Comparative Quality Estimation for Machine Translation

An Application of Artificial Intelligence
on Language Technology
using Machine Learning of Human Preferences

Dissertation
zur Erlangung des akademischen Grades eines
Doktors der Philosophie
der Philosophischen Fakultät
der Universität des Saarlandes
vorgelegt von

Eleftherios Avramidis

aus Thessaloniki, Griechenland

Saarbrücken, 2019

Der Dekan der Philosophischen Fakultät: Prof. Dr. Heinrich Schlange-Schöningen

Berichterstater/innen: Prof. Dr. Hans Uszkoreit, Prof. Dr. Josef van Genabith, Prof. Dr. Lucia Specia

Tag der letzten Prüfungsleistung: 4.12.2018

Abstract

In this thesis we focus on Comparative Quality Estimation, as the automatic process of analysing two or more translations produced by a Machine Translation (MT) system and expressing a judgment about their comparison. We approach the problem from a supervised machine learning perspective, with the aim to learn from human preferences. As a result, we create the ranking mechanism, a pipeline that includes the necessary tasks for ordering several MT outputs of a given source sentence in terms of relative quality.

Quality Estimation models are trained to statistically associate the judgments with some qualitative features. For this purpose, we design a broad set of features with a particular focus on the ones with a grammatical background. Through an iterative feature engineering process, we investigate several feature sets, we conclude to the ones that achieve the best performance and we proceed to linguistically intuitive observations about the contribution of individual features.

Additionally, we employ several feature selection and machine learning methods to take advantage of these features. We suggest the usage of binary classifiers after decomposing the ranking into pairwise decisions. In order to reduce the amount of uncertain decisions (ties) we weight the pairwise decisions with their classification probability.

Through a set of experiments, we show that the ranking mechanism can learn and reproduce rankings that correlate to the ones given by humans. Most importantly, it can be successfully compared with state-of-the-art reference-aware metrics and other known ranking methods for several language pairs. We also apply this method for a hybrid MT system combination and we show that it is able to improve the overall translation performance.

Finally, we examine the correlation between common MT errors and decoding events of the phrase-based statistical MT systems. Through evidence from the decoding process, we identify some cases where long-distance grammatical phenomena cannot be captured properly.

An additional outcome of this thesis is the open source software Qualitative, which implements the full pipeline of ranking mechanism and the system combination task. It integrates a multitude of state-of-the-art natural language processing tools and can support the development of new models. Apart from the usage in experiment pipelines, it can serve as an application back-end for web applications in real-use scenaria.

Ausführliche Zusammenfassung

In dieser Promotionsarbeit konzentrieren wir uns auf die Vergleichende Qualitätsschätzung der Maschinellen Übersetzung. Die Vergleichende Qualitätsschätzung ist eine eigenständige Art der Qualitätsschätzung, die durch die Möglichkeit motiviert ist, die Objektivität von Übersetzungsvergleichen zu verbessern, wobei die Anforderungen an die Grundwahrheit reduziert werden, indem die Ordinalität der Kardinalität vorgezogen wird. Dieses Konzept wird gegensätzlich zu anderen Formen der Qualitätsschätzung definiert, welche wir Absolute Qualitätsschätzung nennen, da sie sich auf die Schätzung der Qualität einzelner Ergebnisse in Form von numerischen Bewertungen konzentrieren.

Wir sehen Vergleichende Qualitätsschätzung als ein automatisches Verfahren zur Analyse von zwei oder mehr Übersetzungen, die von Maschinenübersetzungssysteme erzeugt wurden, und zur Beurteilung von deren Vergleich. Wir gehen an das Problem aus der Perspektive des überwachten Maschinellen Lernens heran, mit dem Ziel, von menschlichen Präferenzen zu lernen. Als Ergebnis erstellen wir einen Ranking-Mechanismus. Dabei handelt es sich um eine Pipeline, welche die notwendigen Arbeitsschritte für die Anordnung mehrerer Maschinenübersetzungen eines bestimmten Quellsatzes in Bezug auf die relative Qualität umfasst.

Methoden

Die Dissertation besteht aus einer Reihe von empirischen Experimenten. Bevor wir die Durchführung und die Ergebnisse dieser Experimente erläutern, stellen wir den methodischen Hintergrund dar, auf dem die Experimente basieren. Wir formulieren daher zunächst die Konzepte der Vergleichenden Qualitätsschätzung, einschließlich der zugrunde liegenden Konzepte des Rankings, der relativen Darstellung von Qualitätsurteilen, der Unentschieden und des Ranking-Mechanismus. Nach dem Einführen des Konzepts des Ranking-Mechanismus, erläutern wir die Details seiner Funktionalität als eine typische Anwendung des überwachten Maschinellen Lernens, bestehend aus der Merkmalskonstruktion und den Algorithmen des Maschinellen Lernens.

Qualitätsschätzungsmodelle werden mit Maschinellem Lernen trainiert, um Vergleichsurteile mit einigen bestimmten Merkmalen statistisch zu verknüpfen. Zu diesem Zweck konzipieren wir eine breite Palette von Merkmalen mit besonderem Fokus auf diejenigen mit einem grammatikalischen Hintergrund. Zusätzlich setzen wir verschiedene Methoden

der Merkmalsauswahl und des Maschinellen Lernens ein, um die Vorteile dieser Merkmale zu nutzen.

Wir schlagen die Verwendung von binären Klassifikatoren nach Zerlegen des Rankings in paarweise Entscheidungen vor. Die Rekomposition eines Rankings aus paarweisen binären Klassifikatorentscheidungen steht vor dem Problem, dass Unentschieden aus unklaren und widersprüchlichen paarweisen Entscheidungen resultieren. Um die Anzahl der Unentschieden zu verringern, gewichten wir die paarweisen Entscheidungen mit deren Klassifikationswahrscheinlichkeit, bevor sie aggregiert werden, um vollständige Rankings zu erstellen (Soft-Rank-Rekomposition).

Die Leistung des automatischen Rankings wird gegen menschliche Rankings gemessen. Das Interesse liegt dabei darin, ob es einen Zusammenhang zwischen dem automatischen Ranking und den Beurteilungen eines Menschen gibt. Wenn eine solche Beziehung besteht, sollte es möglich sein, den Grad der Übereinstimmung zwischen diesen beiden Rankings zu messen. Dazu wird dem Ranking-Mechanismus ein Testsatz gegeben und die Korrelation der von ihm erzeugten Rankings (eins pro Satz) zu den ursprünglichen menschlichen Rankings gemessen. Zur Bewertung der Leistungsfähigkeit verwenden wir die Korrelationsmetriken Kendalls Tau und Non-Discounted Cumulative Gain (NDCG). Bei der Bewertung gehen wir davon aus, dass die Testsätze keine Unentschieden enthalten, da Unentschieden als unsichere Proben angesehen werden, die nicht für die Bewertung verwendet werden können.

Vergleichende Qualitätsschätzung für das Ranking

Mithilfe einer Reihe von Experimenten zeigen wir, dass der Ranking-Mechanismus Rankings lernen und reproduzieren kann, die mit denen von Menschen übereinstimmen. Weitere Vergleiche zeigen, dass die Leistungsfähigkeit des Mechanismus besser als alle Baselines ist. Die wichtigste Erkenntnis ist, dass der Mechanismus erfolgreich mit referenzbasierten Metriken und anderen bekannten Ranking-Methoden auf dem neusten Stand der Technik für verschiedene Sprachpaare verglichen werden kann. Der Ranking-Mechanismus ist auch besser als andere Metriken in Sprachpaaren, bei denen die Merkmalskonstruktion von anderen Sprachpaaren übernommen wurde, abgesehen von einer Referenzmetrik, METEOR, die mit dem Ranking-Mechanismus vergleichbar ist. Diese Ergebnisse deuten darauf hin, dass aufwendige Merkmale und Maschinelles Lernen mehr Informationen über die relative Übersetzungsqualität liefern können als der direkte Vergleich mit Referenzen.

Die Qualitätsschätzmodelle werden auf der Grundlage von menschlichen Rankings trainiert, die sich aus der Shared Task zur maschinellen Übersetzung ergeben, die im Rahmen der Workshops zur Statistischen Maschinellen Übersetzung (WMT2008-2014) organisiert wurde. Diese Daten enthalten Tausende von Sätzen, die von allen an der Aufgabe beteiligten Systemen der maschinellen Übersetzung übersetzt wurden. Diese Übersetzungen werden manuell von menschlichen Annotatoren bewertet, die die maschinellen Übersetzer nach der Qualität ihren Übersetzungen einstufen.

Der komplette Fokus des Ranking-Mechanismus liegt auf der Übersetzungsrichtung von Deutsch nach Englisch und sekundär von Englisch nach Deutsch. Die erfolgreichen Einstellungen aus diesen Sprachrichtungen wurden mit minimalem zusätzlichen Aufwand auf weitere vier Sprachrichtungen angewendet: von Französisch nach Englisch, von Englisch nach Französisch, von Spanisch nach Englisch und von Englisch nach Spanisch. Die Modelle für jede Sprachrichtung werden separat trainiert, basierend auf den entsprechenden sprachspezifischen Daten, Labels und Merkmalen.

Maschinelles Lernen

Die Tatsache, dass das Ranking in paarweise Entscheidungen zerlegt wurde, ermöglicht die Integration mehrerer maschineller Lernalgorithmen mit positiven Ergebnissen. Die leistungsfähigsten Algorithmen sind Gradient Boosting, Logistic Regression mit Stepwise Feature Set Selection, AdaBoost, Linear Discriminant Analysis und Gaussian Naïve Bayes. Nur ein einziger maschineller Lernalgorithmus, die einfachen Entscheidungsbäume, erreichen keine signifikante Korrelation mit den menschlichen Einstufungen. Ensemble-Klassifikatoren, die auf Entscheidungsbäumen aufgebaut sind, schneiden deutlich besser ab als die einzelnen Entscheidungsbäume.

Logistic Regression mit Stepwise Feature Set Selection wird für den in dieser Arbeit vorgestellten grundlegenden Ranking-Mechanismus ausgewählt, weil hierbei die gute Leistungsfähigkeit mit der Fähigkeit zur Interpretation des statistischen Modells kombiniert wird. Der am höchsten bewertete Algorithmus, Gradient Boosting, wird für den fortgeschrittenen Ranking-Mechanismus gewählt.

Durch die Experimente kommen wir auch zu dem Schluss, dass die Methode zur Reduzierung von Unentschieden, die als Soft-Rank-Rekomposition bezeichnet wird, unterschiedliche Auswirkungen auf die Rankingvorhersage haben kann, wenn sie auf verschiedene Algorithmen angewendet wird. Sie hat einen positiven Einfluss auf die meisten der untersuchten Algorithmen, einschließlich der drei leistungsfähigsten Algorithmen. Bei nur zwei Algorithmen (Linar Discriminant Analysis und Gaussian Naïve Bayes) ergab sich ein kleiner, aber nicht statistisch signifikanter Verlust bezüglich der Korrelation mit menschlichen Beurteilungen.

Es wird ebenfalls empirisch bestätigt, dass ein Zusammenhang zwischen dem Auftreten von Unentschieden und dem Unterschied in der Gesamtleistung der zu vergleichenden maschinellen Übersetzungssysteme besteht. Wenn der BLEU-Unterschied zwischen zwei Systemen im Durchschnitt hoch ist, führen paarweise Vergleiche zu einer kleinen Anzahl von Unentschieden, wobei die Anzahl der Unentschieden für Vergleiche zwischen Systemen erhöht wird, wenn sie einen geringeren Unterschied in BLEU haben.

Merkmalskonstruktion

Mit Hilfe eines iterativen Verfahrens der Merkmalskonstruktion untersuchen wir verschiedene Merkmalsreihen, erschließen diejenigen, die die beste Leistung erzielen, und leiten linguistisch motivierte Beobachtungen über die Beiträge der einzelnen Merkmale ab. Um die Relevanz grammatikalischer Merkmale zu testen, verwenden wir Funktionen aus dem PCFG-Parsing, die darauf hinweisen, dass das Parsen des Textes von automatisch generierten Ausgaben ein gutes Maß an Fluency liefern kann. Dazu gehören die Wahrscheinlichkeit der automatischen grammatischen Analyse des Satzes, die Anzahl der alternativen grammatische Analysen sowie die Bezeichnungen der Baumknoten. Angelehtn an die Annahme des Isomorphismus verwenden wir auch die Ausrichtungen der Baumknoten und CFG-Regeln zwischen der Quelle und der Übersetzung.

Das Verfahren der Merkmalskonstruktion wurde in drei Phasen durchgeführt. In der ersten Phase (Vorphase) wird ein grundlegender Satz von Merkmalen identifiziert, welche mit einem einfachen Klassifikator trainiert werden, um Rankings vorherzusagen, die eine signifikante Korrelation mit menschlichen Rankings aufweisen. Zudem wird gezeigt, dass grammatikalische Merkmale nützlicher als die von einem Sprachmodell generierte Merkmale sein können.

In der zweiten Phase (Grundphase) wurde es eine signifikante Verbesserung der Korrelation im Vergleich zur Vorphase erreicht. Die Verbesserung erfolgt durch das Hinzufügen von kontrastierenden Scores und anderen Merkmalen der absoluten Qualitätsschätzung in einem Modell, das mit Logistic Regression trainiert wurde. Wir bestätigen, dass die grundlegenden Merkmale vom Quellsatz nicht zum Übersetzungsvergleich beitragen und dass Logistic Regression diese Merkmale besser mit SFSS als mit L2-Regularisierung handhabt. Schließlich untersuchen wir die Modellkoeffizienten, um ein Eindruck der Beiträge jedes Merkmals zu erhalten.

In der letzten Phase (Vertiefungsphase) wird eine Vielzahl von Merkmalen zu Grammatik, Alignment und Position in zwei Sprachrichtungen eingeführt, wobei wird eine Verbesserung erreicht, wenn sie mit einem Gradient-Boosting-Klassifikator trainiert werden. Das Verfahren der Merkmalsauswahl der Recursive Feature Elimination hat efolgreich die Anzahl der Merkmalen auf ein Drittel verringert, ohne nennenswerten Leistungsminderung.

Die dominantesten Merkmale für das Modell der Logistic Regression sind die Anzahl der unbekannten Wörter, die Anzahl aller Wörter sowie die kontrastierenden Scores, zusammen mit einigen grammatikalischen Merkmalen, wie z.B. die Anzahl der Verbsätze und der alternativen automatischen Grammatikanalysen. Oberflächliche Merkmale des Quellsatzes sind für die Zwecke des Vergleichs nicht sinnvoll. Gleichwohl scheinen Merkmale der Ausrichtung bestimmter Baumknoten zwischen dem Quell- und Zielsatz sowie Alignment-Scores (IBM-Modell 1) wichtig zu sein. Zusätzlich hängt die Wahl der zielseitigen grammatikalischen Merkmale von der Sprachrichtung ab. Für die Übersetzung von Englisch nach Deutsch sind Merkmale wichtig, die die Position der Verbsätze im Satz markieren, während für die Übersetzung von Deutsch nach English Merkmale wie die Anzahl der untergeordneten Infinitiven und Präpositionsphrasen wichtiger zu sein scheinen.

Vergleichende Qualitätsschätzung für Systemkombinationen

Neben der Funktionalität der Methode als vollwertiger Ranking-Mechanismus untersuchen wir eine mögliche Anwendung: die Möglichkeit, das Ranking für eine hybride Systemkombination auf die Ausgabe verschiedener maschineller Übersetzer anzuwenden. Hierbei wenden wir den Ranking-Mechanismus auf den Output von drei verschiedenen Systemen an und zeigen, dass er erfolgreich eine Systemkombination erzeugen kann, die besser ist als jede der drei Komponenten.

