for translation, it is passed first to the SelecT system to determine which of the three MT systems would perform best for the sub-segment. After the MT systems is selected, it is used to get the translated sub-segment τ or τ', respectively.

We use the best performing model (FastText) from our MT experiments to test the use of SelecT for FMR. As we did in Chapter 2, results for the FMR system with FastText are reported using WER and the selected MT systems; selection is done on a segment basis.

We report WER when each MT system is used to translate the whole source segment, when FMR is performed with each individual MT system, and when SelecT is used to select the MT system for FMR. It is worthwhile to note that there are cases when a fuzzy-match score is not met and the entire segment (s') is translated using MT. In those cases, we *also* use SelecT to choose the MT system to be used to translate the whole segment.

4.6 Results

We provide results of two experiments: the first experiment measures the accuracy of the classifiers of SelecT using BLEU and WER as evaluation metrics; the second experiment uses SelecT as a predictor to choose an MT system for FMR. For the first experiment, we use 20,321 development sentences with three classifiers: BIRNN, FastText, and Logistic Regression.

Label	Prec.	Rec.	F1	Accuracy
RNN SelecT MT System				
Apertium	61.05	50.65	55.37	
Moses	59.25	58.60	58.92	65.79%
Nematus	70.94	75.48	73.14	
FastText SelecT MT System				
Apertium	70.52	46.79	56.25	
Moses	60.72	60.27	60.49	**68.12%**
Nematus	71.86	80.30	75.84	
Logistic Regression SelecT MT System				
Apertium	71.30	37.26	48.94	
Moses	57.60	52.20	54.76	65.05%
Nematus	67.71	82.61	74.42	

Table 4.1: Evaluation of three classifiers on three MT system labels.

4.6.1 MT experiments

Results for the first experiment are reported in Table 4.1. Each segment is assigned an MT system label which corresponds to the system with the highest BLEU score. It is the classifier's job during testing to predict the best-performing MT system for each segment. The best classifier on the 20,321 segments is the FastText classifier. It is 68.12% accurate and slightly outperforms (less than 1%) the BIRNN classifier. It is important to note that precision (Prec. in the table), recall (Rec. in the table), and F1 scores are displayed for each respective MT-system label. True positives are measured as the count of segments where the classifier correctly predicts the respective MT system; false positives are the count of segments where the classifier should have predicted the respective MT system but it did not; false negatives (FN) are the count of segments where the classifier incorrectly labels segments that should have been labeled as the respective MT system. Precision, Recall, and F1 scores are the standard evaluation measures for classification tasks.

In Table 4.2, we report more information from the first experiment that shows: 1) the maximum (upper) and minimum (lower) bounds, that is the best-performing and worst-performing systems; 2) performance using each type of SelecT classifier; and, 3) how well each system performs in isolation – if we were to use the respective system as the sole translation engine for all 20,321 segments.

Table 4.2 provides a general idea of how well a classification system like SelecT works when compared to the three MT systems in isolation. It is clear that the BLEU performance from the best-performing SelecT systems (Logistic Regression and FastText) is nearly the same as using Nematus in isolation. However, word-error rate for both systems is somewhat better than Nematus. The FastText classifier provides a 19.04 improvement over the BLEU lower-bound (90.2% of the potential difference) and a 14.36 improvement over the WER lower-bound (83.4% of the potential difference).

Upper and Lower Bounds		
	BLEU	**WER**
Best	40.08	46.70
Worst	18.97	63.91
SelecT Classifiers		
	BLEU	**WER**
RNN	37.36	49.69
FastText	38.01	49.55
LR	38.03	49.97
MT Systems in Isolation		
	BLEU	**WER**
Apertium	20.96	59.19
Moses	30.05	54.02
Nematus	37.36	51.77

Table 4.2: A comparison of SelecT classification with MT systems and upper/lower bounds.

System	**RNN**	**FT**	**LR**	**Ref**
Apertium	2855	2283	1798	3441
Moses	6530	6553	5983	6602
Nematus	10936	11485	12540	10278

Table 4.3: Count of segment for 3 predictive SelecT models.

The FastText classifier also outperforms the best individual system (Nematus) by 0.65 BLEU score and 2.22 WER. The BLEU and WER score differences are statistically significant according to paired bootstrap re-sampling (Koehn, 2004) with a p-level of 0.05.

The average between the upper and lower bounds is a good baseline to beat and demonstrates that our system is successful at predicting the correct high-scoring system most of the time. Random selection could also be used as a lower bound for the baseline; but, in our experiments, random tests did not do much better than the worst baseline (about 2 points BLEU score).

Table 4.3 shows the predictive capability of each classifier at the segment level. It provides a segment count for each classifier and a corresponding system reference count. This helps to show how well the predictors perform at choosing an MT system.

When using the FastText system (FT in Table 4.3) as a predictor, Apertium outperforms Moses and Nematus on 2,283 of the 20,321 total sentences; Moses gets 6,553 of them correct when compared to the reference; and, Nematus gets 11,485 correct

		Apertium		Moses		Nematus		SelecT	
	TM	MT	FMR	MT	FMR	MT	FMR	MT	FMR
FMT: 60%									
Error (%)	55.0	65.3	36.5	45.8	29.2	48.6	30.1	44.8	27.9
Er. (%) on matches	20.1	65.3	17.9	45.8	16.2	48.6	17.1	44.8	16.0
# matches	1184	1993	1184	1993	1184	1993	1184	1993	1184
FMT: 70%									
Error (%)	61.0	65.3	38.5	45.8	30.5	48.6	31.15	44.8	29.2
Er. (%) on matches	16.3	65.3	14.6	45.8	13.7	48.6	13.9	44.8	13.5
# matches	828	1993	828	1993	828	1993	828	1993	828
FMT: 80%									
Error (%)	69.7	65.3	42.6	45.8	32.6	48.6	33.7	44.8	31.7
Er. (%) on matches	13.1	65.3	11.9	45.8	11.3	48.6	11.4	44.8	11.2
# matches	660	1993	660	1993	660	1993	660	1993	660

Table 4.4: Word-Error Rate (WER) for FMR using SelecT to select, on a sub-segment basis, the MT system to be used for fuzzy-match repair.

(over-predicting by more than 1000 examples). This shows that the FastText model tends to prefer Nematus in more cases than the BIRNN

4.6.2 FMR experiments

For FMR, we evaluate our best performing classifier (FastText) from Table 4.1 by using it to select, on a sub-segment basis, the MT system to be used for translating sub-segments σ and σ' for building the repair operators to be used. Much like results from Table 2.5 in Chapter 2, we report on two error rates: 1) WER computed on the whole test set and 2) WER computed only on the segments for which a translation unit (TU) with a fuzzy-match score above the fuzzy-match threshold is found (error on matches). We use those two different forms of measurement to better understand how a translator or CAT tool user would use FMR in a production setting since they would typically only see matches.

Table 4.4 shows the performance of our approach for 3 different fuzzy-match score thresholds (FMT) —60%, 70% and 80%—. We report the WER for the same three MT systems (Apertium, Moses, and Nematus) as before and when using the FastText classifier from SelecT. The last column (SelecT) shows how well SelecT performs when used for selecting the MT system in isolation (the MT sub-column) and then when used for FMR (the FMR sub-column).

The SelecT system outperforms the results from Chapter 2. In addition to outperforming the stand-alone FMR work from before, it seems to score well when compared to other work that uses various MT paradigms (also from Chapter 2). SelecT outperforms all systems in both fuzzy-match situations (matched or not). This is more evident when the FMT is lower. For example, at 60% FMT, SelecT is about 1 point

better than the best-scoring MT system in isolation (Moses) and 1.3 points better than the best MT system for FMR (Moses also). Scores are similar for the same MT system using an 80% FMT and consistently score better than Nematus, the best scoring MT system in isolation during development from Table 4.2. It performs best for FMR when there's no fuzzy-match and the MT system has to translate the entire source segment (s'). This is first seen in the performance from Table 4.2 but even more detailed in the direct comparisons in Table 4.4. We attribute this behavior to training on segments instead of sub-segments.