Da die Systemkombination mit dem gesamten Satz als Kombinationseinheit arbeitet, hat sie im Vergleich zu fortgeschrittenen Kombinationsmethoden eine relativ geringe Granularität. Dennoch werden die Entscheidungen auf dem vollständigen Satz getroffen, weshalb die Möglichkeit besteht, informative Merkmale zu nutzen, die von einer vollständigen syntaktischen Struktur abhängen, z.B. grammatikalische Merkmale, die nicht aus Satzteilen extrahiert werden konnten. Des Weiteren ist die hier vorgestellte Kombination eher hybrid in dem Sinne, dass sie drei Systeme aus zwei verschiedenen maschinellen Übersetzungstechnologien kombiniert: Regelbasierte Maschinelle Übersetzung und Statistische Maschinelle Übersetzung. Wir gehen davon aus, dass eine Selektionsmethode, die den vollen Satz berücksichtigt, daher vorteilhaft sein kann, um von den Fernabhängigkeitsregeln des regelbasierten Systems zu profitieren. Darüber hinaus können sich die beiden Arten von Systemen in vielerlei Hinsicht ergänzen.

Die verfügbaren maschinellen Übersetzungen für jeden Satz des Testsets werden mit dem Ranking-Mechanismus geordnet und die Übersetzung mit dem höchsten Rang wurde für die Ausgabe der Systemkombination ausgewählt. Mit der Systemkombination ist es gelungen, eine kombinierte Ausgabe zu erzeugen, die deutlich höhere automatische metrische Werte auf Dokumentebene als jede ihrer Komponenten aufweist. Die Auswahl führt auch dann noch zu ähnlichen Leistungsverbesserungen, wenn maschinelle Übersetzungen aus einer minderwertigen Komponente enthalten sind.

Verschiedene Konfigurationen von Merkmalsets und maschinellen Lernmethoden werden getestet, wobei die besten Ergebnisse mit einem erweiterten, syntaxbewussten Merkmalset erzielt werden, der mit Gradient Boosting trainiert wird, wobei die Ranking-Labels mit dem rgbF-Score, gemessen an den Referenzübersetzungen, erstellt werden. Insgesamt ist trotz der Unterschiede im Experimentaufbau derjenige Konfiguration des maschinellen Lernalgorithmus, der die beste Leistung liefert, der des generischen Ranking-Experiments ähnlich.

Eine interessante Beobachtung ist, dass das Merkmal der kontrastiven Referenzmetrik METEOR einen negativen Beitrag zur Leistung der Systemkombination leistet, obwohl es in den generischen Ranking-Experimenten nützlich war. Dies kann mit dem Unterschied im Experimentaufbau zusammenhängen, da dieser eine geringere Anzahl von maschinellen

Übersetzungen pro Satz und unterschiedliche Arten von Systemen beinhaltet. Im Gegensatz zum generischen Ranking-Modell, bei dem das Sprachmodell einen negativen oder gar keinen positiven Effekt hat, hat hier das Sprachmodell zudem einen hohen Anteil an der Systemkombination.

Übersetzungsfehler und Dekodierungsereignisse

Abschließend untersuchen wir den Zusammenhang zwischen häufig vorkommenden Fehlern der maschinellen Übersetzung und Vorgängen, die während des internen Dekodierungsverfahrens der phrasenbasierten Statistischen Maschinellen Übersetzer ablaufen. Von professionellen Übersetzern post-editierte Übersetzungen wurden verwendet, um Fehler anhand der Bearbeitungsdistanz automatisch zu identifizieren. Binäre Klassifikatoren, die die Existenz eines Fehlers im Satz voraussagen, werden mit einer Logistic Regression trainiert, die Merkmale aus dem Dekodierungs-Suchdiagramm verwendet. Die Modelle sind für zwei gängige Fehlertypen und drei Sprachpaare in beide Richtungen mit zufriedenstellender Genauigkeit und Trefferquote trainiert.

Wir konzentrieren uns auf die Bereitstellung von statistischen Nachweisen darüber, wie die Interna des phrasenbasierten Dekodierungsverfahrens der Statistischen Maschinellen Übersetzung mit der Existenz von zwei häufigen Fehlerarten korrelieren, nämlich den Umordnungsfehlern und den fehlenden Wörtern. Durch die Gruppierung der Beobachtungen nach Fehlertyp zeigen wir wichtige Merkmale (welche Stufen des Dekodierungsverfahrens darstellen), die für mehrere Sprachpaare und Richtungen gleichzeitig gelten. Anhand der statistisch signifikanten Koeffizienten werden Indikationen und Tendenzen aufgezeigt.

Zwischen Umordnungsfehlern und unbekannten Wörtern besteht eine positive Korrelation, insbesondere, wenn sie über den Quellsatz verteilt sind. Des Weiteren existiert eine positive Korrelation zur Länge des Quellsatzes sowieso wenn die Zielphrasen länger sind als ihre jeweiligen Quellphrasen. Die letzten Korrelation wird den Fällen zugeschrieben, in denen deutsche Strukturen typischerweise länger als ihre Übersetzungen sind (z.B. Perfekt).

Fehlende Wörter haben eine negative Korrelation zur Länge des Quellsatzes und eine positive Korrelation zur Länge der Spanen aus dem Quellsatz, die der Decoder verwendet. Bemerkenswert ist unter anderem, dass die unbekannten Wörter einen negativen Zusammenhang mit der Wahrscheinlichkeit für fehlende Wörter aufweisen. Eine manuelle Untersuchung zeigt, dass ein unbekanntes Wort am Ende des Satzes verhindern kann, dass der Decoder eine falsch ausgerichtete Phrase wählt, die das deutsche untergeordnete Verb auslässt.

Quelloffene Software für die Qualitätsschätzung

Ein weiteres Ergebnis dieser Arbeit ist die quelloffene Software für die Qualitätsschätzung, die hauptsächlich für die Vergleichende Qualitätsschätzung und das automatische Ranking von Übersetzungen auf Satzebene entwickelt wurde. Die Software implementiert eine Pipeline, um die Übersetzungen mit sogenannten Black-Box-Merkmalen zu annotieren und den Algorithmus des Maschinellen Lernens anzuwenden. Ziel ist es, vorgegebene Testsets auf Grundlage vortrainierter Modelle zu bewerten oder neue Modelle zu trainieren und zu bewerten. Optional kann die Pipeline auch Übersetzungen von externen maschinellen Übersetzern abrufen und direkte Systemkombinationen auf Satzebene durchführen.

Die Merkmalskonstruktion beinhaltet Unterstützung für Sprachmodelle, PCFG-Parsing mit 2 Parsern, Sprachprüfungstools und verschiedene andere Präprozessoren und Merkmalsgeneratoren. Der Code folgt den Prinzipien der objektorientierten Programmierung, um Modularität und Erweiterbarkeit zu ermöglichen. Er integriert 25 hochmoderne externe Anwendungen in einem einzigen Python-Programm durch ein interoperables Framework mit 9 verschiedenen Integrationsansätzen. Das Tool kann sowohl Batch-Dateien als auch einzelne Sätze verarbeiten. Für die Anbindung von Webservices steht eine XML-RPC-Schnittstelle zur Verfügung.

Experimente, die das Training neuer Modelle beinhalten, sind so aufgebaut, dass eine umfangreiche Untersuchung mehrerer Experimente parallel möglich ist. Die Experimente können im Falle einer unerwarteten Unterbrechung wiederaufgenommen werden. Die Experimentpipeline führt ein strukturiertes Protokoll über jeden Schritt des Experiments, das die Ergebnisse der Auswertung, aber auch Details über das Verfahren des Maschinellen Lernens (z.B. die Beta-Koeffizienten eines loglinearen Modells oder die Gewichte eines linearen Modells) enthalten kann. Die trainierten Modelle werden zudem in externe Dateien ausgelagert, so dass sie später wiederverwendet werden können. Nachdem alle Iterationen und Cross-Validierungsfalten des Experiments abgeschlossen sind, ermöglicht ein Skript die Erstellung einer kommagetrennten Tabelle, die alle experimentellen Einstellungen mit einem gewünschten Metrikset vergleicht.

Weitere Arbeit

Ziel dieses Forschungsansatzes bleibt es, die Forschung zur Qualitätsschätzung und die Lösung gemeinsamer Probleme in der Praxis einander näher zu bringen. Es gibt Berichte, dass die Qualitätsschätzung in mehrere professionelle und industrielle Übersetzungspipelines als zusätzliche Bewertungs- oder Auswahlkomponente zusätzlich zu den bestehenden Übersetzungssystemen integriert wurde. Dennoch ergänzt der Einsatz von Qualitätsschätzung eindeutig die fehlenden Fähigkeiten der zugrundeliegenden Übersetzungssysteme. Ein weiteres offensichtliches Ziel wäre also die Integration der Qualitätsschätzung in den gesamten Design- und Entwicklungszyklus der Systeme oder sogar als inhärenter Bestandteil der Systemarchitektur selbst. Es wurden Anstrengungen hierzu unternommen, aber es wurde keine optimale Lösung erzielt. Darüber hinaus wurden trotz der positiven

Ergebnisse der letzten 7 Jahre (von denen einige auch in dieser Arbeit enthalten sind) nur geringe Fortschritte bei der Verwendung der Qualitätsschätzung als Bewertungsinstrument erzielt. Dies geht natürlich Hand in Hand mit der noch andauernden Nutzung von BLEU durch die Forschungsgemeinschaft, obwohl seine Fehler betont wurden und verbesserte Metriken zur Verfügung standen. Wir können davon ausgehen, dass die beobachtete Situation darauf zurückzuführen ist, dass der technische Aufwand für die Konfiguration und Durchführung der Qualitätsschätzung oft nicht durch den potenziellen Gewinn gerechtfertigt ist. Der Overhead selbst ist auf viele Aspekte der Qualitätsschätzung zurückzuführen, die sprachspezifisch oder systemorientiert sind und oft detailgenaue Arbeit für die Merkmalskonstruktion sowie andere notwendige Schritte erfordern.

Nachdem die Experimente dieser Arbeit abgeschlossen waren, entwickelte sich die Neurale Maschinelle Übersetzung, was zu einer signifikanten Veränderung des Forschungsfeldes führte. Trotz der offensichtlichen Verbesserungen zeigt die qualitative Analyse, dass noch immer Fehler sichtbar sind, was darauf hindeutet, dass die Rolle der Qualitätsschätzung in mehreren Anwendungen relevant sein kann. Das gesamte Spektrum der maschinellen Übersetzungsfehler verschiebt sich jedoch, da die Ausgabe der maschinellen Übersetzung immer weniger von menschlichen Übersetzungen unterscheidbar wird. Die generierten Sätze sind deutlich flüssiger und befolgen grammatikalische und syntaktische Regeln besser als statistische Systeme. Bei ausreichenden Daten würde ein Qualitätsschätzungsmodell eine verbesserte maschinelle Lernfähigkeit und technische Leistungsfähigkeit benötigen, um dieses neue Fehlerspektrum zu bewältigen und die Qualität besser zu differenzieren.

Die wichtigste Änderung ist jedoch auf der konzeptionellen Ebene absehbar. Die neue Lernmethode hat erfolgreich verstreute Module und Ad-Hock-Lösungen durch ein einheitliches, in sich geschlossenes Konzept ersetzt, das als einer der Hauptvorteile gesehen wird. So gesehen steht die genaue Form der Qualitätsschätzung auf dem Spiel: Wird die Qualitätsschätzung als eigenständiges Modul weiterhin nützlich sein, oder werden neuronale maschinelle Übersetzungsmodelle irgendwann in der Lage sein, die Qualitätsschätzung auf ihrem Output durchzuführen? Können die Ergebnisse der Qualitätsschätzung oder sogar die Qualitätsschätzung selbst effektiv in die Lernmethode integriert werden?

Unter Berücksichtigung dieser Aspekte sind wir der Meinung, dass zukünftige Arbeit die Rolle der Qualitätsschätzung angesichts der Einführung der Neuronalen Maschinellen Übersetzung bewerten und neu definieren sollte und die Verwendung von Ansätzen untersuchen sollte, die neuronale Netze für die Zwecke der Qualitätsschätzung verwenden sowie umgekehrt.

Acknowledgments

Special thanks to the Professors Hans Uszkoreit for the supervision and Josef van Genabith and Lucia Specia for agreeing to be official reviewers of this dissertation. Also thanks to Aljoscha Burchardt, Vivien Macketanz, Maja Popović and David Vilar for their support and their valuable comments; to Prof. Philipp Koehn for supervising my MSc thesis and my first publication which set the path for my research on Machine Translation; to Prof. Gesche Joost for the understanding and the support in the last phase, prior to the submission of the dissertation.

Additionally I would like to thank my colleagues José Camargo de Souza, Tracy Yu Chen, Trevor Cohn, Andreas Eisele, Alexandra Elbakyan, Christian Federmann, Stella Grylia, Frank Harrell, Kim Harris, Hieu Hoang, Sabine Hunsicker, Valia Kordoni, Arle Lommel, Stephan Peitz, Slav Petrov, Daniele Pighin, Lukas Poustka, Sven Schmeier, Cindy Tscherwinka, Sarah Weichert and Glen B. for their contribution in several stages of the PhD.

Most importantly thanks to friends and relatives who supported me along the way including (but not limited to): Stavroula Avramidou, Christos Avramidis, Nikolaos Kachrimanis, Anna Anagnostopoulou, Maria Amenta, Yannis Kalantidis, Eleni Kaouna, Olia Lizou, Patima Pataramekin, Amy Roumbou, Nikiforos Zacharof.

The work is devoted to the Community Radio 1431AM and the Independent Media Center network, whose vision for direct non-mediated access to information and news in various languages from all around the world inspired my engagement with Machine Translation in the first place.

Contents

List of Figures

List of Tables

Acronyms

AdaBoost Adaptive Boosting. 32, 50, 54, 55, 84, 86, 87, 108, 115

BLEU Bilingual Evaluation Understudy. xvii, xx, 2, 3, 25, 38, 49, 52, 53, 69, 74, 80, 81, 84, 87, 90, 107, 109, 111, 115, 116

CCG Combinatory Categorial Grammar. 8, 11, 47

CE Confidence Estimation. 3, 11

CFG Context-Free Grammar. 10, 22, 23, 68, 69, 71, 73, 74, 80, 82, 85

DT determiner. 22, 42, 74

ExtRa Trees Extra Randomized Trees. 50, 54, 86, 87

Gaussian NB Gaussian Naïve Bayes. 50, 54, 55, 86, 87, 108

HTER Human-targeted Translation Error Rate. 6, 14, 15

kNN k-Nearest Neighnoors. 30, 50, 54, 87, 108

LDA Linear Discriminat Analysis. 31, 50, 54, 55, 84, 86, 87, 108, 115

len sentence length. 25, 60, 66, 67, 94, 101

LM Language Model. xx, 10, 21, 24, 26, 38, 42, 58, 59, 60, 64, 66, 67, 69, 73, 79, 80, 82, 85, 86, 92, 94, 96, 98, 103, 107, 114, 116

LogReg Logistic Regression. 30, 50, 54, 73

ML Machine Learning. 4, 5, 7, 8, 9, 10, 11, 17, 28, 30, 41, 45, 57, 58, 59, 80, 89, 90, 108

MT Machine Translation. xvii, xix, xx, xxi, 1, 2, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 21, 24, 27, 28, 34, 41, 42, 43, 47, 50, 52, 53, 55, 58, 62, 76, 77, 78, 79, 80, 84, 85, 88, 89, 90, 103, 105, 106, 111, 113, 114, 116, 117

NDCG Normalised Discounted Cumulative Gain. xx, 34, 37, 50, 51, 52, 54, 70, 109

NLP Natural Language Processing. 12, 104, 105, 109, 111

NN noun. 22, 23, 42, 74

NP noun phrase. 22, 23, 22, 42, 58, 59, 60, 61, 68, 69, 71, 74

PCFG Probabilistic Context-Free Grammar. 10, 11, 21, 22, 23, 38, 58, 61, 77, 80, 82, 85, 103, 107, 111, 113

PP prepositional phrase. 22, 42, 59, 74

QE Quality Estimation. xix, xx, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19, 21, 22, 24, 25, 39, 41, 42, 46, 47, 57, 60, 61, 62, 63, 64, 65, 67, 69, 73, 82, 98, 103, 104, 105, 107, 114, 115, 116, 117

RBF Radial Basis Function. 7, 31, 47

RBMT Rule-based Machine Translation. xvii, 24, 76, 78, 79, 80, 81, 107, 116

RFECV Recurcive Feature Elimination with Cross-Validation. xvii, xx, 33, 42, 57, 70, 71, 70, 73, 108

S sentence. 22, 23, 22, 42, 69, 71, 74, 82, 85

SFSS Stepwise Feature Set Selection. 31, 33, 42, 47, 50, 51, 54, 55, 65, 69, 70, 84, 90, 94, 108, 116

SMT Statistical Machine Translation. xvii, xxi, 9, 10, 11, 13, 14, 15, 25, 64, 76, 77, 78, 79, 80, 81, 84, 85, 86, 85, 87, 89, 101, 107, 116

SVM Support Vector Machine. 7, 31, 34, 47, 57, 70, 86, 87, 108

TER Translation Error Rate. 2, 38, 49, 80, 81, 109, 115

unk unknown words. 25, 59, 60, 66, 82, 85, 96, 98

VB verb. 22, 23, 82, 85

VBG verb gerund. 82, 85

VBZ verb, 3rd person singular present. 82, 85

VP verb phrase. 22, 23, 22, 23, 42, 58, 59, 60, 61, 66, 67, 68, 69, 71, 74, 82, 85, 116

WER Word Error Rate. 2, 38, 49, 90, 107, 109

WMT Workshop on Statistical Machine Translation. xix, 12, 35, 42, 43, 46, 47, 59, 62, 63, 78, 80, 82, 91, 92, 107

WSD Word-Sense Disambiguation. 107

Chapter 1

Introduction

This thesis presents a method for automatically ranking several machine translation outputs for one given source sentence according to their comparative quality. In order to explain what we mean by this, we will motivate core concepts and ingredients in the following sections. Further below, we will provide an introduction to the concepts related to our main topic (Section 1.1). Then, we will set our work in perspective to the state of the art (Section 1.2), present our scientific goals (Section 1.3), list our main publications and contributions (Section 1.4) and finally explain the structure of this thesis (Section 1.5).

1.1 Introduction to Comparative Quality Estimation

1.1.1 Machine Translation

Machine Translation (MT) has been defined as *the now traditional and standard name for computerised systems responsible for the production of translations from one natural language into another, with or without human assistance* (Hutchins and Somers, 1992). It has long been considered as one of the most active fields of computational linguistics and also recently of machine learning.

The first experiments in the field started in the 1950s as the first non-numerical application of computers (Hutchins, 2003). Fighting through the initial enthusiasm and the consequent disappointment, the relevant research had a long way until the 1980's, when the development of the micro-computers allowed for a broader development, mostly based on *rule-based systems* as a result of significant manual language engineering effort (Nitta, 1986; Ryan, 1989).

The first statistical MT model, introduced by Brown et al. (1990), paved the way for replacing manual rule-based effort with *statistical methods*. This word-based approach was later extended by Koehn et al. (2003) to form the *phrase-based* MT, that in the following decade became widely used and it allowed for the rapid development of relatively satisfactory MT systems in hundreds of language pairs (e.g. Koehn et al., 2009). Several variations of the phrase-based approach or other statistical approaches were developed, such as the tree-based models (Zhang et al., 2007), the hierarchical models (Chiang, 2005) and the factored models (Koehn and Hoang, 2007). The development of Neural Machine Translation (Bahdanau et al., 2014) has been the latest step in the scale of progress, claiming that the MT quality can be brought very close to the quality of human translations (Wu et al., 2016).[1]

During the last two decades, MT became accessible to the everyday user. In 1997, the first online translator started offering free MT services (Ament, 1998). The usage of such end-user-oriented systems expanded during the 21st century, as they were added into browsers and were made available through smart-phones. Yet not of perfect quality, these public systems are capable of providing instant translations to a large audience for content gisting and quick exchange of information.

The industry has also recently turned into adopting MT into their business scenarios. Translation agencies nowadays often recommend the usage of MT for particular translation areas to the professional translators (Harris et al., 2016), whereas MT systems have been successfully employed for large-scale e-commerce platforms (Russell and Gillespie, 2016).

1.1.2 Evaluation of Machine Translation

The evaluation of MT refers to the process of analysing the translations produced by a MT system and expressing a judgment about their correctness in a meaningful and possibly measurable way. MT systems are not perfect (still/yet) and the produced translations may be incorrect or contain errors. Therefore, evaluation is a necessary step to any further decisions concerning either the translations per se, or the MT system that produced them. MT evaluation is a necessary tool for MT research and development, since it can allow for measurable comparisons between various systems, methods, data, or other characteristics.

Among the methods for conducting the evaluation, *manual MT evaluation* refers to the process when humans provide a qualitative judgment on the output, a relatively trustworthy but rather time-consuming and costly operation. On the contrary, *automatic MT evaluation* refers to programs which can quickly provide a numerical value that represents some qualitative characteristics of the output. The latter, although not always accurate, is commonly used during the development process, due to the low cost and the possibility for infinite repetitions.

The most common metrics for automatic evaluation compare the MT output with one or more *reference translations*, produced by humans and therefore assumed to be ground truth. The most common metrics include Word Error Rate (WER; Nießen et al., 2000), Bilingual Evaluation Understudy (BLEU; Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and Translation Error Rate (TER; Snover et al., 2006).

[1] Since the first results of Neural MT appeared after the experimental phase of this dissertation was concluded, issues of Quality Estimation (QE) related particularly to Neural MT are not explicitly handled.

1.1.3 Quality Estimation

The ongoing integration of MT in professional workflows has increased the need for evaluating the quality of the produced output. However, the required evaluation most often differs from the classical one introduced above. Whereas automatic reference-based metrics can be useful in situations when a reference translation is available, e.g. for assessing the quality of a produced system during the development phase, in many applications the quality of the automatic translations needs to be assessed in cases when the correct translation is unknown. Similarly, even though manual evaluation could be in principle useful in this case, its time and budget requirements make it inapplicable for most real application scenarios. What is needed is in fact some quality prediction, or, as it is commonly called, *Quality Estimation* (QE; Specia et al., 2010a).

In other words, Quality Estimation can be described as the automatic process of analysing the translations produced by a MT system and predicting a judgment about their correctness in a measurable way without access to the reference translation. The result of QE is often one or more numerical values that express some notion of quality for the given translation(s). Depending on the task, a predicted judgment may refer to a QE on the *document level*, i.e. when an entire piece of text needs to be evaluated at once (Soricut and Echihabi, 2010; Scarton and Specia, 2014a), the *sentence level*, when a single sentence is analyzed and judged (Specia et al., 2009a), the *phrase level* (e.g. Logacheva et al., 2016a) or the *word-level* (Ueffing and Ney, 2005; Logacheva et al., 2015).

1.1.4 Types of Quality Estimation

The way humans perceive quality is considered subjective per se (Garvin, 1984). One of the possible definitions of the quality, concerning translation, states that *a quality translation demonstrates accuracy and fluency required for the audience and purpose and complies with all other specifications negotiated between the requester and provider, taking into account end-user needs* (Koby et al., 2014). Therefore, in order to increase the objectivity of possible judgments of the quality, one can define a QE task by specifying a particular use-case. This way, several different types of tasks have been defined, referring to the estimation of *perceived adequacy* or *meaning preservation*, *post-editing time*, *post-editing rate* and *post-editing effort*. A complex automatic metric (such as BLEU or METEOR) may also be the goal of the prediction. Numerically, the task may be aiming at a binary value (accept/reject), a discrete set of labels, a real number or a relative ranking.

Confidence Estimation (CE) refers to the process of providing a translation judgment based on information of how the the translation process was performed (Section 1.2.5). In its most common case, it is the MT translation system itself that provides a value referring to the confidence of its produced translation (Specia et al., 2010b). We see CE as a subset of of QE, as the latter can make use of a broader range of quality indicators, which may or may not include system-internal information.

1.1.5 Comparative Quality Estimation

Whereas most QE tasks aim at predicting a judgment given one single translation, there have been concerns on how confident one can be in quantifying quality. Humans themselves seem to have difficulties in scoring the quality of single translations, particularly in defining the distinction between the level of quality that each score represents (Callison-Burch et al., 2007). A possible solution is to reduce the requirements for the ground truth by favouring ordinality against cardinality. This can be done by eliciting judgments of relative quality, through the direct comparison between two or more translation items (Duh, 2008). For problems that require the identification of improvements or comparative performance, it may be beneficial to remove qualitative observations that are irrelevant to the comparison and interfere to the decision without a reason.

Following this idea, we will resort to *Comparative Quality Estimation* as *the automatic process of analysing two or more translations produced by a MT system and expressing a judgment about their comparison*. As opposed to Comparative QE, we will thereof refer to the other types of QE as *Absolute Quality Estimation*.

1.1.6 Operation of Quality Estimation

QE can be often seen as a post-processing mechanism to MT decoding and it relies majorly on Machine Learning (ML). In the most common case, QE requires the derivation of automatic quality indicators from the text or the process of the translation (Section 2.3). These quality indicators form a vector, which stands as the independent variable in our problem. A value representing a quality judgment forms the dependent variable. Consequently, a statistical model that associates the dependent with the independent variable is estimated by applying ML over existing data including translations annotated with quality judgments (Section 2.5).

This work focuses on a specific scenario: we need an automatic way to rank several machine translation outputs for one given source sentence, according to their comparative quality. This kind of ranking, performed by human annotators, has been established as a practice for evaluating MT output. Therefore, we attempt to perform *machine ranking*, by employing ML approaches in order to imitate the human behaviour. This is done through a statistical classifier, which is trained given existing human ranks and several qualitative criteria on the text. Based on the definitions above, we will be demonstrating methods and experiments of *Comparative Quality Estimation of human-perceived meaning preservation*.

1.1.7 Applications of Quality Estimation

Quality Estimation can be an integral part of automated content-translation pipelines. In fully automated cases, where the type of the multilingual information requires instant translations produced by MT, the user may be given a predicted quality score about the confidence of the translation. In semi-automated pipelines, translations below a predicted quality threshold are not served to the users, or are passed for further translations to humans.

In the professional translation industry, MT output is often being used for many products as a first step, given to professional translators for post-editing. To avoid cases where low-quality MT outputs create a heavy correction overhead, QE is used to serve only the translations that are worth post-editing against e.g. being translated from scratch directly out of the source sentence.

Comparative Quality Estimation can serve several applications in MT also driven as input to the above mentioned examples. The possibility of making choices on a sentence level can be ideal for real-time *MT System Combination*, where output provided by various MT systems is chosen to serve particular purposes, such as meaning preservation or minimisation of post-editing time. As a more specific case, the possibility of applying Comparative QE on a black box MT system allows for *Hybrid MT* by blending output from different types of systems, e.g rule based and statistical systems, on the sentence level. Finally, the ability of distinguishing improvements between two systems can lead to applications of *MT Evaluation and Development*, i.e. by identifying strengths and weaknesses of two versions of the same system even on test sets with unknown translations.

1.2 Related work

The notion of estimating the sentence-level quality of the MT output given internal system information existed already since the first rule-based MT systems. Bernth (1999) argued that it is useful for a machine translation system to be able to provide the user with an estimate of the translation quality for each sentence and described a specific implementation and examples for a Prolog-based system of that time. Meanwhile, the possibility of the word-based statistical models to estimate the confidence of their own predictions was also known (Melamed, 1997) and it was utilized accordingly (Vogel et al., 2000a,b).

The use of external automatic qualitative analysis for judging the quality of the translation was introduced by Kaki et al. (1999) and followed up by Callison-Burch and Flournoy (2001). Later work introduced a wide variety of grammatical and statistical features (Corston-Oliver et al., 2001; Akiba et al., 2002; Gandrabur and Foster, 2003). The current work re-uses and extends some of these findings. A detailed overview of related work in this aspect is given below in Section 1.2.5.

ML was employed to learn from human annotations how to take advantage of the quality indicators (Tidhar and Küssner, 2000; Cavar et al., 2000), an approach with was later extended for various purposes with more sophisticated algorithms (Corston-Oliver et al., 2001; Yasuda et al., 2002) and feature selection (Quirk, 2004; Specia et al., 2009a). Later works also included active learning (Beck et al., 2013) and domain adaptation (de Souza et al., 2014b).

A significant part of work in sentence-level QE has focused on prediction of absolute quality judgments by considering single outputs, for predicting either binary decisions or continuous scores. Important samples of this direction are detailed in Section 1.2.1. Nevertheless, the work that is most related to our methods refers to the prediction of comparison of translation outputs or ranking of translation outputs from various systems. An overview of this work and comparison to our approaches will be given below, in Sections 1.2.2, 1.2.3 and 1.2.4.

1.2.1 Absolute Quality Estimation

Predicting a single value to represent the quality of a sentence first focused on a *binary value*, e.g. to distinguish if a given translated sentence is a machine or a human translation (Corston-Oliver et al., 2001) or to assign bad/good (Blatz et al., 2003) or accept/reject (Quirk, 2004) labels. In the case of Blatz et al. (2003), the binary labels were only produced by thresholding the continuous score of reference-based metrics, but Quirk (2004) observed that training on a small dataset of human annotations is preferable to training on larger dataset produced with the automatic reference-based metric.

Prediction of *continuous* sentence-level quality scores was first done by training a regression model on either human or automatic labels of adequacy (Specia et al., 2009a). A significant part of related work in this direction focused on predicting post-editing effort. This aimed at predicting scores of perceived post-editing effort assigned by human translators (Specia et al., 2009b), or at predicting continuous labels observed on actual post-editing, namely the time required for post-editing the sentence (Bojar et al., 2013a) and the ratio of performed post-editing operations as expressed by Human-targeted Translation Error Rate (HTER Specia and Farzindar, 2010). Similarly, predictions were also done on the word level (Bojar et al., 2013a) and the document level (Bojar et al., 2015, 2016a).

In particular, human labels of perceived post-editing effort were originally assigned in a 1-4 scale (Specia et al., 2009b, 2010a), then in a 1-5 scale (Callison-Burch et al., 2012; Soricut et al., 2012) and lately in a more restrained 1-3 scale (Bojar et al., 2014). State-of-the-art methods for predicting these labels include Referential Translation Machines (RTM) with Feature Decay Algorithms (Biçici and Way, 2014), Multi-task Gaussian Processes (MTGP; Beck et al., 2014) and alternative MT-system consensus (Scarton and Specia, 2014b). RTM and MTGP showed also state-of-the-art performance in predicting post-editing time.

Concerning prediction of sentence-level HTER scores and word-based binary labels, state-of-the-art methods use multi-level task learning, by combining QE at the sentence, word and phrase level. Kim et al. (2017) use an end-to-end stack neural model that can jointly learn the word prediction model and the QE model. Martins et al. (2017) combine a pure QE system, a feature-rich sequential neural model and the predictions of an automatic post-editing system.

1.2.2 Comparative Quality Estimation with binary and continuous approaches

Having defined Comparative QE (Section 1.1.5), we hereby refer to previous work that uses various methods in order to compare and rank alternative MT outputs. Previous work can be divided into three categories, depending on whether they compare the system outputs via: (a) continuous prediction, (b) binary classification and (c) binary classification via thresholding a continuous score.