4.7 Concluding remarks

Fuzzy-match repair has already shown its potential for improving translator's productivity. The SelecT system presented here shows performance gains of as much as 2 points in WER over the initial work in this thesis. We believe that the gains presented here are due to Moses and Apertium's phrase-based and rule-based technology, respectively, that allow it to come somewhat closer to translator's needs at the sub-segment level. In order to explain this, we observe that Nematus is more likely to correctly handle polysemous words (should English *march* be translated to Spanish as *marzo* (the month) or *marcha* (the action)). However, some of Nematus' errors involve seemingly arbitrary translations of words or the addition of arbitrary words. For example, the English "*identification numbers*" is correctly translated as "*números de identificación*" by Apertium, but Nematus translates it as *identificación de identificación* (Moses translates it nearly correctly, but leaves off the "s" in "*números*"). Similarly, Apertium correctly translates the English "*saffron*" as *azafrán*, whereas Moses leaves it untranslated ("*saffron*") and Nematus translates it mysteriously as "*lágrimas de los perros*". Sub-segments in FMR are usually shorter and have more punctuation involved (especially in the DGT-TM 2015 corpus).

Our experiments show that SelecT can be used as an ensemble system that covers more cases than any MT system tested and could, thus, be more valuable for a translator or CAT-tool user. SelecT is agnostic with respect to the MT systems being used and does not require underlying process changes. Our results also help to explain how well various classifiers perform on FMR. The classifiers could also be trained with more detailed, feature-level information such as sentence and word-length ratio as done for quality estimation in Chapter 3.

Chapter 5

Combining FMR with automatic post-editing

Previous chapters have focused on fuzzy-match repair performance in isolation as a modular element of the CAT environment. While CAT tool users can use FMR with TMs as a primary device for improving their productivity, they can also use automatic post-editing (APE) systems, among others, to get further productivity gains. In this chapter, we combine FMR and APE: first, a fuzzy-match repaired segment is produced from the translation unit proposed by the TM and the source segment to be translated; then, the repaired segment is further improved by an APE system specifically tuned for this purpose. Experiments conducted on the translation of English texts into German show that, by combining the two technologies, the quality of the translations improves up to 23% compared to a pure MT system. The improvement over a pure FMR system is of 16%, showing the effectiveness of our joint solution.

5.1 Introduction

The focus of this chapter is on extending the baseline work presented in Chapter 2 to improve fuzzy-match repaired segments by adding an additional post-editing technique called *automatic post-editing* (APE) (Chatterjee et al., 2017). APE methods have been introduced in CAT tools as a post-editing technique to correct machine-translated segments. Research has shown that translators can be more productive when using state-of-the-art post-editing techniques (Isabel, 2017). As shown in recent WMT tasks (Bojar et al., 2017; Chatterjee et al., 2018a), APE, as a secondary device for correcting MT segments, requires less post-editing effort to convert it into an adequate translation for the intended purpose. In this chapter, we demonstrate that the application of APE to the best possible repaired segment produced by the FMR algorithm presented in Chapter 2 leads to improved translation proposals.

As motivated by Parton et al. (2012), an APE system can help to improve MT output in two major ways: (1) by exploiting information not available during translation or (2) performing a deeper text analysis than a typical MT system decoder. Additionally, the APE system can adapt the output of a general-purpose MT system to the lexicon/style requested in a specific application domain. Altogether, APE provides professional translators with improved MT output to reduce (human) post-editing effort.

APE systems do not rely on translation memories and are effective without the initial intervention of the translator. Nonetheless, our experiments show that APE can be seamlessly integrated into the typical translation pipeline as a post-editing technique to improve *segment-level* proposals from FMR, not only MT output. FMR is first used to produced a repaired translation proposal and then APE is used as a tool to better the quality of that proposal. Our proposed system outperforms both a competitive neural MT system and FMR alone.

The remainder of this chapter is organized as follows. First, we review the state-of-the-art APE system used in our experiments. Second, in Section 5.3, we show how the two technologies are "glued" together to form a new system that is added in a modular way to a traditional CAT pipeline. Third, we describe our experimental settings. Fourth, we present our results in Section 5.5. Then, we introduce human reviews as a sanity check to confirm our findings from the qualitative analysis. Finally, we wrap up our findings with a conclusion.

5.2 Automatic post-editing

Automatic post-editing is the task of correcting recurring errors made by an MT system by learning from human corrections. Starting from the seminal work by Simard et al. (2007), the problem has been tackled as a "monolingual translation" task in which the MT output is translated into an improved text in the target language. Under this definition, the "parallel data" used for training an APE system consist of pairs of the form (target, post-edited target) rather than the (source, target) pairs normally used in MT. Following the translation-based approach, initial solutions relied on the phrase-based paradigm (Simard et al., 2007; Dugast et al., 2007; Terumasa, 2007; Pilevar, 2011; Béchara et al., 2011; Chatterjee et al., 2016). Yet, in the past couple of years, top results have been achieved by neural architectures (Pal et al., 2016; Junczys-Dowmunt and Grundkiewicz, 2016; Chatterjee et al., 2017; Junczys-Dowmunt and Grundkiewicz, 2017). In particular, most of the neural architectures cited here are based on recurrent attention encoder-decoder networks and use triplets of the form (source, target, post-edited target) to train the APE system.

Recent advancements made by participants in the APE shared task at WMT 2017 have shown the capability of the APE system to significantly improve the performance of a black-box MT system gaining up to seven BLEU points (Bojar et al., 2017).

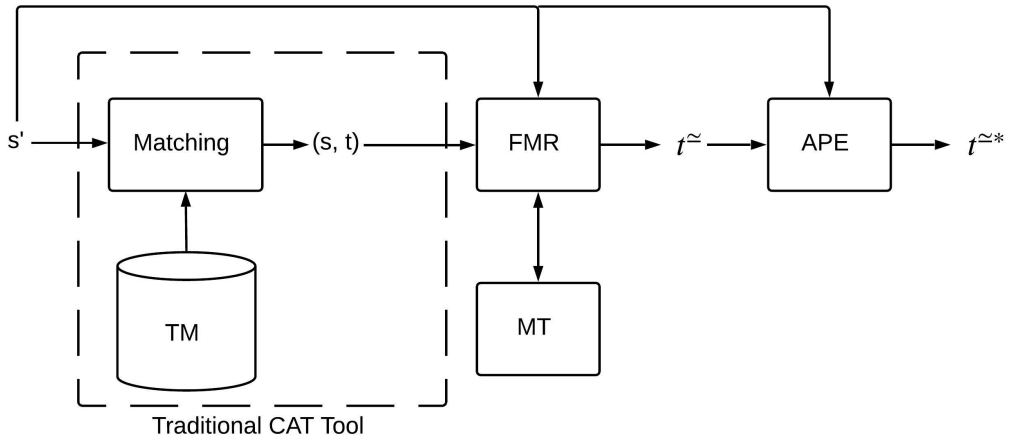


Figure 5.1: Seamless addition of fuzzy-match repair (FMR) and automatic post-editing (APE) in a traditional computer-aided translation (CAT) pipeline. A source segment s' is first *fuzzy-matched* with a translation unit (s, t) from the translation memory (TM). The target segment t is then fuzzy-match repaired to get $t^{\simeq}$. Finally, the automatic post-editing system modifies $t^{\simeq}$ to create $t^{\simeq *}$.