In the first category, ML is used to assign/predict a continuous score independently for each system output and then they rank the outputs based on their individual score. Early works in this category did not use any ML, as they did the selection based on a single score (Kaki et al., 1999; Callison-Burch and Flournoy, 2001; Sumita et al., 2002). ML for this purpose was first used in a combinatorial optimisation approach, in order to weight confidence scores from different types of systems so that they are comparable to each other (Tidhar and Küssner, 2000; Cavar et al., 2000). More advanced models for predicting a continuous score were trained with Regression Trees (Akiba et al., 2002) and Support Vector Machine (SVM) Regression with a linear kernel (Nomoto, 2003) albeit with only two features. A large scale sentence-level system selection was done by Specia et al. (2010b) who predicted a continuous score given a multitude of features, trained with SVM Regression with a Radial Basis Function (RBF) kernel.

In the second category, ML is given two alternative MT outputs for each source sentence and learns to produce a binary decision on which of the two is preferred. The oldest example (Yasuda et al., 2002) used ML over human preferences in order to train a binary classifier that compares the output of an example-based and a transfer-based MT system given two glass-box features. Similarly, He et al. (2010) trained a binary classifier for sentence selection between two outputs, originating from a statistical MT system and a Translation Memory. Another approach (Sánchez-Martínez, 2011) built a classifier which decides by using only source-language information, which MT system should translate a sentence but the results show a small, non-significant improvement.

In the third category, a ML approach is used in order to accept or reject a basic system and then back-off to a supplementary system without judging it. For example, Quirk (2004) used ML with a big amount of features to select between an example-based and a rule-based system. The selection is done by predicting a continuous value for the output of the example-based system whereas a cut-off value was set, under which the output of the rule-based was selected as a back-off. Similarly, Zwarts and Dras (2008) produced a binary accept/reject decision having only seen the output of an improved system and back-off to a baseline system in case of rejection.

In contrast to the above methods, our proposed one operates under a different learning concept. It employs a machine ranking approach in order to compare an undefined number of black-box systems. This is close to the binary classification approach above, with the difference that the underlying classifier is system-agnostic and that it operates based on all possible pairwise comparisons (Section 2.5). This way, contrary to the regression approach, the model only learns a relative notion of the translation quality, by having quality indicators from all outputs under comparison.

From another aspect, whereas the the main focus of most of these works is on the system combination, we maintain the broader experimental scope around the concept of ranking (Chapter 3) and we consider system combination only as an application among others (Chapter 5).

1.2.3 Comparative Quality Estimation with Machine Ranking

Related work presented previously treated the estimation of quality as a machine learning problem of predicting a continuous score or a binary value. The prediction of a full ranking by allowing multiple comparisons, seen as a problem of *learning to rank*, appears later in the related work.

Pairwise ranking is used by Pighin and Màrquez (2011) where features are simply summed with no machine learning against preferences, whereas the evaluation is done by measuring accuracy on the pairwise level, without reconstructing full rankings.

Little after we proposed the idea of sentence-level ranking in QE (Avramidis et al., 2011; Avramidis, 2012a) and its application on sentence selection for system combination (Avramidis, 2011), another experiment on the use of machine ranking for system combination (Federmann, 2013) showed positive results for one language pair.

Later, Formiga et al. (2013b) indicated more positive results with system combination. Most notably, they concluded that machine ranking makes better predictions as compared to separate regression models. Additionally, the authors observed that models trained with human rankings obtained better system-selected scores. Their models were learned on a dataset with professional human annotations based on absolute scores of adequacy, in contrast to our models learned on human rankings. Although their results on that dataset obtained higher ranking correlations than ours, when tested on the same type of dataset (Formiga et al., 2013a), they got significantly lower correlations. This comparison is yet not fully accurate due to various different implementation choices.

Additionally, QE ranking was one of the shared tasks of the 8th Workshop on Statistical Machine Translation (Bojar et al., 2013b). Participants were invited to submit QE approaches capable of ranking up to five alternative translations for the same source sentence, produced by multiple MT systems. The works submitted follow four different ML approaches.

The first approach performs ranking by predicting individual continuous values with a similarity threshold (Biçici, 2013) for each alternative translation. A second approach, sees ranking as a multiple classification problem (Han et al., 2013). The third approach follows the methods of n-best list reranking (Hildebrand and Vogel, 2013). The last approach is similar to ours, since it performs ranking by using a binary classifier on all pairwise comparison of the ranking list. In this group of submissions, Almaghout and Specia (2013) uses the same ML method as our work, but is based on Combinatory Categorial Grammar (CCG) features. In contrast to our findings (Section 3.6), Formiga et al. (2013a) found that Random

Forest classifiers with ties perform better in their setting, which also includes a wide variety of about 90 features. Empirical comparison of our work to these methods is detailed in Section 3.4.

The two former aforementioned methods, prediction of continuous values and multiple classification, are employed the purpose of Comparative QE by Shah and Specia (2014), who also demonstrate their use for sentence selection and system combination. Here we attempt a similar experiment, with the difference that we are using the pairwise classification approach instead (Chapter 5).

1.2.4 Machine Ranking for MT Evaluation and tuning

Although our work focuses on QE, the methods of machine ranking and learning over human annotations is used in related fields, such as MT evaluation and internal components of Statistical Machine Translation (SMT) systems.

Akiba et al. (2001) introduced the use of ML over human preferences to perform reference-aware MT evaluation. Multiple edit distances to the references are combined with a decision tree classifier to produce a 4-scale quality score. Ye et al. (2007) and Duh (2008) introduced the idea of using ranking in MT evaluation by developing a ML approach to train on human rank data. They show better correlation with human assessments at sentence level for the fluency score, as compared to the previously-followed non-ranking scenario.

Another similarity between Comparative QE and MT Evaluation has to do with the use of human rankings. State-of-the-art evaluation metrics such as METEOR (Lavie and Agarwal, 2007) and BEER (Stanojevic and Sima'an, 2014) use human rankings in order to tune their internal parameters.

Hopkins and May (2011) used the pairwise approach of ranking with a classifier in order to improve Statistical MT tuning (Hopkins and May, 2011).

Machine Ranking also poses some similarity to another stage of a statistical system, the re-ranking of n-best lists. These approaches generate n-best translation hypotheses with the decoder and then combine multiple model scores to calculate the objective function value which favors one translation hypothesis over the other (Och et al., 2004). The scores are typically combined log-linearly and their weights are estimated via an optimisation algorithm.

1.2.5 Feature Engineering

The identification of quality features has been a goal since several decades and related work has tried to derive features by considering several aspects of quality. Based on how they are derived, features are often distinguished into *glass-box* (Specia et al., 2009a) and *black-box* (Blatz et al., 2004) features, depending on whether they have access to internal information of the MT systems or not, respectively. An alternative meaningful categorisation of features divides them based on whether they indicate: (a) the intrinsic difficulty of the particular source sentence, (b) the difficulty to translate the sentence in general, or (c) the difficulty for the current translation model to translate the sentence (Gandrabur and Foster, 2003)

Glass-box features and particularly system-internal confidence estimators were used in early work for estimating and comparing the translation quality of different MT systems (Tidhar and Küssner, 2000; Yasuda et al., 2002). These features include the translation probability for statistical systems, the database matching for example-based systems and the amount of coverage of the input string by the selected parse for transfer-based systems with probabilistic selection (Cavar et al., 2000). More advanced ML methods were later used in order to combine a multitude of glass-box features among others. Namely, Blatz et al. (2004) used twelve feature functions from the statistical maximum entropy model, pruning statistics from the base model's search algorithm and scores from the n-best list of the translation candidates, confirming that that n-best features are more useful than the ones from single hypotheses. While Blatz et al. (2004) concluded that features that depend on the base model are more useful than those who do not, later work (Specia et al., 2009a) observed that although glass-box features may be very informative, it is possible to represent the same information using simpler features.

The first black-box quality indicators were the probabilities by n-gram models, since a character-based n-gram model was used on MT output for Asian languages (Kaki et al., 1999) and a tri-gram token-based Language Model (LM) was used for MT into English (Callison-Burch and Flournoy, 2001). Following works considered variations of n-gram perplexity, density of function words (Corston-Oliver et al., 2001), the translation score by an external word-based SMT system (Akiba et al., 2002), the score of semantic category matching, the ratio of alignment between source and target (Sumita et al., 2002), various n-gram frequency statistics and semantic similarity based on WordNet (Blatz et al., 2004). The latter, having experimented with 91 features, reported that features including target text information are better than features without. The alignment probability from IBM Model 3 was found to be a good feature when both of the languages involved exhibit a stable word order, but not when one of them has a free word order (Nomoto, 2003).

Since QE evolved as a main topic of research and settled as a main topic of yearly shared tasks, a multitude of features kept appearing. These include the projection of semantic structures (Pighin and Màrquez, 2011), morpheme-based and POS-based IBM-1 scores (Popović, 2012a), semantic similarity (Biçici et al., 2013), style classification (Moreau and Rubino, 2013) etc. The use of Recursive Feature Elimination for the purpose of choosing features for absolute QE is also used by Biçici et al. (2013).

A special feature category that we are particularly interested, are the grammatical features. Features from the rule-based parse of the translations, such as branching properties, function word density and constituent length were used in early QE experiments to distinguish human translations from MT output (Corston-Oliver et al., 2001). Features from Context-Free Grammar (CFG) productions are first used for QE by (Gamon et al., 2005), whereas Probabilistic Context-Free Grammar (PCFG) parsing appeared first as a means of fluency evaluation for generation systems, by using the parse log-probability and the number of tree fragments (Mutton et al., 2007). Similar features, acquired by parsing MT outputs, were used as QE for accepting or rejecting a translation (Zwarts and Dras, 2008), while the ones produced out of parsing source sentences were used as a hint for the translation difficulty (Hildebrand and Vogel, 2013). Additionally, Hardmeier et al. (2012) use tree kernels over constituency and dependency parse trees of both source and MT outputs.

The grammatical features presented in this thesis (first introduced in Avramidis et al., 2011) follow closely the idea of using PCFG parsing features for QE (Zwarts and Dras, 2008), but to the best of our knowledge it is the first time that such features were used for the direct comparison of two or more translations by including them in a binarised vector.

Other grammatical or fluency features were based on CCG supertags (Almaghout and Specia, 2013) and essay-correction rules (Parton et al., 2011), whereas Kaljahi et al. (2013) focused on the impact of parser accuracy on the QE.

1.2.6 Analysis of decoding events

As explained above, the first experiments on CE make use of a small number of SMT features in order to train a supervised model for predicting the quality of the Translation (Blatz et al., 2004). Later work, defines such features as *glass-box* features (Specia et al., 2009a). Glass-box features are used in order to predict other numerical indications of translation quality, such as post-editing effort or post-editing time. Contrary to these works, in Chapter 6 on *Translation errors and decoding events*, we only predict specific error types, with the focus on understanding the contribution of the features.

Prediction of specific error types was included in the shared tasks starting from the 8th Workshop on Statistical Machine Translation (Bojar et al., 2013a, 2014). Several participants contributed systems that predict error types (Besacier and Lecouteux, 2013; Biçici and Way, 2014; de Souza et al., 2014a). In that case, prediction was done on the word level and contrary to our experiments, no glass-box features were used, therefore there was no connection of the ML with the decoding process.

The work most related to the analysis of the decoding process is the one by Guzmán and Vogel (2012). It aims to identify the contribution of these features. Similar to several previously mentioned works, a multivariate linear regression model is trained in order to predict continuous quality values of complex metrics. Although the aim of this work is similar to ours, we work in a more fine-grained way: instead of modelling continuous values, we train a binary classifier to predict and explain the contribution of the decoding features to the occurrence of specific error types.

1.2.7 Quality Estimation Software

The first collaborative work for development on this field was done in the frame of the *WS'03 Summer Workshop at the John Hopkins University on Confidence Estimation of MT* (Blatz et al., 2003), but to our knowledge no application or code has been publicly available, as a result of this effort. Additionally, relevant contributions introduced several open-source tools offering MT evaluation, such as METEOR (Banerjee and Lavie, 2005), Hjerson (Popović, 2011c) and Addicter (Berka et al., 2012), whereas a good amount of such metrics and scripts were gathered in Asiya (Giménez and Marquez, 2010). All this work is based on comparing the produced translations with reference translations, which generally falls out of the scope of QE.

Few pieces of software on QE have been released with an open source. QuEst (Specia et al., 2013), previously also known as Harvest, is the most established one, as it has been used as a baseline for the yearly Workshop on Statistical Machine Translation (WMT) Shared Tasks on QE (e.g. Callison-Burch et al., 2012). The main difference with our approach is that QuEst uses two different pieces of software for feature generation and machine learning respectively, where the former is written in Java and the latter in Python. Moreover, many of its parts operate only in batch mode. For these two reasons, in contrast to our software, operating in a real-usage scenario (e.g. server mode) with sentence-level requests is non-trivial. Its latest version, QuEst++ (Specia et al., 2015), additionally supports word-level and document-level QE. In contrast to the regression-based orientation of QuEst, our software is aimed to sentence-level ranking and selection of MT output, by implementing comparative QE. Nevertheless, since not the same quality features have been implemented in both toolkits our software includes a QuEst wrapper for conformity with WMT baselines.

Some most recent software focuses on an another level of granularity, namely word-level QE. WceLig (Servan et al., 2015) is a tool which introduces support for various target languages, handles glass-box, lexical, syntactic and semantic features for estimating confidence at word-level. Marmot (Logacheva et al., 2016b), focuses on word-level and phrase-level QE and is written in Python. It offers a modular architecture, users can easily add or implement new parsers, data representations and features that fit their particular use cases, whereas it can be easily plugged into a standard experiment workflow.

In contrast to most of the above software, the approach of the software presented here focuses on a double-usage scenario for both scientific experimentation and real-usage. Feature generators and machine learning support both batch mode and sentence-level mode, whereas the functionality can be easily plugged into web-services and other software that requires QE functionality. Furthermore, it offers a dynamic pipeline architecture, including wrappers for Natural Language Processing (NLP) tools written in several programming languages.

1.3 Scientific goals

In this work, we shall focus on Comparative Quality Estimation as another aspect of Quality Estimation. We aim at using this concept for creating a ranking mechanism, which could perform automatic quality ranking of several alternative translation outputs of the same source sentence. We are interested in approaching this by formulating it as a supervised machine learning problem.

We intend that the ranking mechanism is able to predict translation rankings similar to the way humans would do. This mechanism should also perform equally or better than other methods of automatic ranking and state-of-the-art reference-aware metrics. We shall investigate whether elaborate features and machine learning may provide more information about relative translation quality than a direct comparison with references. We aim at identifying the machine learning architecture and methods that can achieve the best performance and solve issues such as decision uncertainty.

Then, we are interested to select a well-performing set of features that provide quality indications based on the text of the source sentence and is translations. We intend to suggest new grammatical features, understand the linguistic and/or probabilistic intuition behind them and confirm their contribution to the models. Additionally, we shall test how features used in other aspects of Quality Estimation, such the ones referring to complexity, adequacy and fluency, can be used in this case.

We aim at applying the ranking mechanism to combine output from different types of systems, resulting in better performance than its components. We shall see how the overall machine learning design should be modified to serve this variation of the problem.

We aim at understanding how common MT errors are associated with the way MT systems operate. We intend to run an empirical analysis of the occurrence of such errors and try to identify possible reasons in the internal functionality of phrase-based SMT systems.

We aim at creating an open-source software that is able to automatically analyse translations, produce a multitude of features, learn Comparative QE models and evaluate them. This software should serve for the reproduction of the thesis experiments and other common experiment pipelines. Additionally, it should follow basic principles for re-usability and deployment in real-use scenarios and be extensible for other types of QE.