Our APE system is a re-implementation of the multi-source attention-based recurrent encoder-decoder system (Chatterjee et al., 2017) that achieved the best performance in the automatic evaluation by Fondazione Brunno Kezzler (FBK) at the APE shared task at the WMT 2016. It uses two different encoders to independently process the source and the MT segments. Each encoder consists of a bi-directional GRU and has its own attention layer that is used to compute weighted context. To obtain a single context, the two context vectors are combined via a feed-forward network. To regularize the multi-source network and to avoid over-fitting, a shared dropout is applied to the hidden state of both encoders and to the merged context. The APE multi-source architecture makes it particularly suitable for the FMR task because it corrects common MT caveats, such as erroneous translations due to a lack of domain-specific context, that may occur when using MT as a black box, as is done in stand-alone FMR.

5.3 Combination of FMR with APE

Our combination system, depicted in Figure 5.1, works as follows. After fuzzy-matching, the CAT tool system uses the FMR technique from Chapter 2 to propose a fuzzy-match repaired segment $t^{\simeq}$, not necessarily present in the TM, by using the segment to be translated s' and the TU (s,t) retrieved from the TM. Then, the APE system uses $t^{\simeq}$ and s' to produce the final FMR and APE-repaired segment $t^{\simeq *}$.

Source:	article 18 , paragraph 1 , of the co2 act
TM:	article **45** , paragraph 1 , of the co2 **ordinance**
FMR:	artikel 18 absatz 1 der co2-verordnung
APE:	artikel 18 absatz 1 des co2-gesetzes
Reference:	artikel 18 , absatz 1 des co2-gesetzes

Table 5.1: An example of integration of fuzzy-match repair (FMR) and automatic post-editing (APE).

As a quick way of showing that our combination system outperforms the FMR baseline in Chapter 2, we use the best (oracle) fuzzy-match repaired segment instead of an approximation via quality estimation as was done in Chapter 3. That is, the fuzzy-match repaired segment that is most similar to the reference translation is provided as input to the APE system.

Table 5.1 shows the combination of FMR and APE on a real example. First, FMR repairs the TM proposal by replacing two words (*45* and *ordinance*); notice that FMR incorrectly translates *co2 act* as *co2-verordnung*. APE then takes the FMR proposal and produces an improved translation, *co2-gesetzes*, that is closer to the reference translation.

APE has been combined with other techniques in a CAT tool setting. We briefly describe those combinations that could be considered related to our work. The first, and probably the most relevant work, is based on the combination of MT quality estimation with APE. Chatterjee et al. (2018b) present three different combination types: one in which sentence-level MT QE is used to activate an APE system, a second one in which word-level MT QE is used to guide the APE system, and a third one that uses MT QE to chose between the original MT output and its automatically post-edited version.

Hokamp (2017) includes word-level MT QE features as additional inputs to an APE system and trains several neural models using different input representations, but sharing the same output space. These models are finally ensembled together and tuned for APE and MT QE.

Lastly, Tan et al. (2017) attempts to correct a common problem in APE known as "overcorrection". They do this by specifying two models (called neural post-editing models) and then combine MT QE to help select a model for the translation. This by no means is related to fuzzy-match repair; but, the idea of combining several systems around APE is similar to what we are doing.

5.4 Experimental settings

Unlike previous chapters that attempt to show how well FMR works with specific language pairs, namely English and Spanish, here we use English and German because they performed well in a recent (2018) WMT task (Chatterjee et al., 2018a) and have

become a de-facto standard for APE experiments. We experiment with a combination of FMR and APE using a phrase-based MT system as the SBI for FMR. In addition, we use APE on the output of two MT systems, a phrase-based MT system and a neural MT system, as a point of comparison. This section goes over the details of the data and systems we use. One of our objectives is to maintain practical (non domain-specific) settings for both our data and MT systems despite other works (Knowles et al., 2018; Chatterjee et al., 2018b) that have already shown that training the MT systems on in-domain data, especially in the case of a neural MT system, can be advantageous.

5.4.1 Data

Our entire dataset is based on 4,000 randomly selected sentences from the DGT translation memory (DGT-TM-release 2018).[1] This TM is available in several languages containing many translation units.[2] In our evaluation we use the English–German (en–de) TM extracted using the formal DGT extraction methodology mentioned on their website.

As done in Chapter 2, FMR is used to generate fuzzy-match repaired translation proposals for the 4,000 sentences by using the entire en–de DGT TM to look for translation units to repair; it is worth noting that the whole DGT TM is not used in any way by the APE system. If a translation unit with a fuzzy match score above 60% is found, it is used for FMR; otherwise, Moses (Koehn et al., 2007), the SMT system used as the SBI for FMR, is used to translate the source sentence.

Of the 4,000 sentences selected at random from the DGT TM, 2,500 were randomly selected and used to fine-tune the APE system (see Section 5.4.3), 500 were used for development, and 1,000 for testing. Altogether, about 350 sentences were not successfully repaired by FMR; in those cases we used the output of Moses.

5.4.2 Machine translation systems

We use the phrase-based statistical MT system Moses (Koehn et al., 2007) as a SBI for FMR; it has shown to perform well in previous experiments and in the black-box setting (Knowles et al., 2018). In addition to Moses, we use the neural MT system Nematus (Sennrich et al., 2016) as a point of comparison. We do not use Nematus as a SBI for FMR because of time constraints; in any case, the phrase-based MT system performed better with APE alone than the neural MT system (see Table 5.2).

For Moses, we use pre-trained models downloaded from `http://www.statmt.org/moses/RELEASE-3.0/models/` with default configuration settings such as the inclusion of the UNK symbol for words that were not translatable.[3] Nematus, on the other

[1] `https://ec.europa.eu/jrc/en/language-technologies/dgt-translation-memory`

[2] For some statistics about this TM, please visit `https://wt-public.emm4u.eu/Resources/DGT-TM_Statistics.pdf`

[3] This is left on purpose for clear distinction during human evaluation.

hand, is trained on a collection of corpora belonging to different domains and containing approximately 4 million sentences. In particular, we use domain-specific parallel corpora from the European Central Bank, Gnome, JRC-Acquis, KDE4, OpenOffice, PHP and Ubuntu,[4] and generic training sets obtained from the CommonCrawl dataset[5] and Europarl.[6] The Europarl corpus can be considered an in-domain dataset because it belongs to the same domain of the DGT TM collection.

We train Nematus with default hyper-parameters[7] The training corpus is first processed using byte pair encoding (Sennrich et al., 2016), so that the less frequent words are segmented into their sub-word units, resulting in vocabularies of maximum size of 90k entries. The size of word embeddings and hidden layers is set, respectively, to 500 and 1024. Source and target dropout is set to 10%, whereas, encoder and decoder hidden states and embedding dropout is set to 20%. The learning rate is set to 0.001. The cost function is computed on mini-batches of 100 sentence pairs with maximum length of 50 tokens, extracted from the randomly shuffled data after each epoch. The models are optimized using Adagrad (Duchi et al., 2011) and every 10,000 mini-batches they are evaluated with BLEU on the development set. Training stops after ten evaluations with no BLEU improvement.

5.4.3 APE settings

The APE system is trained on the eSCAPE corpus (Negri et al., 2018), a collection of 7.2 million triplets (source, MT output, and reference), where the MT outputs have been created by a phrase-based MT system. It consists of datasets belonging to different domains and it has been filtered by removing duplicates and too short (3 words) or too long (60 words) segments.

To adapt the generic APE system to the FMR task, the model has been fine-tuned (Luong and Manning, 2015) on 2,500 triplets (see Section 5.4.1), where the source input is paired with the fuzzy-match repaired translation proposal produced by FMR using an oracle approximation from the original set of 4,000 sentences that matched a 60% fuzzy-match threshold or above.

Similar to the neural MT system, the APE system is trained on sub-word units by using BPE (Sennrich et al., 2016). The APE vocabulary is created by selecting 50k most frequent sub-words. Word embedding and GRU hidden state size is set to 1024 – default settings for most APE systems. Network parameters are optimized with Adagrad with a learning rate of 0.01 following the work by Farajian et al. (2016), which empirically showed that Adagrad has a faster convergence rate and better performance than Adadelta (Zeiler, 2012). Source and target dropout is set to 10%, whereas, encoder and decoder hidden states, weighted source context, and embedding dropout are set to

[4] All available at `http://opus.lingfil.uu.se`.

[5] `http://www.statmt.org/wmt13/training-parallel-commoncrawl.tgz`

[6] `http://www.statmt.org/europarl/`

[7] `https://github.com/EdinburghNLP/nematus`

20% as the default system in WMT 2017 (Chatterjee et al., 2017). After each epoch, the training data is shuffled and the batches are created after sorting 2,000 samples in order to speed-up the training. The batch size is set to 100 samples, with a maximum sentence length of 60 sub-words. The fine-tuning step is performed using the same parameters of the generic training.

5.4.4 Combined FMR and APE settings

Our FMR approach is identical to the FMR approach presented in previous chapters. The differences are in the MT system used as the SBI, the language pair, and the TM. The fuzzy-match repaired segment $t^{\simeq}$ produced for each source segment s' serves as input to the APE system. We experiment with one fuzzy-match repaired segment in particular: the oracle fuzzy-match repaired proposal which is chosen as the best possible fuzzy-match repaired segment for each segment s' by computing the word-based edit distance between the fuzzy-match repaired segment and the reference translation.

5.4.5 Evaluation setting

For evaluating the combination of FMR and APE, we use two major metrics: BLEU (Papineni et al., 2002) and WER. We report on BLEU because it is a centerpiece of the development of MT systems; and, we report on WER because it is the main FMR evaluation metric from previous chapters.

In addition to automatic evaluation metrics, a native German speaker conducted a human evaluation. The evaluator is not a translator; yet, does have a background in natural language processing and evaluation. The evaluator is presented with several random sentences to compare and contrast the differences. We report the evaluators overall evaluation on the best performing systems in our results.

5.5 Results

Table 5.2 reports the BLEU and WER results of three main sets of experiments: (1) MT and TM; (2) MT with APE; and, (3) FMR and FMR with APE. The first set of experiments (MT and TM) consists of (a) the output of the phrase-based MT system Moses; (b) the output of the neural MT system Nematus; and (c) the translation proposal as found in the TM. As can be seen, the TM performs the best when compared to the two MT systems (+∼25 BLEU points over the phrase-based MT and +∼13 BLEU over the neural MT). We attribute the performance of the TM approach to the fact that the DGT-TM is highly repetitive: it is likely that a match is found when a high fuzzy match threshold is used. As a matter of fact, the TM was carefully selected so that for the 1000 test sentences there is a match at a minimum of 60% fuzzy-match threshold; that is, for all of the test segments there is a translation unit for which the fuzzy match score is 60% or above. For 70% fuzzy-match score there are 763 matches, for 80% 436, and for 90% 184.

System	BLEU	WER
Phrase-based MT	39.62	49.74
Neural MT	51.54	38.00
TM	64.95	24.91
Phrase-based MT-APE	60.02	33.30
NMT-APE	56.58	35.00
FMR	68.36	23.17
FMR-APE	80.54	15.54

Table 5.2: Performance of three approaches (use of a phrase-based MT system, use of a neural MT system, and use of the TM proposal without repairing), of the use of APE to better the MT outputs, the use of FMR alone when the fuzzy-match repaired segment is selected using an oracle, and of the combination of FMR and APE.

While the TM results are better than those reported in Chapter 2, FMR and FMR-APE outperform it. Combining FMR with APE improves the translation quality of the stand-alone FMR system by a large margin (+12 BLEU points). In all experiments the addition of the APE system helps achieve better results. In a production system, ideally the MT and APE systems would be properly trained on in-domain data to get the maximum benefit from combining the two methods. Our results exclude other system combinations, such as the inclusion of in-domain data or quality estimation techniques, in order to solely show that the combination of FMR and APE under generic (out-of-the-box) settings is favorable.

The overall APE gains over FMR alone can be classified into two main categories: (1) addition of missing tokens and (2) lexical substitution. In the former, since APE is configured to accept as input the source segment and the fuzzy-match repaired segment, APE may insert tokens that are not present in the FMR translation proposal. One example from our test set is described as follows: FMR proposes the segment, *30 2015*, for the source segment *30 October 2015*, FMR discards the word *October*. When the APE system receives the FMR proposal with the word *October* missing, it decides to reinsert the word and, as a by-product of that, ends up matching the reference sentence, *30 Oktober 2015*. The latter category (lexical substitution) is mainly related to the identification of words and their correctness; it is very important when dealing with one or more TMs, where two suggestions can only differ by one word. An example from the test set of lexical substitution follows: FMR proposes, *Verordnung 2015 / 8*, for the source segment *Regulation 2015 / 7*. FMR introduces an incorrect number for the month (8 instead of 7). Since APE leverages the source segment and the FMR output, the APE is able to use lexical substitution to set the correct value which matches the reference *Verordnung 2015 / 7*.

The two evaluation metrics (BLEU and WER) show how well our best system performs and would probably be enough to show that it is worthwhile to combine FMR with APE. However, as an extra qualitative check, we verify the translations from our best performing systems with a native German evaluator. The evaluator is

Best System	Human Rating
TM	2.84
Phrase-based MT-APE	2.82
FMR	2.90
FMR-APE	3.67

Table 5.3: Average human evaluation for fluency and coherence given the source sentence for the best system combining FMR and APE where the native evaluator was asked: "Is the translation understandable and a valid translation given the source sentence?". Translations are rated using a 5-point Likert (Likert, 1932) scale where 1 means strongly disagree and 5 means strongly agree.

quite tuned to natural language processing and has a good idea of the typical problems that may occur with an MT system. The system translations were measured for fluency and coherence given the source sentences where the native evaluator was asked: "Is the translation understandable and a valid translation given the source sentence?". Table 5.3 shows a quick overview of how the best systems perform on a Likert scale (Likert, 1932) (1=Strongly Disagree, 5= Strongly Agree): the human evaluation score is in line with the automatic metrics reported above.

We also asked the human evaluator to provide general comments on each of the best-performing systems. We did this to get a better idea of the types of errors each system made. Below is an overview of what the evaluator found.

TM. The most common error from the TM was "missing" or "wrong" data which describes typical information in the parliamentary texts like an article changing from 33 to 45. This is one of the reasons that a translator would like to use a TM because the translator would typically only have to change the numbers in those situations. There are also a few comments such as "wrong" part-of-speech, e.g. an adjective or noun being wrong.

Phrase-based MT-APE. Unlike the TM, we see some common phrase-based MT mistakes such as "noun cases wrong". Also, the evaluator stressed the fact that there were several unknown words. In addition to the normal mistakes, the evaluator noticed that quite a few of the translations just "did not make sense", even more than the TM. That could be coupled with another finding, "repetition", to form what seems to be somewhat common in phrase-based MT-backed APE systems (Chatterjee et al., 2016).

FMR. The best FMR system is not immune to issues either. This could be due to the MT systems used. Many of the errors were similar to the phrase-based MT-APE system; however, other errors were reported such as "punctuation is weird" and "important" words are missing. Nonetheless, in more cases than others, it seems that the FMR system gets the underlying meaning correct.

FMR-APE. This system performed the best in all cases. While there were comments concerning unknown words (typical of the phrase-based MT translations), we saw some issues of morphology such as problems with inflection. For the most part, the evaluator made few comments because the translations were easier to understand than all other systems.

5.6 Concluding remarks

In this chapter, we propose a two-step process able to generate improved translations. This approach relies on two state-of-the-art techniques: fuzzy match repair (FMR) and automatic post-editing (APE). Given a translation unit and the segment to be translated, the FMR module creates a set of fuzzy-match repaired segments. The selected repaired segment is then fed as input to the APE system that fixes its errors. When compared against MT, TM-based approach and FMR alone, the combined solution outperforms all these methods indicating the effectiveness of the proposed technique.