1.4 Publications

The contents of this thesis have been presented as publications, as listed below:

Avramidis et al. (2011) introduces the use of machine ranking of MT output via pairwise comparisons. It describes the preliminary ranking mechanism (Section 4.1) which competes with several reference-aware automatic metrics for German-English in the *Metrics Task* of the *Sixth Workshop on Statistical Machine Translation* (Callison-Burch et al., 2011).

Avramidis (2011) uses the idea of pairwise comparisons for system combination. It demonstrates a pilot selection mechanism which is able to rank system outputs from 4 different types of MT systems using glass-box features on the sentence level.

Avramidis (2012a) presents the positive results of a primary version of the basic machine ranking mechanism (Section 4.2), including the method of tie elimination via weighting decisions with their classification probabilities (Sections 2.4.2.2 and 3.7). The best configuration achieves acceptable correlation with human judgments for German-English, which is higher than that of state-of-the-art reference-aware automatic MT evaluation metrics such as METEOR and Levenshtein distance. The method is further detailed in Avramidis (2013b).

Avramidis (2012b) indicates improvements on QE by using several feature scoring and feature selection methods (Section 4.1.1) and also includes some first ways of converting decoding statistics into glass-box features (Section 6).

Avramidis and Popović (2013) presents the full basic ranking mechanism (Section 4.2) as the winning submission of the *Quality Estimation Ranking Task* at the *Eighth Workshop on Statistical Machine Translation* (Bojar et al., 2013a).

Avramidis et al. (2014a) presents the development of the corpus used in the Chapter 6. The corpus is a result of a detailed large scale human evaluation consisting of three tightly connected tasks: ranking, error classification and post-editing.

Avramidis and Popović (2014) investigates situations in the decoding process of phrase-based SMT and how they correlate with particular post-editing corrections on the output of the translation that, as deemed necessary by humans post-editors (Chapter 6).

In Avramidis et al. (2015a,b, 2016a,b) we present several variations of applying the ranking mechanism for a hybrid system combination (Chapter 5).

Avramidis (2017a) presents the positive results of the advanced machine ranking mechanism (Section 4.3), including the use of Gradient Boosting, Recursive Feature Elimination and further grammatical features.

In Avramidis (2013a) we present an open source implementation of the evaluation methods suitable for machine ranking, whereas in Avramidis et al. (2014b), (Avramidis, 2016a,b) the development of the open-source QE software Qualitative is detailed (Chapter 7).

Relevant contributions

On the side of the experiments presented in this thesis, a multitude of relevant tasks were accomplished which are presented via the listed publications:

In Avramidis et al. (2012) we contribute to a multilingual parallel corpus automatically annotated with MT glass-box features, as part of the organisation of the *ML4HMT Workshop on Optimising the Division of Labour in Hybrid Machine Translation* (Federmann et al., 2012).

In Burchardt et al. (2013) we contribute by reporting the empirical observation that post-editing the MT output is significantly faster than translating the same sentences from scratch (Bojar et al., 2013a).

In Popović et al. (2013) we contribute to the investigation of the selection process with the purpose of post-editing.

In Avramidis (2014) we show that while building a QE model for the metric Human-targeted Translation Error Rate (HTER), it is easier to learn the whole complex metric, than its components separately.

In Lommel et al. (2014) we contribute to the specification of a new analytic measure for the annotation and analysis of MT errors.

In Popović et al. (2014) we contribute to the investigation of relations between different types of post-editing operations, cognitive effort and temporal effort.

In Shah et al. (2013) we contribute with adding an implementation of probabilistic grammatical features into the state-of-the-art open-source QE software QuEst.

In Klejch et al. (2015) we contribute to the development of MT-ComparEval, a graphical evaluation interface that can assist MT development.

In Aranberri et al. (2016) we addresses the need to aid MT development cycles with a complete workflow of MT evaluation methods.

In Srivastava et al. (2016) we contribute to the investigation of techniques to enrich SMT with automatic deep linguistic tools and evaluate them with a deeper manual linguistic analysis, using English–German IT-domain translation as a case-study.

In Avramidis (2017b) we present a tool offering a graphical user interface for QE by connecting the back-end of the QE decision-making mechanism with a web-based application.

In Avramidis (2017c), we suggest that there can be significant improvements for absolute QE, when predicting HTER as a multi-component metric using a multi-layer perceptron.

In Macketanz et al. (2017) we contribute to a linguistically driven evaluation method and apply it to the main approaches of MT (Rule-based, Phrase- based, Neural).

In Avramidis et al. (2018) we present an alternative method of evaluating QE systems based on a linguistically-motivated Test Suite.

1.5 Dissertation structure

After a short review of previous related work, the list of the scientific goals and the relevant published work (Chapter 1), we define the problem and describe the methods (Chapter 2), including the underlying pairwise mechanism, the machine learning algorithms, the features used and the evaluation.

Chapter 3 includes the description and the results of the experiments concerning the usage of Comparative Quality Estimation for ranking purposes, including the measurement of the correlation with human rankings and the comparisons with basic unintelligent rankings, with other ranking methods and with reference-based evaluation metrics. Then, we give the results of the experimentation with several machine learning methods in the context of automatic ranking. We showcase the positive results achieved by the weighting of the pairwise decisions on our best models but also for various learning algorithms.

Chapter 4 includes details about the feature engineering process. Three feature engineering phases are detailed, including the results of choices over the participating features, the feature selection methods and assumptions for the role of each feature.

Chapter 5 includes the experiments on system combination by using the automatic selection mechanism. The performance of the selection mechanism is measured against its components and the best choice of learning method and feature sets is presented.

Chapter 6 includes a statistical analysis of MT errors corrected by human post-editors and statistical results on the decoding steps that correlate with them.

Chapter 7 introduces an open-source software for Comparative QE, capable of parallel feature generation and including several machine learning features and the possibility of experimental parameter explorations. Conclusions and further work are outlined in Chapter 8.

Chapter 2

Methods

The thesis consists of a set of empirical experiments. Before providing details about the execution and the results of these experiments, it is necessary to present the methodological background that the experiments are based on. This Chapter therefore aims at providing a detailed and concise overview of all methods used, including the necessary references to the theory behind them. In particular, formal definitions for the ranking and the ranking mechanism are provided in Section 2.1, whereas the structure of the latter is outlined in Section 2.2. In Section 2.3, the generation of features is detailed along with the linguistic or empirical intuition behind them. The learning techniques of Machine Ranking and the underlying ML methods are described in Sections 2.4 and 2.5 respectively, followed by the Feature Selection methods (Section 2.6). Finally, the measures used for the empirical evaluation are explained in Section 2.7. The Chapter is concluded with a note on tools used for the implementation (Section 2.8) and a summary (Section 2.9).

2.1 Definitions

In this section, the concepts of Comparative QE are explained and defined formally. This includes the concepts of the *rank*, the *ranking process*, the *ties*, the *ranker*, the *training set*, the *learning process*, the *ranking model* and the *ranking mechanism*.

2.1.1 Ranking

This work aims at developing an empirical system which is able to order multiple translation outputs in the same way humans would do. In particular, the system is given one source sentence and several translations which have been produced for this sentence. The goal is to *rank* them, i.e. to order the translations based on their quality after considering several qualitative criteria over the translations.

In this *ranking* process, each translation is assigned a real number (further called a *rank*), which indicates a quality judgment relative to the competing translations for the same source sentence. E.g. given one source sentence and its m translations, each translation would get a rank in the range $[1, m]$. The rank value can be explicitly or implicitly given by humans or derived by an automatic evaluation score. Typically, a lower rank means a better translation output.

Formally, we define a ranking

$$R = \{s, \mathbf{t}, \mathbf{r}\} \tag{2.1}$$

where a source sentence s is associated with a set of translations **t**:

$$\mathbf{t} = (t_1, t_2, \ldots, t_m) \tag{2.2}$$

and t_j is the j-th translation of s and m the number of the translations. Furthermore, each set of translations **t** is associated with a list of judgments (ranks) **r**:

$$\mathbf{r} = (r_1, r_2, \ldots, r_n) \tag{2.3}$$

where r_j is the judgment on translation t_j, as compared to the other translations in **t**.

This kind of qualitative ordering does not imply any absolute or generic measure of quality. Ranking takes place on a sentence level, which means that the inherent mechanism focuses on only one sentence at a time, considers the available translation options and makes a decision. Any assigned rank has therefore a meaning only for the sentence-in-focus and given the particular alternative translation candidates. For example, a translation output that gets a rank 2 is worse than the translation output which has got a rank 1 and better than the translation outputs 3, 4 etc. Accordingly, no quality indication can be assumed between the aforementioned translation with rank 2 and the translation of another sentence with the same rank.

2.1.2 Ties

In each ranking, the same rank r_i may be assigned to two or more translations of the same source, if no preference can be expressed among them. Such a case defines a *tie* between the relevant translation candidates.

$$\text{tie}(t_i, t_j) : \text{if}(r_i = r_j) \tag{2.4}$$

Ties are common phenomenon when judges assign subjective judgments to translation outputs. A tie occurs when there is a genuine indistinguishability on the quality of the compared translations, or when the judges fail to distinguish any quality difference. In our setup we consider ties an unreliable piece of information, when they originate from humans. Therefore, pairwise ties are removed from the testing gold labels. [1]

[1] Later, we will try to reduce or even penalise the prediction of ties when predicted by the system. The handling of ties is explained in Section 2.4.1 concerning the handling of ties during the training process, in Section 3.1.3 concerning the existence of ties by humans within the test set and in Section 2.7.1.1 concerning the evaluation of ties predicted by the system. Methods to reduce the prediction of ties are detailed in Section 2.4.2.2.

2.1.3 Ranking mechanism

Extending equation 2.1, we assume a set of n source sentences S:

$$S = s^{(1)}, s^{(2)}, \dots s^{(n)} \tag{2.5}$$

Similar to equation 2.2, each source sentence $s^{(i)}$ is associated with a set of translations:

$$\mathbf{t}^{(i)} = (t_1^{(i)}, t_2^{(i)}, \dots, t_m^{(i)}) \tag{2.6}$$

where $t_j^{(i)}$ is the j-th translation of the i-th source sentence and m the number of the translations. Following equation 2.3, each list of translations is associated with a list containing relative judgments (ranks)

$$\mathbf{r}^{(i)} = (r_1^{(i)}, r_2^{(i)}, \dots, r_n^{(i)}) \tag{2.7}$$

where $r_j^{(i)}$ is the judgment on the j-th translation of the i-th source sentence.

A *feature vector* is defined as $\mathbf{x}$ as:

$$\mathbf{x}^{(i)} = \Gamma(s^{(i)}, \mathbf{t}^{(i)}) \tag{2.8}$$

and it is created from every pair of source and its translations $(s^{(i)}, \mathbf{t}^{(i)})$, where $i = 1, 2, \dots n$. The function Γ that produces the feature vector given a source and its translations is referred to as *feature generation* (further analyzed in Section 2.3).

Each feature vector $\mathbf{x}^{(i)}$ derived from the i-th source sentence (see eq. 2.8), and the corresponding list of ranks define an *instance* I:

$$I = (\mathbf{x}^{(i)}, \mathbf{r}^{(i)}) \tag{2.9}$$

A *training set* of n instances is consequently defined as T:

$$T = \{(\mathbf{x}^{(i)}, \mathbf{r}^{(i)})\}_{i=1}^{n} \tag{2.10}$$

A *ranker* is a function ρ which given a feature vector $\mathbf{x}^{(i)}$ produces a list of *predicted* ranks $\widehat{\mathbf{r}}^{(i)}$. The goal of the *learning process* (or *training* process) is therefore to define a function ρ that minimizes the total error $\mathcal{E}$ between the predicted list of ranks and the golden list of ranks, as seen in the training data:

$$\sum_{i=1}^{m} \mathcal{E}(\mathbf{r}^{(i)}, \widehat{\mathbf{r}}^{(i)}) \tag{2.11}$$

Every differentiated instance of ρ will be thereafter referred to as a *model*. The learning process is further analyzed in Section 2.5.

Following the definitions above, the *ranking mechanism* is the function aggregating the functionality of both the feature generation function Γ and the ranker ρ; i.e. given a source sentence s and its translations $\mathbf{t}$, the ranking mechanism can predict the corresponding ranks $\widehat{\mathbf{r}}$.

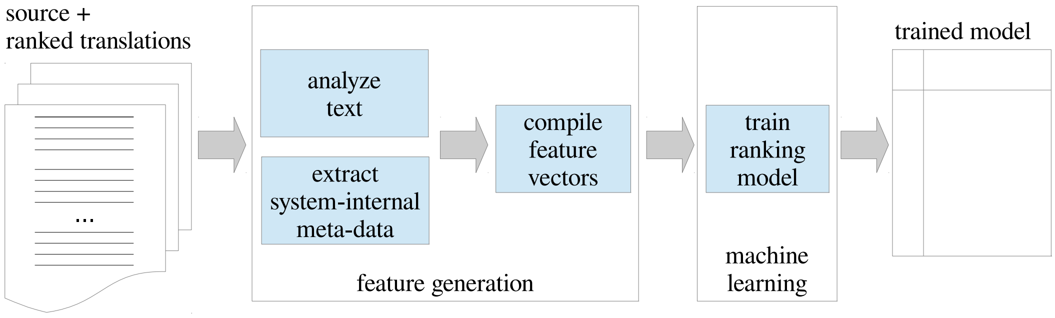

Figure 2.1: *The training process of the ranking mechanism*

2.2 Overview of the ranking mechanism

As defined above, the creation of a Quality Estimation mechanism requires the training process, which receives a training set and results into a trained Quality Estimation model. The sequence of the stages of the training process are depicted in Figure 2.1:

- The training process can take place given a *training set*. This is a data set containing a big amount of source sentences, where for each source sentence there are several alternative translations. Each set of alternative translations is associated with a list of judgments, which depending on the problem, can be provided either by human annotators or derived via automatic scores based on the reference translation(s).

- *Feature generation* (Section 2.3) produces numerical features that can serve as indications for the quality of the translations. These features may be derived directly from automatic analysis on the text (Section 2.3.1), and from various parts of the translation process (Section 2.3.2). Additionally, *feature selection* (Section 2.6) may be required in order to reduce the amount of features. The final result of the feature generation process is a numerical feature vector.

- The *machine learning* stage employs statistical methods in order to produce a model that associates the numerical features of each translated sentence with the respective ranking labels. Several machine learning algorithms employed for this purpose are explained in Section 2.5.

After the training process has finished, the produced model can be stored. In a later time, the same trained model can be applied to predict the ranking of previously unseen instances that have undergone the same feature generation process (Figure 2.2).

In order to *evaluate* the trained model, one can apply it on a set of instances, whose assumingly correct ranks (referred to also as *golden ranks*) are known but hidden from the ranking mechanism at the application stage. This test set has been also excluded from the training

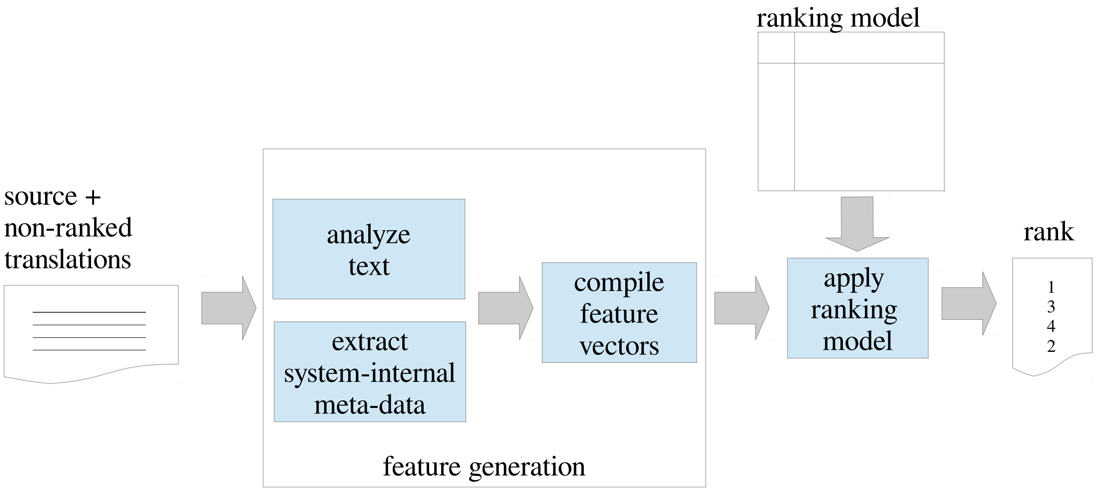

Figure 2.2: *The application of the trained ranking mechanism on previously unseen instances*

process. Finally, the evaluation process is based on comparing the predicted ranks with the golden ones and calculating the level of the prediction success or the respective error. The evaluation methods are detailed in Section 2.7.