We measured performance using common MT performance metrics: BLEU and WER. In addition to BLEU and WER, we provided a human rating from a native German speaker as insight into how the best-performing systems fair to the average reader (not necessarily a translator).

Chapter 6

Concluding remarks

This chapter summarizes the dissertation work by providing a high-level review of the different steps taken to both present and test the fuzzy-match repair approach presented in this dissertation. We cover the major contributions to the state of the art and summarize research lines for future work that could potentially improve the performance of our FMR approach.

6.1 Summary

The main focus of this dissertation is the introduction of a fuzzy-match repair algorithm for its use in computer-assisted translation tools, capable of using any available source of bilingual information (we use MT systems in our experiments) in a black-box manner, i.e. without access to its internal workings. The hope is that by applying the different techniques described within this dissertation, human intervention from the post-editing point of view could be kept to a minimum while at the same time preserving translation quality. Not only do we introduce a state-of-the-art algorithm for fuzzy-match repair; but, we present several improvements that can be considered important contributions to the state of the art. To be more specific, these are the principal components that this dissertation focuses on:

- a novel, fuzzy-match repair algorithm that is capable of using any source of bilingual information as a black box to propose fuzzy-match repaired segments based on translation memory entries; and, that models all edit operations (insertions, deletions, and substitutions) in the same way;

- the design of a set of language-independent features for quality estimation that can be used to select the best fuzzy-match repaired segment in a multi-lingual environment independent of the FMR system or source of bilingual information used;

- the combination of fuzzy-match repair with automatic post-editing to further reduce the post-editing needed by the proposals in a seamless manner;
- the design of black-box features, i.e. without access to the inner workings of the FMR algorithm, that allow our QE approach to be used for evaluating the quality of fuzzy-match repaired segments produced using other FMR approaches;
- a classifier that is able to select, a-prior, the MT systems to be used for FMR on a segment basis.

With regard to the fuzzy-match repair algorithm, this dissertation presents and empirically tests a new approach that is designed to require only the presence of a translation memory and a source of bilingual information while at the same time being highly effective in a CAT tool environment. Other principal advantages include: source and target language independency and easy integration into any CAT tool environment. In Chapter 2, experiments are presented with three language pairs (English–Spanish, Spanish–Portugese, and Spanish–French) that show how well the algorithm performs. Three major machine translation paradigms (rule-based, statistical, and neural) are used to test the algorithm's potential. The findings show that our approach is more effective than using a machine translation system alone and it can be used with modern software backed by translation memories like OmegaT.

Another innovative technique that this dissertation brings to light is a quality estimation technique for selecting fuzzy-match repaired segments. We show that complex machine learning algorithms based on non-linear regression can accurately estimate, given several segment-level features, the best fuzzy-match repaired proposal among many. This novel approach is an important contribution, in addition to the main algorithm for which it is titled, because it can also be used as a test-bed for other fuzzy-match repair techniques. Additionally, the quality estimation technique, which is found to work best with extremely randomized trees, is highly effective as a regressor for several language pairs.

The modularity of our fuzzy-match repair solution is an important feature. It is not solely that our system can be used with any source of bilingual information; it can also be used in combination with other systems that typically exist in a CAT tool environment. Much like a machine translation system, it takes as an input a source sentence and produces an output. Indeed, it also requires the presence of a translation memory. However, in this dissertation we show how easy it is to combine our system with a state-of-the-art automatic post-editing system. Other combinations are left for future work; but, the ease-of-integration that our approach displays as a modular device is a big advantage.

In regards to the predictive system (SelecT), it is a classifier that is developed and tested on deep learning techniques (a bi-directional recurrent neural network) along with other machine learning techniques such as logistic regression. SelecT's addition to this dissertation provides evidence that the black-box nature of our FMR approach

can benefit from other systems in an agnostic way. The classifier is able to be used in conjunction with the MT system that backs FMR to further improve the entire process.

6.2 Future research lines

What follows is a list of open research lines that may be followed to study more in depth some of the approaches proposed in this dissertation:

1. The sub-segments that our FMR method operates on are gotten using a phrase-pair extraction algorithm (Koehn, 2010, section 5.2.3). In Section 2.5.3, we show that FMR favors sub-segments that have context around them. Future research lines could consider other work (Bulté et al., 2018; Koehn and Senellart, 2010) to determine which sub-segments should be applied and on what occasions. In order to preserve the black-box nature of our approach, this would have to be done in a way that does not take into consideration the MT engine used.

2. An analysis of the morphology of the repaired operators (sub-segment length, number of gaps, anchored context, etc.) could be used to devise contextual features to decide the repair operators to be used to produce the best repaired segments. It could be used to discard low quality repair operators and reduce the amount of fuzzy-match repaired segments that are generated per source segment and translation unit to be repaired.

3. In our experiments, we show that several techniques can be used with FMR to reduce post-editing needs. However, with the exception of Chapter 3, we use an oracle evaluation to show how well FMR performs. Since our QE approach is effective at selecting fuzzy-match repaired segments, future research should include it instead of an oracle. The quality estimation technique described in Chapter 3 could also be used as a mechanism for choosing the best fuzzy-match repaired segments from a multitude of systems like the ones proposed by Bulté et al. (2018), Dandapat et al. (2011), Koehn and Senellart (2010), Biçici (2008), and Hewavitharana et al. (2005).

4. In Chapters 2 and 4, there is evidence that FMR can be used with any MT engine; and, for some input, it may be advantageous to use one MT engine over another. Further investigation could be done to test other MT engines or sources of bilingual information such as bilingual dictionaries, lexicons, translation search engines or translation memories, similar to previous work on word alignment (Espla-Gomis et al., 2012).

5. The QE technique from Chapter 3 could be extended to include features from post-editing tasks to train regressors that use human-level information similar to the QE tasks presented by Specia (2011) and Chatterjee et al. (2015). Post-edited information, such as the position of the sub-segments modified by the

post-editor in a fuzzy-match repaired proposal, could serve as strong input signal to the quality estimator of the FMR system.

6. The QE technique from Chapter 3 uses a regressor to determine the best possible fuzzy-match proposal at the *segment* level. Future investigation could use a QE technique that determines the best fuzzy-match repair at the *repair operator* level instead.

7. SelecT is a classifier used in Chapter 4 for selecting an MT engine for each source segment to be translated (s'). We experimented with several (out-of-the-box) classifiers such as a logistic regression model, a recurrent neural network (Schuster and Paliwal, 1997), and a FastText classifier (Joulin et al., 2017). Future lines of research could use classifiers with more complex architectures to better approximate which MT engine to use depending on the domain and/or language.

8. The SelecT system is trained on out-of-domain data in a black-box manner. We strongly feel that the FMR results presented in Chapter 4 could be improved in future research by training SelecT models on in-domain corpora like previous work (Knowles et al., 2018) that shows in-domain trained systems work best with FMR.

9. SelecT training was performed by using each system's sentence-level BLEU (Papineni et al., 2002) score to determine the class label (or best MT engine) in Chapter 4. Metrics such as Meteor (Lavie and Agarwal, 2007), HTER (Snover et al., 2006), CHFR (Popović, 2015), or WER Wagner and Fischer (1974) could be used in future research lines to find correlations that better align with human judgment.

10. In Chapter 5, we covered the juxtaposition of two orthogonal technologies (FMR and APE) to improve post-editor productivity. However, we did not cover the use of QE or in-domain corpora when combining FMR with APE. In-domain corpora, like Knowles et al. (2018) suggest, could be used to improve FMR performance when combined with an APE system. Additionally, QE could be used as a way to rank fuzzy-match repaired segments that are combined with APE.

11. Future lines of investigation could benefit from an in-depth study of translator's productivity when using FMR. In order to perform a productivity study on FMR, much like the previous MT study by Cadwell et al. (2016) that use corpora (DGT-TM) similar to that of Chapter 2, future work would need to take into account several human circumstantial factors such as ergonomics, translator's needs, well-being, and limitations. By including translators in the evaluation, one could simulate a *human-to-machine* (O'Brien, 2011) collaboration where the FMR system is capable of learning from the translator by updating the quality estimation model after translations were made. This could lead to a metric for correlating FMR performance to human acceptance. One example line of investigation to accomplish correlation with human judgment could measure the

post-editor productivity difference between the use of FMR, QE, and APE methods in CAT tools and the TM alone.

Index of abbreviations

Index of frequently used symbols

List of Figures

List of Tables

Bibliography

Alva-Manchego, F., Bingel, J., Paetzold, G., Scarton, C., and Specia, L. (2017). Learning how to simplify from explicit labeling of complex-simplified text pairs. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 295–305, Taipei,Taiwan.

Arenas, A. G. (2013). What do professional translators think about post-editing. *The Journal of Specialized Translation*, 19(19):75–95.

Avramidis, E. (2013). Sentence-level ranking with quality estimation. *Machine Translation*, 27:239–256.

Baeza-Yates, R. and Gonnet, G. H. (1992). A new approach to text searching. *Communications of the ACM*, 35(10):74–82.

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*, San Diego, California, USA.

Basak, D., Pal, S., and Patranabis, D. C. (2007). Support vector regression. *Neural Information Processing – Letters and Reviews*, 11(10):203–224.

Béchara, H., Ma, Y., and van Genabith, J. (2011). Statistical post-editing for a statistical mt system. In *Proceedings of the Thirteenth Machine Translation Summit*, pages 308–315, Xiamen, China.

Beeferman, D., Berger, A., and Lafferty, J. (1999). Statistical models for text segmentation. *Machine learning*, 34(1-3):177–210.

Bentivogli, L., Bisazza, A., Cettolo, M., and Federico, M. (2016). Neural versus phrase-based machine translation quality: a case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 257–267, Austin, Texas, USA.

Biçici, E. (2008). Consensus ontologies in socially interacting multiagent systems. *Multiagent and Grid Systems*, 4(3):297–314.

Biçici, E. and Dymetman, M. (2008). Dynamic translation memory: Using statistical machine translation to improve translation memory fuzzy matches. pages 454–465, Haifa, Israel.

Blatz, J., Fitzgerald, E., Foster, G., Gandrabur, S., Goutte, C., Kulesza, A., Sanchis, A., and Ueffing, N. (2004). Confidence estimation for machine translation. In *Proceedings of the Twentieth International Conference on Computational Linguistics*, pages 315–321, Geneva, Switzerland.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5(5):135–146.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., Soricut, R., Specia, L., and Tamchyna, A. (2014). Findings of the 2014 Workshop on Statistical Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huang, S., Huck, M., Koehn, P., Liu, Q., Logacheva, V., et al. (2017). Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Jimeno Yepes, A., Koehn, P., Logacheva, V., Monz, C., Negri, M., Neveol, A., Neves, M., Popel, M., Post, M., Rubino, R., Scarton, C., Specia, L., Turchi, M., Verspoor, K., and Zampieri, M. (2016). Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany.

Bookstein, A., Kulyukin, V. A., and Raita, T. (2002). Generalized hamming distance. *Information Retrieval*, 5(4):353–375.

Bowker, L. (2002). *Computer-aided translation technology: a practical introduction*. University of Ottawa Press.

Breiman, L. and Cutler, A. (2019). randomforest: Breiman and cutler's random forests for classification and regression. `https://www.stat.berkeley.edu/breiman/RandomForests/`, last accessed: 12th November 2019.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. (1984). *Classification and Regression Trees*. Taylor & Francis.

Bulté, B., Vanallemeersch, T., and Vandeghinste, V. (2018). M3TRA: integrating TM and MT for professional translators. In *Proceedings of the Twenty-First Annual Conference of the European Association for Machine Translation*, pages 69–78, Alacant, Spain.

Cadwell, P., Castilho, S., O'Brien, S., and Mitchell, L. (2016). Human factors in machine translation and post-editing among institutional translators. *Translation Spaces*, 5(2):222–243.

Carbonell, J. G., Klein, S., Miller, D., Steinbaum, M., Grassiany, T., and Frey, J. (2006). Context-based machine translation. In *Proceedings of the Seventh Conference of the Association for Machine Translation in the Americas*, pages 19–28, Cambridge, Massachusetts.

Carl, M., Gutermuth, S., and Hansen-Schirra, S. (2015). Post-editing machine translation. *Psycholinguistic and cognitive inquiries into translation and interpreting*, 115(115):145.

Carl, M. and Way, A., editors (2003). *Recent Advances in Example-Based Machine Translation*. Springer.

Chatterjee, R., C. de Souza, J. G., Negri, M., and Turchi, M. (2016). The FBK's Participation in the WMT 2016 Automatic Post-editing Shared Task. In *Proceedings of the First Conference on Machine Translation*, pages 745–750, Berlin, Germany.

Chatterjee, R., Farajian, M. A., Negri, M., Turchi, M., Srivastava, A., and Pal, S. (2017). Multi-source neural automatic post-editing: FBK's participation in the WMT 2017 APE shared task. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 630–638, Copenhagen, Denmark.

Chatterjee, R., Negri, M., Rubino, R., and Turchi, M. (2018a). Findings of the WMT 2018 Shared Task on Automatic Post-Editing. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 723–738, Belgium, Brussels.

Chatterjee, R., Negri, M., Turchi, M., Blain, F., and Specia, L. (2018b). Combining quality estimation and automatic post-editing to enhance machine translation output. In *Proceedings of the Thirteenth Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 26–38, Boston, Massachusetts, USA.

Chatterjee, R., Weller, M., Negri, M., and Turchi, M. (2015). Exploring the planet of the apes: a comparative study of state-of-the-art methods for MT automatic post-editing. In *Proceedings of the Fifty-Third Annual Meeting of the Association for Computational Linguistics and the Seventh International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 156–161, Beijing, China.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Costa-jussà, M. R. (2012). An overview of the phrase-based statistical machine translation techniques. *The Knowledge Engineering Review*, 27(4):413.

Cramer, J. S. (2002). The origins of logistic regression. 02(119):1–16.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176.

Dandapat, S., Morrissey, S., Way, A., and Forcada, M. L. (2011). Using example-based MT to support statistical MT when translating homogeneous data in a resource-poor setting. In *Proceedings of the Fifteenth conference of the European Association for Machine Translation*, pages 201–208. Leuven, Belgium.

Deng, Y., Kumar, S., and Byrne, W. (2007). Segmentation and alignment of parallel text for statistical machine translation. *Natural Language Engineering*, 13(3):235–260.

Di Gangi, M. A., Bertoldi, N., and Federico, M. (2017). FBK's participation to the English-to-German news translation task of WMT 2017. In *Proceedings of the Second Conference on Machine Translation*, pages 271–275, Copenhagen,Denmark.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(7):2121–2159.

Dugast, L., Senellart, J., and Koehn, P. (2007). Statistical post-editing on systran's rule-based translation system. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 220–223, Prague, Czech Republic.

Durrani, N., Sajjad, H., Joty, S., Abdelali, A., and Vogel, S. (2015). Using joint models for domain adaptation in statistical machine translation. pages 117–130, Miami, Florida, USA.

Escartín, C. P. and Arcedillo, M. (2015a). A fuzzier approach to machine translation evaluation: A pilot study on post-editing productivity and automated metrics in commercial settings. In *Proceedings of the Fourth Workshop on Hybrid Approaches to Translation (HyTra)*, pages 40–45, Beijing, China.