2.3 Features

The features are numerical values that represent characteristics related to the translation. These numerical values are processed by a machine learning algorithm, so as to be statistically associated with labels about the quality of the translation, creating a QE model. The contribution of the features into the QE model has been explained earlier, in equations 2.8, 2.9 and 2.10 (Section 2.1.3).

2.3.1 Black-box features

Similar to the previous work on QE, the source sentence and the corresponding translations are analyzed by several linguistic tools in order to provide a set of features indicative of the translation quality. The features used fall into the following categories:

2.3.1.1 Grammatical features

One of the common issues that affect MT quality and acceptability is the grammaticality of the generated sentences. Such issues occur often in statistical systems (particularly the ones following the phrase-based approach) since they treat the generation process in a rather shallow way by using LMs. As an additional measure of quality, which can capture more complex phenomena (such as grammatical fluency, long distance structures, etc.), we include features derived by parsing the generated translations with a Probabilistic Context-Free Grammar (PCFG; Petrov et al., 2006). These features extend previous work, which has indicated that parsing the text of automatically-generated outputs can provide good measures of fluency (Mutton et al., 2007; Zwarts and Dras, 2008; Wong and Dras, 2010).

We apply coarse-to-fine n-best PCFG parsing (Petrov and Klein, 2007) which operates by creating many possible tree parses for a given sentence, forming an n-best list of parse hypotheses. These hypotheses are scored in a probabilistic way, leading to the selection of the tree with the highest overall probability.

Numerical parse features We allow an n-best list with a size of n=1000 and count the number of trees generated. Although for a majority of the sentences the n-best list reaches the limit, some sentences are parsed to a smaller number of trees, which signifies fewer possible tree derivations, i.e. less parsing ambiguity, a feature which would be useful for our purpose. Additionally, we extract the basic parsing statistics. These include:

- the sentence log-likelihood, i.e. summing out all parse trees in the n-best list: $P(w)$,
- the joint log-likelihood of the best tree in the n-best list and its words: $P(t, w)$,
- average log-likelihood of all parse trees in the n-best list
- the full height of the best parse tree

The values of these parsing statistics reflect the parsability of the sentence. Lack of parsability would indicate possible grammatical or fluency issues (Wagner and Foster, 2009).

Tree label counts rely on the assumption of isomorphism, i.e. the fact that the same or similar grammatical structures should occur on both source sentence and translation(s). Furthermore, one would assume that the grammatical structure of the competing translations should be similar. Therefore, we count the basic node labels of the parse tree, namely the noun phrase (NP), verb phrase (VP), prepositional phrase (PP), verb (VB), noun (NN), determiner (DT), sentence (S), subordinate clause and punctuation occurrences. Indicatively, such tree label counts should be able to capture e.g. the failure of a system to translate a VP.

Additionally, for every node label, we calculate its depth and height in the tree and the number of leaves this node is projected to. If a label appears more than once, we add the average and the maximum of the respective feature values.

To the best of our knowledge, these features do not appear in previous research of QE, except for some similar features for confidence estimation on the parsing constituents of rule-based systems (Corston-Oliver et al., 2001).

Context-Free Grammar rules For every sentence, its parse tree is decomposed into a list of Context-Free Grammar (CFG) rules. This expands the use of CFG rules as done by Gamon et al. (2005), with the addition that for every rule, we calculate its depth and height in the tree and the number of leaves this rule is projected to.

The vertical position of some CFG rules in a parse tree may be a hint of how successful the translation of this grammatical substructure was, particularly useful when compared to

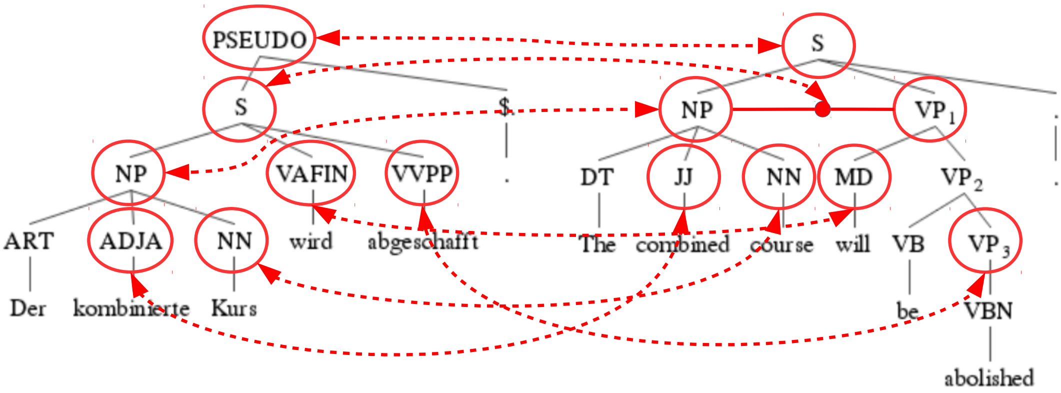

Figure 2.3: *Indicating alignment between source/target tree nodes when translating from German to English. The produced features can be seen in Table 2.1*

alignment			count	depth
PSEUDO	↔	S	1	1
S	↔	NP VP	1	2
NP	↔	NP	1	3
ADJA	↔	JJ	1	4
NN	↔	NN	1	4
VAFIN	↔	MD	1	4
VVVP	↔	VP	1	3
$.	↔	.	1	2

Table 2.1: *Parse node alignment features from the example in Figure 2.3*

competitive translations. Additionally, the fluency of the sentence may be indicated by the occurrence of some important rules. For instance, the existence of the CFG rule S → NP VP may indicate that the parser succeeded in constructing a proper sentence.

For the rules that contain a VP or a verb, two additional features indicate their distance from the beginning and the end of the sentences. This is of particular interest for translations into German, where the position of the VPs in the sentence is important.

Alignment of tree nodes and CFG rules The nodes and the CFG rules between the source and the target PCFG tree are aligned based on the scores of the IBM-1 model. This is similar to the alignment of dependencies performed by Pighin et al. (2012) but applied on PCFG parses instead. For every source node, several target nodes may be aligned. For every alignment between nodes, we get:

- the count of occurrences of the particular aligned nodes and CFG rules in the sentences
- the depth of the source node in the source tree
- the distance of the aligned nodes from the beginning and the end of the source sentence

sentence	correction
Right after **hearing** about it, he described it as a "challenge"	*disambiguate -ing*
An fully comprehensive insurance with tax exemption	*"an"+consonant*
Tired and disappointed are **the the** fishermen	*repeated word*
A strategy Republican **hinder** the re-election of Obama	*verb agreement*
These measures undermine the democratic system."_	*unpaired quotes*
These measures are clearly **limit** the Hispanic vote.	*"be" agreement*
Therefore I recommend taking the test.	*conjunctive+comma*
Most of the tariffs incl._basis and will not be changed._	*punctuation spacing*
similar steps are also preparing for France.	*lowercased start*

Table 2.2: *Sample suggestions generated by rule-based language checking tools, observed in the development data*

Additionally, the number of unaligned source nodes is added as a feature. An example of features tree node alignment can be seen in Table 2.1 as derived from the trees shown in Figure 2.3.

2.3.1.2 Language checking

Automatic rule-based *language quality checking*, similar to the one integrated on word processors, is applied on source and target sentences. This approach extends the idea of Parton et al. (2011), who evaluated MT output fluency with an essay correction tool. Since the essay correction method is not openly available, we use here a similar error correction approach (Naber, 2003), which provides a wide range of quality suggestions concerning *style*, *grammar* and *terminology* and the corresponding quality scores.

We use the number of the occurrences of each error rule as a feature. Since the individual occurrences of particular rules are rather sparse, we also use the length of the problematic chunks and we sum the number of the suggestions per category and in total. Examples from common suggestions acquired from language quality checking for English can be seen in Table 2.2.

2.3.1.3 Language Model

Language Models (LMs, used in related work by Kaki et al., 1999; Callison-Burch and Flournoy, 2001; Blatz et al., 2004) provide statistics on how likely the sequences of the words are for a particular language, so they are also an indication of fluency. Although Statistical MT systems are already optimised over one or more LMs, the highest possible LM score for a particular translation may be significantly low. Furthermore, using LMs may be beneficial for other types of systems, such as Rule-based Machine Translation (RBMT) systems. From the category of LM features we use the smoothed n-gram *probability* and the *perplexity* of the sentence.

2.3.1.4 IBM-1 model scores

IBM-1 model scores have been shown to be good indicators of translation quality for QE (Blatz et al., 2004; Popović, 2011a). They function similar to probabilistic lexicons, indicating the probability that a single word in the source language is translated by a single word in the target language. Given their prior positive performance, they have been adopted as features. In order to produce the features, all translations are scored based on the same external IBM-1 model and the *overall score* of the sentence is computed. The scores are calculated on both directions. Additionally (following Specia, 2010; Callison-Burch et al., 2012) we count the *average number of translations per source word*, as given by IBM-1 model, thresholded so that prob$(t|s) >$ threshold. The usual thresholds are 0.2 and 0.01.

2.3.1.5 Contrastive evaluation scores

Each translation is scored with an automatic evaluation metric (e.g. Papineni et al., 2002), using the competitive translations as (multiple) references. Whereas each competitive translation may convey several errors, there are good possibilities that they collectively are closer to, or contain parts of a correct translation. As most automatic evaluation metrics are based on n-gram comparisons, this feature may essentially indicate how much the sentence in focus “stands out” from a majority formed by its competitors, an indication which can be useful for the comparison.

This approach has shown to perform well as a feature in QE tasks predicting an absolute quality score (Soricut et al., 2012) and is expected to be even more useful for our task, since comparison is its main goal. Contrastive evaluation scores include sentence-level smoothed BLEU (Papineni et al., 2002) and METEOR (Lavie and Agarwal, 2007).

2.3.1.6 Count-based features

These are features based on simple superficial counts on the surface of the sentences. They originate from previous work on absolute QE (Specia et al., 2009a; Callison-Burch et al., 2012) and they include the count of tokens (sentence length or len), the average count of characters per token, the average number of occurrences of a word in the sentence (e.g. indicating repetitions), the ratio of tokens count in source and target, the count of commas, dots, numbers, and the ratio of tokens in the target which contain one or more letters other than the ones of the latin alphabet (a-z). Additionally, we identify the unknown words (unk), based on a big monolingual corpus and add their number as a feature too.

We augment these features with features related to the position of unknown words in the sentence. We add as features the absolute and the relative position of the first and the last unknown word, as well as the average and the standard deviation of all positions of all unknown words in the sentence.

2.3.2 Glass-box features

The glass-box features are extracted from the decoding process of a phrase-based SMT system (Koehn et al., 2003) with cube-pruning (Huang and Chiang, 2007). The decoding process performs a search in various dimensions, calculating scores for many phrases and hypothesis expansions. Most scores are difficult to be interpreted as glass-box features in their initial form. The amount of scores calculated per sentence is not fixed since it depends on the search process for forming the translation hypotheses. Meanwhile, a requirement for each feature is to have only one value that is valid on a sentence level, so that it can be used in the sentence error prediction model.

For this purpose, we process the verbose output of the decoder and derive scores, counts and other statistics that can have this sentence-level interpretation. When decoding steps contain a number of scores which is not fixed for every sentence, we extract features out of their statistics, such as the mean and the standard deviation, the minimum/maximum value and their position in the sentence. An example of how some of these features are extracted is illustrated in Table 2.3. On the upper part of the table, one can see the log-probability and the future cost estimate for each one of the phrases in the sentences. On the lower part we demonstrate some statistics that are derived from the scores and the positions of the words in the upper part.

Similar practice is applied to extract the entire set of 104 glass-box features. Many of these features appear in previous work (Blatz et al., 2004; Specia et al., 2009a), but to the best of our knowledge, the introduction of features for the position, the time, the standard deviation, the minimum and maximum values is novel. The feature set includes:

Phrase counts and positions The produced translation consists of sets of phrases that are chosen as the most probable hypothesis. On this hypothesis we count the number of phrases, words, the length of the phrases, the length difference between source and aligned target phrases and also the position of the shortest and the longest phrase in the sentence.

Unknown tokens are words or phrases whose translation was not found during the decoding process. These features are similar to the ones that appear as black-box features above, with the difference that here they are detected based on the words that are not found in the inherent phrase table. Their count, their ratio and their position in the translated sentence (average position, standard deviation of their positions, position of first and last unknown word) are included as features.

Translation probabilities *Log probability* (pC) and *future cost estimate* (c) are available for each phrase of the chosen hypothesis. We extract their average, standard deviation and also their minimum and maximum values and their position in the sentence. Additionally, we count the number of the phrases whose pC or c is too low or too high. This is done by checking whether their values are out of the standard deviation of all phrases in the sentence.

Time The decoder reports the time required for the entire translation process, the search, the Language Model calculation, the generation of hypotheses other than the ones chosen

source: [überraschenderweise] [zeigte sich] [, dass die neuen] [Räte] [in Bezug auf] [diese neuen] [Begriffe] [etwas] [im Dunkeln tappen .]

translation: [surprisingly ,] [showed] [that the new] [councils] [in relation to] [these new] [concepts] [slightly] [in the dark .]

position	phrase	pC	c
[0..0]	surprisingly ,	−0·770335	−2·69341
[1..2]	showed	−1·54184	−2·81277
[3..6]	that the new	−0·563381	−2·65923
[7..7]	councils	−0·386571	−1·98291
[8..11]	in relation to	−1·29663	−2·85591
[12..13]	these new	−0·332607	−2·17422
[14..14]	concepts	−0·540415	−2·01213
[15..15]	slightly	−0·585549	−2·00382
[16..19]	in the dark .	−1·48327	−3·90992
minimum		−1·54184	−3·90992
maximum		−0·332607	−1·98291
average (avg)		−0·83334	−2·56715
standard deviation (std)		0·448	0·5862
no of phrases with score lower than avg-std		3	1
no of phrases with score higher than avg+std		1	0
averaged position of phrase with lowest score		0·11111	0·88889
averaged position of phrase with highest score		0·55556	0·33333

Table 2.3: *Example glass-box feature extraction from the decoding result. Decoding scores such as phrase* log probability *(pC) and* future cost estimate *(c), whose number is not the same for every sentence (upper part of the table), are reduced to a fixed feature vector based on basic statistics (shown on the lower part of the table)*

and for collecting translation options. We use these as features, also averaged over the entire translation time.

Decoding graph These features come from the entire set of alternative phrase hypotheses generated during the search. From the entire set of alternative hypotheses we derive statistics from their log probability, the future estimate (average, standard deviation, count of alternative phrase hypotheses lower and higher than the standard deviation).