Escartín, C. P. and Arcedillo, M. (2015b). Machine translation evaluation made fuzzier: A study on post-editing productivity and evaluation metrics in commercial settings. In *Proceedings of the Fifteenth Machine Translation Summit*, pages 131–144, Miami, Florida, USA.

Espla-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(49):93–99.

Esplà-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(1):169–222.

Esplà-Gomis, M., Sánchez-Martínez, F., and Forcada, M. L. (2019). Predicting insertion positions in word-level machine translation quality estimation. *Applied Soft Computing*, 76(76):174–192.

Esplá-Gomis, M., Sánchez-Martínez, F., and Forcada, M. L. (2011). Using machine translation in computer-aided translation to suggest the target-side words to change. In *Proceedings of the Thirteenth Machine Translation Summit*, pages 172–179, Xiamen, China.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(8):1871–1874.

Farajian, M. A., Chatterjee, R., Conforti, C., Jalalvand, S., Balaraman, V., Di Gangi, M. A., Ataman, D., Turchi, M., Negri, M., and Federico, M. (2016). FBK's neural machine translation systems for IWSLT 2016. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 67–75, Seattle, Washington.

Federico, M., Cattelan, A., and Trombetti, M. (2012). Measuring user productivity in machine translation enhanced computer assisted translation. In *Proceedings of the Tenth Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 44–56, Madison, Wisconsin.

Forcada, M. L., Ginestó-Rosell, M., Nordfalk, J., O'Regan, J., Ortiz-Rojas, S., Pérez-Ortiz, J. A., Felipe Sánchez-Martínez, G. R.-S., and Tyers, F. M. (2011). Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144.

Freitag, M. and Al-Onaizan, Y. (2016). Fast domain adaptation for neural machine translation. *arXiv preprint arXiv:1612.06897*.

Freitag, M., Huck, M., and Ney, H. (2014). Jane: Open source machine translation system combination. In *Proceedings of the Demonstrations at the Fourteenth Conference of the European Chapter of the Association for Computational Linguistics*, pages 29–32, Gothenburg, Sweden.

Garmash, E. and Monz, C. (2016). Ensemble learning for multi-source neural machine translation. In *Proceedings the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1409–1418, Osaka, Japan.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63(1):3–42.

Geurts, P. and Louppe, G. (2011). Learning to rank with extremely randomized trees. In *Proceedings of Machine Learning Research, Volume 14: Proceedings of the Learning to Rank Challenge*, pages 49–61, Haifa, Israel.

González-Rubio, J. and Casacuberta, F. (2015). Minimum bayes' risk subsequence combination for machine translation. *Pattern Analysis and Applications*, 18(3):523–533.

Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422.

Heafield, K. and Lavie, A. (2010). Combining machine translation output with open source: The Carnegie Mellon multi-engine machine translation scheme. *The Prague Bulletin of Mathematical Linguistics*, 93(93):27–36.

Hewavitharana, S., Vogel, S., and Waibel, A. (2005). Augmenting a statistical translation system with a translation memory. In *Proceedings of the Tenth conference of the European Association for Machine Translatio*, pages 126–132, Carnegie Mellon University, Pittsburgh, USA.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hokamp, C. (2017). Ensembling factored neural machine translation models for automatic post-editing and quality estimation. In *Proceedings of the Second Conference on Machine Translation*, pages 647–654, Copenhagen, Denmark.

Hutchins, J. (1997). From first conception to first demonstration: the nascent years of machine translation, 1947–1954. a chronology. *Machine Translation*, 12(3):195–252.

Hutchins, J. (2007). Machine translation: A concise history. *Computer aided translation: Theory and practice*, 13(29-70):11.

Isabel, L. (2017). *Cognitive Effort in Translation, Editing, and Post-editing*, chapter 21, pages 386–401. Wiley-Blackwell.

Jakobson, R. (1959). On linguistic aspects of translation. *On translation*, 3(3):30–39.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the Fifteenth Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain.

Junczys-Dowmunt, M., Dwojak, T., and Hoang, H. (2016). Is neural machine translation ready for deployment? a case study on 30 translation directions. *arXiv preprint arXiv:1610.01108*.

Junczys-Dowmunt, M. and Grundkiewicz, R. (2016). Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. In *Proceedings of the First Conference on Machine Translation*, pages 751–758, Berlin, Germany.

Junczys-Dowmunt, M. and Grundkiewicz, R. (2017). An exploration of neural sequence-to-sequence architectures for automatic post-editing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, pages 120–129, Taipei,Taiwan.

Knowles, R. and Koehn, P. (2016). Neural interactive translation prediction. In *Proceedings of the Twelfth Conference of the Association for Machine Translation in the Americas*, pages 107–120, Austin, Texas, USA.

Knowles, R., Ortega, J. E., and Koehn, P. (2018). A comparison of machine translation paradigms for use in black-box fuzzy-match repair. In *Proceedings of the Thirteenth Conference of the Association for Machine Translation in the Americas*, pages 249–255, Boston, Massachusetts, USA.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the Tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand.

Koehn, P. (2010). *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (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*, ACL '07, pages 177–180, Prague, Czech Republic.

Koehn, P. and Knowles, R. (2017). Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver, Canada.

Koehn, P. and Senellart, J. (2010). Convergence of translation memory and statistical machine translation. In *Proceedings of AMTA Workshop on MT Research and the Translation Industry*, pages 21–31, Edinburgh, United Kingdom and Paris, France.

Kolen, J. F. and Kremer, S. C. (2001). *A field guide to dynamical recurrent networks*. John Wiley & Sons.

Kovstial, M. and Davrena, F. (2017). Using word embeddings for analysing texts from the educational domain. In *21st European Scientific Conference of Doctoral Students*, pages 129–136, Brno, Czech Republic.

Kranias, L. and Samiotou, A. (2004). Automatic translation memory fuzzy match post-editing: a step beyond traditional TM/MT integration. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, pages 331–334, Lisbon, Portugal.

Lavie, A. and Agarwal, A. (2007). Meteor: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *ACL 2007 Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic.

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of psychology*, 22(140):5–55.

Louppe, G., Wehenkel, L., Sutera, A., and Geurts, P. (2013). Understanding variable importances in forests of randomized trees. In *Proceedings of the Twenty-Sixth International Conference on Neural Information Processing Systems - Volume 1*, pages 431–439, Lake Tahoe, Michigan, USA.

Luong, M.-T. and Manning, C. D. (2015). Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 76–79, Da Nang, Vietnam.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2014). Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

Ma, Y., He, Y., Way, A., and van Genabith, J. (2011). Consistent translation using discriminative learning - a translation memory-inspired approach. In *Proceedings of the Forty-Ninth Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1239–1248, Portland, Oregon, USA.

Macherey, W. and Och, F. J. (2007). An empirical study on computing consensus translations from multiple machine translation systems. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 986–995, Prague, Czech Republic.

Marino, J. B., Banchs, R. E., Crego, J. M., de Gispert, A., Lambert, P., Fonollosa, J. A., and Costa-Jussà, M. R. (2006). N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549.

Matusov, E., Mauser, A., and Ney, H. (2006). Automatic sentence segmentation and punctuation prediction for spoken language translation. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 158–165, Kyoto, Japan.

Mendonça, A., Jaquette, D., Graff, D., and DiPersio, D. (2011). Spanish gigaword third edition LDC2011T12. Web Download. Philadelphia: Linguistic Data Consortium.

Mikheev, A. (2003). Text segmentation. In *The Oxford handbook of computational linguistics*, chapter 10.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2018). Advances in pre-training distributed word representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, Miyazaki, Japan.

Miłkowski, M. and Lipski, J. (2009). Using SRX standard for sentence segmentation. In *Proceedings of the fourth Language and Technology Conference (LTC 2009)*, pages 172–182, Poznan, Poland.

Negri, M., Turchi, M., Chatterjee, R., and Bertoldi, N. (2018). eSCAPE: a large-scale synthetic corpus for automatic post-editing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, Miyazaki, Japan.