2.4 Machine Ranking

Ranking MT outputs is treated as a typical machine learning problem. A ranker is *learned* from training material containing existing human rankings. The learning process results in a statistical model. This model can later reproduce the same task on unknown sentences or test data. Whereas the setup and the evaluation of the system takes place on a ranking level, for the core of the decision-making mechanism we follow the principle of decomposing full ranks in pairwise comparisons (Herbrich et al., 1999; Hüllermeier et al., 2008). Then, given one pair of translation candidates at a time, a classifier has to predict a binary decision on whether one translation candidate is better than the other. This process, illustrated in Figure 2.4, is detailed below.

2.4.1 Pairwise decomposition

In this context we train one classifier for the entire data set. Each ranking of *n* candidate translations is decomposed into $n \times (n-1)$ pairs of all possible combinations of two system outputs with replacement. Each of the resulting pairs is a training instance for the classifier and consists of a class value c and a set of features $(f_1, \ldots, f_n)$. For the pairwise comparison of two translation candidates t_i, t_j with human ranks r_i and r_j respectively, the class value is therefore set as:

$$c_{i,j} = \begin{cases} 1 & r_i < r_j \\ -1 & r_i > r_j \end{cases}$$

The approach of pairwise comparisons is chosen because it poses the ML question in a much simpler manner. Instead of treating a whole list of ranks, the classifier has to learn and provide a binary (positive or negative) answer to the simple question '*which of these two sentences is better?*". This also provides the flexibility of experimenting with many machine learning algorithms for the classification, including those which only operate on binary decisions.

As explained already (Section 2.1.2), we consider ties an unreliable piece of information, when they originate from humans and we won't try to explicitly learn them or evaluate their prediction. Therefore, pairwise ties are removed from the test-set. Regarding the training material, ties that appear on a pairwise comparison are filtered out, since they do not provide any useful information about the simple comparison explained above. This means that the pairwise comparisons of the tied outputs with the other outputs are not filtered out; only those between the two tied MT outputs are.

It would be possible to explicitly learn ties, by introducing a third class or a cascade of two classifiers, but this would increase the complexity of the learning process, so we won't consider it as part of the problem. As we will see later in this section, the ties are treated as an uncertainty of the system for either of the classes.

2.4.2 Ranking recomposition

During the application of the statistical model on test data, data processing follows the same idea: The test instances are broken down to pairs of sentences and given to the classifier for a binary decision. Consequently there is a need to recreate a ranking list out of the binary pairwise classification decisions.

2.4.2.1 Hard rank recomposition

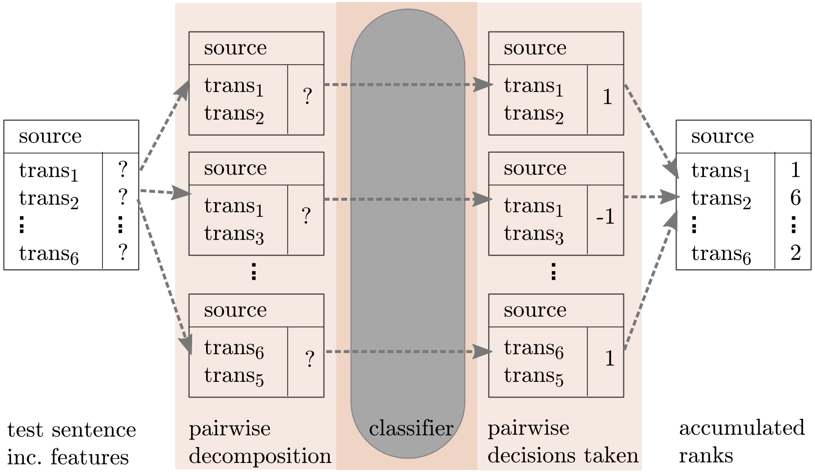

Figure 2.4: *The application of the statistical model, through the pairwise decomposition (left) and recomposition (right)*

The simplest way to go ahead with this is to sum up the decisions of the classifier. For a number of n systems, following the previous notation, the rank r_i of translation t_i would be:

$$r_i = \sum_{j \neq i}^{n} c_{i,j} \tag{2.12}$$

The translation output which has "won" the most pairwise comparisons would get first on the list and then the outputs with fewer pairwise wins would follow accordingly (Figure 2.4). We call this a *hard rank recomposition*, as only the binary decision of the classifier is taken into consideration upon summing up the predicted values.

2.4.2.2 Soft rank recomposition

One of the problems seen in previous work is that what we described here as a *hard rank recomposition* allows for the creation of ties (formally defined in Section 2.1.2). Indeed, the classifier may predict an equal number of wins for two or more translation outputs and therefore generate a tie among them. This may be intensified by the fact that the pairs have been generated in both directions, which would also result in a tie if the classifier is unable to distinguish the best out of two outputs but is forced to choose one of them. Nevertheless, while defining our problem, we considered ties as an issue of uncertainty and therefore assumed that our gold data is free of ties.

However, the probabilistic setup contains information which implies that not all classifier decisions are of "equal importance": statistical classifiers build their binary responses on a probabilistic basis. A translation output which has a number of wins with high certainty should be ranked higher than an output with an equal number of wins but with lower certainty. One can therefore use the probability of each decision to weight the sum described in Section 2.4.2.1. This is thereof referred to as *soft recomposition*. This way, the rank r_i of translation t_i would be:

$$r_i = \sum_{j \neq i}^{n} p_{i,j} c_{i,j} \tag{2.13}$$

Since the probability $p_{i,j}$ is a decimal in the range of [0,1] as opposed to a binary value, it is expected that it reduces the cases where two translation outputs end up with an equal sum.

2.5 Binary classification algorithms

In the previous section we explained that the core machine learning of the ranking mechanism operates with pairwise decisions. Consequently, in this section we revisit the most important classification algorithms that can be used for this purpose.

2.5.1 Naïve Bayes

Naïve Bayes predicts the probability of a binary class c given a set of features

$$p(c, f_1, \ldots, f_n) = p(c) \prod_{n}^{i=1} p(f_i|c) \tag{2.14}$$

The probability $p(c)$ is estimated on relative frequencies of the training pairwise examples. Since we are using continuous features f, their probabilities $p(f_i|c)$ are estimated with the locally weighted linear regression LOESS (Cleveland, 1979).

Naïve Bayes works under the assumption that the features are statistically independent, which we cannot guarantee however. It has the advantage that it offers good scalability for the training process, given large data sets.

2.5.2 k-nearest neighbour

The k-Nearest Neighnoors (kNN) algorithm classifies the test instances along with the closest training examples in the search space (Coomans and Massart, 1982). Unlike Naïve Bayes, there are no a priori assumptions about the distributions of the training data. However, a choice for the number (*k*) of the nearest neighbours is required, which is problem-specific. Here we follow the common practice of using the standard Euclidean distance as a distance metrics and setting the *k* equal to the square root of the number of training instances (Khedr, 2008).

2.5.3 Logistic regression

Logistic Regression (LogReg) is a widely-used ML method that optimizes a logistic function to predict values Y in the range between zero and one (Cameron, 1998), given a feature set

X, the beta coefficients of the features β and the intercept α (the value of the criterion when the predictor is equal to zero).

$$Y = \beta \circ X = \frac{1}{1 + e^{-1(a+\beta X)}} \tag{2.15}$$

For fitting the model the Newton-Raphson algorithm is used (Miller, 2002) in order to iteratively minimize the least-squares error given the training data. The algorithm is combined with *L2 Regularisation* (Lin et al., 2007), unless it includes *Stepwise Feature Set Selection (SFSS)* as described in Section 2.6.

2.5.4 Linear Discriminant Analysis

The classifier of Linear Discriminat Analysis (LDA) searches for a linear combination of continuous features that characterizes or separates two or more classes (McLachlan, 1992). Contrary to Logistic Regression presented above, LDA requires the assumption that the independent variables are normally distributed, that there is homoscedasticity (i.e. the class covarianves are identical) and that the covariances have full rank.

2.5.5 Support Vector Machines

The Support Vector Machines (SVMs) are discriminative classifiers defined by one or more separating hyperplanes in a high-dimensional space (Hearst et al., 1998). The samples are represented as points in the space, through a mapping that divides the samples of the different classes. The samples are optimally separated when the hyperplane has the largest distance to the nearest training-data point of any class. Linear classification is possible through the use of a linear kernel, whereas non-linear classification is done by replacing dot products with kernel functions such as Radial Basis Function (RBF). SVMs are in general effective in high dimensional spaces and operate well in cases where number of dimensions is greater than the number of samples

2.5.6 Decision trees

The induction of decision trees (Breiman et al., 1984) is a non-parametric methodology to construct tree structured rules in a supervised manner. The trees are used as a data analysis method. Given a training set of class labels and features, the training process follows the basic algorithm below (Quinlan, 1986).

If all training instances are labeled with the same class, the tree contains a leaf labeled with that class. Otherwise, a tree node is created and it is assigned a *test*, based on one feature, with mutually exclusive outcomes. Then the training set is divided into subsets, each one corresponding to one outcome and the same procedure is applied to each subset.

The decision tree, after having been constructed on the training data, can be applied on previously unseen instances. For every instance, the tree is traversed from the root to the leaves. When a tree node is processed, the test is applied on the features of the instances and the traversing continues to the corresponding branch of the tree. When the process reaches a leaf, the class label of the leaf is assigned to the instance.

The decision trees are interpretable and demonstrate logarithmic computational complexity. On the other side, decision trees are prone to overfitting, they are sensitive to small variations of data or dominant classes and it is not guaranteed that the optimal decision tree will be returned.

2.5.7 Ensemble classifiers

Ensemble classifiers are methods that attempt to combine the predictive power of several base classifiers. Through the ensembling process, it is possible to have base classifiers that generalize better and are more robust. The decision trees presented above are typically used as a base classifier, since ensembles solve most of the issues of single decision trees. There are two categories of ensemble classifiers:

- **averaging classifiers** are based on building several independent classifiers and then averaging their predictions. The idea is based on the observation that the variance of the averaged classifier reduced and therefore its performance is usually better than any individual base classifier. This category includes the **Bagging Classifier** (Breiman, 1996) the **Forest of Randomized Trees** (Breiman, 2001) and the **Extremely Randomized Trees** (Geurts et al., 2006).

- **boosting classifiers** ensemble classifiers by training them sequentially, trying to reduce the overall bias. This category includes Adaptive Boosting (AdaBoost) and Gradient Boosting.

Adaptive Boosting (AdaBoost; Freund and Schapire, 1995) combines the output of the base classifiers is into a weighted sum as the final output of the boosted classifier. At each iteration of the learning process, a base classifier is selected and assigned a coefficient so that the sum training error of the resulting boost classifier is minimized for the particular stage. At each iteration of the training process, a weight is assigned to each sample in the training set equal to the error resulting from the current iteration. These weights are used to inform the training of the base classifiers in favor of those instances misclassified by the previously learned classifiers.

Gradient boosting (Friedman, 2001) consecutively fits models of weak classifiers to provide a more accurate estimate of the response variable. These models are fitted so that they are maximally correlated with the negative gradient of the loss function which is associated with the whole ensemble. Whereas several loss functions may be used, a common loss function is the mean squared error with improvement score. There, the variable influence for the decision tree ensembles follows the influences estimated by the decision trees, calculated based on the number of times a variable is selected for splitting.

2.6 Feature selection

Feature acquisition can result in a huge number of features. Big feature sets may be unusable for training, due to the high processing needs and the sparsity or noise they may lead to. For this purpose we consider reducing the number of features with the following feature selection methods:

2.6.1 Information gain

Information gain (Hunt et al., 1966) estimates the difference between the prior entropy of the classes and the posterior entropy given the attribute values. It is useful for estimating the quality of each attribute but it works under the assumption that features are independent, so it is not suitable when strong feature inter-correlation exists. Information gain is only used for the sentence ranking task after discretisation of the feature values.

2.6.2 ReliefF

ReliefF assesses the ability of each feature to distinguish between very similar instances from different classes (Kononenko, 1994). In an iteration, it selects a random instance and defines the 2 nearest (by Euclidean distance) instances, originated from one different class each. The feature then gets a higher weight, if the nearest instance of the same class is closer than the nearest instance of the other class. It is a robust method which can deal with incomplete and noisy data (Robnik-Šikonja and Kononenko, 2003).

2.6.3 Stepwise Feature Set Selection

The Stepwise Feature Set Selection (SFSS; Hosmer, 1989) is an iterative process for choosing a feature set that optimizes the performance of a classifier, typically applied along with Logistic Regression. It consists of two repetitive phases, forward selection and backward elimination. In *forward selection*, the score χ-square statistic is computed for each feature not in the model and examines the largest of these statistics. Every feature that passes the χ-square significance test is added to the model.

Forward selection may be followed of one or more steps of *backward elimination*. For each one of the features in the model, the Wald test is computed. The least significant feature that does not meet the required significance level gets removed. The stepwise selection process ends if no further features can be added to the model.

2.6.4 Recursive Feature Elimination

Recursive Feature Elimination (Guyon et al., 2002) is a variation of backward elimination. In the beginning, the algorithm trains a classifier using all the features. Consequently, all features are ranked based on the weights or the coefficients assigned by the classifier and the lowest ranking feature is removed. In order to determine the optimal amount of retained features, all possible feature sets combinations are tested with cross-validation (RFECV). The combination that achieves the best classification performance is selected.

In our experiments we use a SVM classifier with a linear kernel as a basic estimator, where the coefficients assigned to the features are used as the ranking criterion.

2.7 Evaluation

During the evaluation phase, the trained model is applied on a set of test data and its predictions are contrasted to some gold truth. The gold truth varies and the metrics employed vary, according to the type of the problem. When the ranking mechanism is applied as a ranking method per se, or as an evaluation metric, then the correlation of the ranking against the gold labels is measured. When the ranking mechanism is applied for system combination, the produced MT output is evaluated with document-level reference-aware metrics.

2.7.1 Ranking performance

The performance of the automatic ranking is measured against human rankings. The interest lies on whether there is any relationship between the automatic ranking and the judgments of a human. If such a relationship exists, then one should be able to measure the degree of correspondence between these two rankings. For this purpose a test set is given to the ranking mechanism and the **correlation** of the rankings it produces (one per sentence) with the original human rankings is measured.

The simplest measure of ranking tau was introduced by Kendall (1938) with the purpose to analyze experiments on psychology, where the order given by different observers is compared. This measure has been analyzed and modified over the years for several purposes (Knight, 1966; Agresti, 1996; Christensen, 2005) and has been also applied to text technologies (Lapata et al., 2003; Cao et al., 2007). Since 2008 it appears modified as an official segment-level measure for the evaluation metrics in the yearly shared task for Machine Translation (Callison-Burch et al., 2008).

More related metrics emerged for use with Information Retrieval. Mean Reciprocal Rank was introduced as an official evaluation metric of TREC-8 Shared Task on Question Answering (Radev et al., 2002) and has also been applied successfully for the purpose of evaluating MT n-best lists and transliteration in the frame of the yearly Named Entities Workshop (Li et al., 2009). Additionally, Expected Reciprocal Rank (Chapelle et al., 2009) was optimised for Search Engine results and used as a measure for a state-of-the-art *Learning to Rank* challenge (Chapelle and Chang, 2011). Here, we will focus on one of the most popular family of metrics of this kind, Directed Cumulative Gain (Järvelin and Kekäläinen, 2002), which was extended to the measures of Discounted Cumulative Gain, Ideal Discounted Cumulative Gain and Normalised Discounted Cumulative Gain (NDCG; Wang et al., 2013), whereas the latter is used for our experiments.

2.7.1.1 Kendall's tau

As a basic correlation metric we use **Kendall's tau** (Kendall, 1938; Knight, 1966), which measures the correlation between two ranking lists on a segment level, by counting *concordant* or *discordant* pairwise comparisons: For every sentence, the two rankings (machine-predicted and human) are decomposed into pairwise comparisons. When the predicted comparison matches the respective one by the human annotator, we count a concordant pair, otherwise we count a discordant pair. Then, τ (tau) is computed by:

$$\tau = \frac{\text{concordant} - \text{discordant}}{\text{concordant} + \text{discordant}} \tag{2.16}$$

with values that range between minus one and one. This means that the ranking is better when the value gets closer to one.