Niehues, J., Cho, E., Ha, T.-L., and Waibel, A. (2016). Pre-translation for neural machine translation. *arXiv preprint arXiv:1610.05243*.

Nirenburg, S., Carbonell, J., Tomita, M., and Goodman, K. (1994). *Machine translation: A knowledge-based approach*. Morgan Kaufmann Publishers Inc.

Nomoto, T. (2004). Multi-engine machine translation with voted language model. In *Proceedings of the Forty-Second Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 494–501, Barcelona, Spain.

O'Brien, S., Ehrensberger-Dow, M., Hasler, M., and Connolly, M. (2017). Irritating CAT tool features that matter to translators. *Hermes: Journal of Language and Communication in Business*, 56(56):145–162.

O'Brien, S. (2011). Collaborative translation. *Handbook of translation studies*, 2(2):17–20.

Pal, S., Naskar, S. K., Vela, M., and van Genabith, J. (2016). A neural network based approach to automatic post-editing. In *Proceedings of the Fifty-Fourth Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 281–286, Berlin, Germany.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Fortieth Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA.

Parton, K., Habash, N., McKeown, K., Iglesias, G., and de Gispert, A. (2012). Can Automatic Post-Editing Make MT More Meaningful? In *Proceedings of the Sixteenth Annual Conference of the European Association for Machine Translation*, pages 111–118, Trento, Italy.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar.

Peter, J.-T., Alkhouli, T., Ney, H., Huck, M., Braune, F., Fraser, A., Tamchyna, A., Bojar, O., Haddow, B., Sennrich, R., et al. (2016). The QT21/HIML combined machine translation system. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 344–355, Berlin, Germany.

Peter, J.-T., Guta, A., Alkhouli, T., Bahar, P., Rosendahl, J., Rossenbach, N., Graça, M., and Ney, H. (2017). The RWTH Aachen university English–German and German–English machine translation system for WMT 2017. In *Proceedings of the Second Conference on Machine Translation*, pages 358–365, Berlin, Germany.

Pilevar, A. H. (2011). Using Statistical Post-editing to Improve the Output of Rule-based Machine Translation System. *International Journal of Computer Science and Communication*, 2(1):97–100.

Popović, M. (2015). chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal.

Rao, C. R. (1973). *Linear statistical inference and its applications*. John Wiley & Sons, Inc.

Rosti, A.-V., Ayan, N. F., Xiang, B., Matsoukas, S., Schwartz, R., and Dorr, B. (2007). Combining outputs from multiple machine translation systems. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 228–235, Rochester, New York, USA.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513 – 523.

Sánchez-Martínez, F. (2011). Choosing the best machine translation system to translate a sentence by using only source-language information. In *Proceedings of the Fifteenth Annual Conference of the European Association for Machine Translation*, pages 97–104, Leuven, Belgium.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Schwenk, H., Rousseau, A., and Attik, M. (2012). Large, pruned or continuous space language models on a GPU for statistical machine translation. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 11–19, Montreal, Canada.

Sennrich, R., Firat, O., Cho, K., Birch, A., Haddow, B., Hitschler, J., Junczys-Dowmunt, M., L"aubli, S., Miceli Barone, A. V., Mokry, J., and Nadejde, M. (2017). Nematus: a Toolkit for Neural Machine Translation. In *Proceedings of the Demonstrations at the Fifteenth Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the Fifty-Fourth Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany.

Sennrich, R. and Zhang, B. (2019). Revisiting low-resource neural machine translation: A case study. *arXiv preprint arXiv:1905.11901*.

Simard, M., Goutte, C., and Isabelle, P. (2007). Statistical Phrase-Based Post-Editing. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 508–515, Rochester, New York, USA.

Simard, M. and Isabelle, P. (2009). Phrase-based machine translation in a computer-assisted translation environment. In *Proceeding of the Twelfth Machine Translation Summit*, pages 120–127, Quebec, Canada.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of the Seventh Conference of the Association for Machine Translation in the Americas*, pages 223–231, Cambridge, Massachusetts, USA.

Somers, H. (2003). Translation memory systems. In Somers, H., editor, *Computers and Translation: a Translator's Guide*, pages 31–47. John Benjamins.

Specia, L. (2011). Exploiting objective annotations for measuring translation post-editing effort. In *Proceedings of the Fifteenth Annual Conference of the European Association for Machine Translation*, pages 73–80, Leuven, Belgium.

Specia, L., Paetzold, G., and Scarton, C. (2015). Multi-level translation quality prediction with QuEst++. In *Proceedings of the Fifty-Third Annual Meeting of the Association for Computational Linguistics and the Seventh International Joint Conference of the Asian Federation of Natural Language Processing - System Demonstrations*, pages 115–120, Beijing, China.

Specia, L., Turchi, M., Cancedda, N., Dymetman, M., and Cristianini, N. (2009). Estimating the sentence-level quality of machine translation systems. In *Proceedings of the Thirteenth Annual Conference of the European Association for Machine Translation*, pages 28–37, Barcelona, Spain.

Steijvers, M. and Grünwald, P. (1996). A recurrent network that performs a context-sensitive prediction task. In *Proceedings of the Eighteenth annual conference of the cognitive science society*, pages 335–339, San Diego, California, USA.

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., and Zeileis, A. (2008). Conditional variable importance for random forests. *BMC bioinformatics*, 9(1):307.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of Advances in Neural Information Processing Systems*, pages 3104–3112, Montreal, Canada.

Tan, Y., Chen, Z., Huang, L., Zhang, L., Li, M., and Wang, M. (2017). Neural post-editing based on quality estimation. In *Proceedings of the Second Conference on Machine Translation*, pages 655–660, Copenhagen, Denmark.

Terumasa, E. (2007). Rule Based Machine Translation Combined with Statistical Post Editor for Japanese to English Patent Translation. In *Proceedings of the Eleventh Machine Translation Summit*, pages 13–18, Copenhagen, Denmark.

Toral, A. and Sánchez-Cartagena, V. M. (2017). A multifaceted evaluation of neural versus phrase-based machine translation for 9 language directions. In *Proceedings of the Fifteenth Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1063–1073, Valencia, Spain.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems*, pages 5998–6008, Long Beach, California, USA.

Vukotic, V., Raymond, C., and Gravier, G. (2016). A step beyond local observations with a dialog aware bidirectional gru network for spoken language understanding. In *Proceedings of Interspeech 2016*, pages 3241–3244, San Francisco, California, USA.

Wagner, R. A. and Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173.

Wuebker, J., Green, S., DeNero, J., Hasan, S., and Luong, M.-T. (2016). Models and inference for prefix-constrained machine translation. In *Proceedings of the Fifty-Fourth Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Berlin, Germany.

Xu, J., Zens, R., and Ney, H. (2005). Sentence segmentation using ibm word alignment model 1. In *Proceedings of the Tenth Conference of the European Association for Machine Translation*, pages 280–287, Budapest, Hungary.

Yin, W., Kann, K., Yu, M., and Schütze, H. (2017). Comparative study of CNN and RNN for natural language processing. *arXiv preprint arXiv:1702.01923*.

Yu, C., Chu, B., Ram, R., Aichinger, J., Qu, L., and Suominen, H. (2016). Pairwise fasttext classifier for entity disambiguation. In *Proceedings of the Australasian Language Technology Association Workshop 2016*, pages 175–179, Melbourne, Australia.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhechev, V. and Genabith, J. V. (2010). Seeding statistical machine translation with translation memory output through tree-based structural alignment. In *Proceedings of the Fourth Workshop on Syntax and Structure in Statistical Translation*, pages 43–49, Dublin, Ireland.

Zwarts, S. and Dras, M. (2008). Choosing the right translation: A syntactically informed classification approach. In *Proceedings of the Twenty-Second International Conference on Computational Linguistics*, pages 1153–1160, Manchester, United Kingdom.