Concordant and discordant counts from all segments (i.e. sentences) are gathered and the fraction is calculated with their sums. This is the τ calculation that appears in WMT results and has therefore been widely used in prior work.

Handling of ties The calculation follows the formula of the Workshop on Machine Translation (WMT; Callison-Burch et al., 2012), in order to be comparable with other methods:

$$\tau = \frac{\text{concordant} - \text{discordant} - \text{ties}}{\text{concordant} + \text{discordant} + \text{ties}} \tag{2.17}$$

Prior to this calculation, pairwise ties in the human-annotated test set are excluded from the calculations, as ties are considered to form uncertain samples that cannot be used for evaluation. For the remaining pairwise comparisons, where human annotation has not resulted in ties, every tie on the machine-predicted rankings is penalised by being counted as a discordant pair.

Significance test for the null case The significance test of Kendall tau correlation is based on investigating the null hypothesis that *there is no correlation between the two sets of ranks*, i.e. the rank produced by the system and the corresponding rank produced by the human annotator.

$\hat{\tau}_i$ is the τ coefficient for each item i. Under the null hypothesis of no correlation, $\hat{\tau}_i$ would be zero and therefore the individual Kendall correlations of the m rank elements would have a zero mean $\mu_{\hat{\tau}_i}$ and a variance $\sigma^2_{\hat{\tau}_i}$ as following:

$$\mu_{\hat{\tau}_i} = 0 \qquad \sigma^2_{\hat{\tau}_i} = \frac{2(2m+5)}{9m(m-1)} \tag{2.18}$$

Because there may be ranking sets with different length k, we need to consider a different variance for each ranking length, defining a different distribution. For this, *inverse-variance weighing* (Hartung et al., 2008) shall be applied on the mean, in order to calculate the significance over all the distributions.

If there are n_k cases where the items per ranking $m = k$, then the sum of n_k independent such correlations, its mean and its variance would be:

$$\text{sum}_{\hat{\tau}_i} = \sum_{i=1}^{n_k} \hat{\tau}_i^{(k)} \qquad \mu_k = 0 \qquad \sigma^2_k = n_k \frac{2(2m+5)}{9m(m-1)} \tag{2.19}$$

and respectively for their average, one would get:

$$\bar{\tau}^{(k)} = \frac{1}{n_k} \sum_{i=1}^{n_k} \hat{\tau}_i^{(k)} \qquad \mu_{\bar{\tau}^{(k)}} = 0 \qquad \sigma^2_{\bar{\tau}^{(k)}} = \frac{1}{n_k} \frac{2(2m+5)}{9m(m-1)} \tag{2.20}$$

The estimate of τ with the smallest variance would weigh inversely proportional to the variance. That is:

$$\hat{\tau} = \frac{\sum_k w_k \bar{\tau}^{(k)}}{\sum_k w_k} \qquad \text{where} \qquad w_k = n_k \frac{9m(m-1)}{2(2m+5)} \tag{2.21}$$

According to Kendall's theory, $\bar{\tau}$ will be approximately following the normal distribution under the null hypothesis. The error on the weighed mean can now be shown to be:

$$\sigma^2_{\hat{\tau}} = \frac{1}{\sum_k w_k} \tag{2.22}$$

And Z would follow the standard normal distribution:

$$Z_{\hat{\tau}} = \frac{\hat{\tau} - \mu_{\hat{\tau}}}{\sigma_{\hat{\tau}}} \tag{2.23}$$

This can be used to test the significance with a one- or two-tailed test. According to Kendall's theory, continuity correction would be required, but given a high number of rankings n, as is the case in our experiments, this is not needed.

Empirical confidence intervals for the non-null case The above analysis serves for proving that an achieved correlation is statistically significant by rejecting the null hypothesis that two rankings are not significantly different, i.e. the observed correlation does not differ from the zero correlation. Nevertheless, in addition to testing against the zero correlation, we are interested in comparing different correlations with each other. This might be the case when ranking correlations have been observed e.g. on the rankins produced by two different models and we want to see if the coefficient difference between these two correlations is significant.

The significant comparison between two such correlations can be tested empirically by calculating the empirical confidence intervals of Kendall's τ using bootstrap resampling, as is the practice in the Metrics task since the Ninth Workshop on Statistical Machine Translation (WMT14; Machacek and Bojar, 2014). The authors vary the "golden truth" by sampling from human judgments. They generate 1000 new sets and report the average of the upper and lower 2.5% empirical bound, which corresponds to the 95% confidence interval.

2.7.1.2 Cumulative Gain

This family of measures is based on the Discounted Cumulative Gain (DCG), which is a weighted sum of the degree of relevance of the ranked items. This introduces a *discount*, which refers to the fact that the rank scores are weighted by a decreasing function of the rank i of the item.

$$\mathrm{DCG}_p = \sum_{i=1}^{p} \frac{2^{\mathrm{rel}_i} - 1}{log_2(i+1)} \tag{2.24}$$

In our case, we consider that relevance of each rank (rel_i) is inversely proportional to its rank index.

The most acknowledged measure of this family is the Normalised Discounted Cumulative Gain (NDCG), which divides the DCG by the Ideal Discounted Cumulative Gain (IDCG), the maximum possible DCG until position p. Then, NDCG is defined as:

$$\mathrm{NDCG}_p = \frac{\mathrm{DCG}_p}{\mathrm{IDCG}_p} \tag{2.25}$$

2.7.1.3 Normalisation of ranking lists

Normalisation emerges as a need from the fact that in practice there are many different ways to order items within the range of the rank values. This becomes obvious if one considers ties. Since there is no standard convention for ordering ties, the same list may be represented as `[1, 2, 2, 3, 4]`, `[1, 2, 2, 4, 5]`, `[1, 3, 3, 4, 5]` or even `[1, 2.5, 2.5, 4, 5]`. The alternative representations are even more, when more ties are involved.

All representations above are equivalent, since there is no absolute meaning of quality in the values involved. Nevertheless, the rank value plays a role for the calculation of some of the metrics explained above. For this purpose, prior to evaluating one can consider several different normalisation options of such ranking lists:

- **minimize**: reserves only one rank position for all tied items of the same rank (e.g: `[1, 2, 2, 3, 4]`)
- **floor**: reserves all rank positions for all tied items of the same rank, but sets their value to the minimum tied rank position (e.g: `[1, 2, 2, 4, 5]`)
- **ceiling**: reserves all rank positions for all tied items of the same rank, but sets their value to the maximum tied rank position (e.g: `[1, 3, 3, 4, 5]`). This is the default setting, inline to many previous experiments.
- **middle**: reserves all rank positions for all tied items of the same rank, but sets their value to the middle of the tied rank positions (e.g: `[1, 2.5, 2.5, 3, 4]`)

In this work, for the purpose of evaluating the experiments, the minimize method is adopted.

2.7.2 Translation performance

The performance of machine translation output is typically measured by comparing it with the reference translations. Various methods for performing these comparisons have resulted into various metrics. In this work we use the metrics Bilingual Evaluation Understudy (BLEU; Papineni et al., 2002), METEOR (Lavie and Agarwal, 2007), Word Error Rate (WER; Och et al., 1999), Translation Error Rate (TER; Snover et al., 2006) and rgbF (Popović, 2012b).

Additionally, Hjerson (Popović, 2011c) is used to induce particular error types based on the edit distance between the machine translation and the reference.

2.8 Implementation

The Language Models (Section 2.3.1.3) are trained with the **SRILM** (Stolcke, 2002) and KenLM (Heafield, 2011) toolkits. The open source **Language Tool** (Miłkowski, 2012)[2] is used to annotate source and target sentences with language suggestions. The annotation process is parallelised with the Ruffus library (Goodstadt, 2010) and the various learning parameters are explored with ExpSuite (Rückstieß and Schmidhuber, 2011). The machine learning implementations of the Orange toolkit (Demšar et al., 2004) and the Scikit-learn (Pedregosa et al., 2011) were used. PCFG parsing features (Section 2.3.1.1) are generated on the output of the Berkeley Parser and BitPar, which are explained below.

Berkeley Parser is a state-of-the-art PCFG parser that supports unlexicalised parsing with hierarchically state-split PCFGs, supporting optimal pruning via a coarse-to-fine method

[2] Open source at http://languagetool.org

(Petrov and Klein, 2007). It has the advantage that it is accurate and fast, by using multi-threading technology. Apart from the best tree for each parse, it also provides the parsing log-likelihood and a number of k-best trees along with their parse probabilities. The English grammar has been trained on the Wall Street Journal. The German grammar, represented as a Latent Variable Grammar (Petrov and Klein, 2008), has been trained on the TIGER (Brants et al., 2004) and TueBaD/Z (Telljohann et al., 2004) tree-banks, as released by the ACL 2008 workshop on Parsing German (Kübler, 2008). The Spanish Grammar has been trained on AnCoRA (Taulé et al., 2008). The parses of Berkeley Parser have been processed with phrase alignment methods in order to map node labels between source and produced translations.

BitPar (Schmid, 2004) is a parser for highly ambiguous probabilistic context-free grammars. It makes use of bit-vector operations that allow parallelising and speeding up the the basic parsing operations (Schmid, 2006). The English grammar is based on the PENN tree-bank (Marcus et al., 1993), whereas the German grammar is also based on the TIGER tree-bank. BitPar was also included on our annotation pipeline in order to provide additional evidence and allow comparisons to the observations on the Berkeley parses. It also provides sentence-level tree likelihood and k-best lists. Unfortunately, contrary to the Berkeley Parser, the k-best lists of BitPar are of limited usability due to small differences in their relative likelihood.

2.9 Summary

In this Chapter we formulated the concepts of Comparative Quality Estimation formally, including the underlying concepts of ranking, of the relative representation of quality judgments, the ties and the ranking mechanism (Section 2.1). After introducing the idea of the ranking mechanism, we explained the details of its functioning as a typical application of supervised machine learning, consisting of the feature generation and the learning methods (Section 2.2). Then we provided a detailed description of the features that are generated and employed (Section 2.3) and the intuition behind them. Consequently, the method of ranking via pairwise decisions was suggested, including the idea of weighting pairwise decisions with the classification accuracy (Section 2.4), whereas the binary classification methods used for the core of the ranking mechanism were detailed in Section 2.5. The most important Feature Selection functions were listed in Section 2.6. The chapter ended with the introduction of a set of evaluation measures related to the problem (Section 2.7).

Chapter 3

Comparative Quality Estimation for Ranking

The goal of this Chapter is to present the ranking mechanism, verify that it correlates with human judgments, prove that its performance is significant against the baselines and compare it with other state-of-the-art methods, including methods for Quality Estimation and reference-based evaluation metrics.

The Chapter is structured as follows: Section 3.1 briefly presents the experiment settings that are followed for all of the experiments of the chapter, including a presentation of the data and the strategy for defining the evaluation set. In Section 3.2 we measure the correlation of the predicted rankings with the human judgments and in Section 3.3 we compare our ranking method with some basic baselines, such as the random or the alphabetical ranking. A comparison with several known ranking methods is given in Section 3.4 followed by a comparison with other reference-aware automatic MT metrics (Section 3.5).

3.1 Experiment design

The ranking mechanism can be functional under a wide range of settings and parameters, mostly referring to the choices of ML methods and feature sets. Since this is an introductory section with a focus on highlighting the performance and usability of the selection mechanism, only two versions of the ranking mechanism are shown, so that they fit to the purpose of the comparisons. A wider range of experimentation on various parameters is presented later (Section 3.6 and Chapter 4).

The full focus of the ranking mechanism is primarily on German-English (de-en) and secondarily on English-German (en-de). The successful settings from these language directions are applied with minimal additional engineering on another 4 language directions; French to English (fr-en), English to French (en-fr), Spanish to English (es-en) and English to Spanish (en-es). The models for each language direction are trained separately, one model per direction, based on the corresponding language-specific data, labels and features.

3.1.1 Learning method and feature set

The experiments in this first par of the chapter are shown on these two versions of the ranking mechanism:

The **basic ranking mechanism** (Avramidis and Popović, 2013) is trained with 8 shallow and grammatical black-box features using Logistic Regression. The feature set was devised for de-en, by using feature scoring methods on the broader set of all available features. The selected feature set was augmented with the baseline features of WMT12, known to perform well in other types of QE. The resulting feature set was optimised for each language direction with SFSS as part of the training process of Logistic Regression (Section 2.5.3).

Indicatively, for de-en, the basic feature set (Section 4.2.5) consists of the number of tokens, the number of commas, the number of dots, the number of the unknown tokens, the contrastive METEOR score, the number of verb phrases, the parsing log-likelihood the number of valid parse trees from the parser's k-best list.

In the **advanced ranking mechanism** the models for all language directions have been trained with Gradient Boosting over an ensemble of Decision Trees. For each of de-en and en-de, the respective feature sets have been selected via Recurcive Feature Elimination with Cross-Validation (RFECV) from an extended feature set (as explained in Section 4.3 and listed in Table 4.12). For the rest of the language pairs, the basic feature set is used.

The extended feature set for de-en contains additional grammatical features, such as features from the probability distribution of the parse trees, the alignment of the main sentence node (S→NP VP) between the source and the target and their position in the sentence, the count of particular tree spans in the target, such as NPs containing a determiner and a noun (NP→DT-NN), prepositional phrase derivations (PP→IN-NP) and derivations of a VP containing an infinitive. Finally, the position and the count of NPs and nouns is used. The extended feature set for en-de contains the count of target NPs aligned with source NPs, the distance of VP-related tree spans from the end of the German sentence and features related to the number and the position of VPs and PPs, also in the German sentence. Additionally, feature sets for both language directions include the IBM-1 model scores, Language Model probabilities, several features based on the count and the characteristics of the words, statistics for the position of the unknown words, the number of errors from the language correction (including a count of spacing errors) and the number or the position of the dots in the sentence.

3.1.2 Data

The experiment is trained on human rankings resulting from the Shared Task on MT, organized as part of the Workshops on Statistical Machine Translation (WMT2008-2014, see Callison-Burch et al., 2008, 2009, 2010, 2011, 2012; Bojar et al., 2013a, 2014). Every year, a test set of several thousands of sentences is translated by all MT systems participating in the translation task. These translations are passed to the evaluation campaign, with the goal to rank the systems based on the quality of the translations. The translations of each sentence

	de-en	en-de	es-en	en-es	fr-en	en-fr
systems	137	119	89	85	127	113
sentences	6644	7125	4499	4653	6047	6467
HITs	25385	22200	13694	12371	18099	20892
ranks	126901	110964	68454	61838	90459	103830
pairs	103048	91340	46305	91340	64474	88872

Table 3.1: *WMT corpus statistics: number of systems, sentences, HITs, ranks and pairs.*

systems	de-en	en-de	es-en	en-es	fr-en	en-fr
2008	14	11	14	12	19	11
2009	21	13	13	11	21	16
2010	26	19	15	17	25	20
2011.combo	9	5	7	5	7	3
2011.newstest	21	23	16	16	19	18
2012	16	15	12	11	15	15
2013	17	15	12	13	13	17
2014	13	18	0	0	8	13
total	137	119	89	85	127	113

Table 3.2: *WMT corpus statistics: number of systems for every yearly dataset*

are grouped randomly into batches of 5 translations, which are distributed also randomly to various annotators.

The annotators are asked to compare the machine-generated translations on a sentence level. In every step of the evaluation (also called a HIT), the annotator is presented 5 different translations for the same source, along with the reference translation. Then the annotator is asked to rank the sentences from 1 to 5, based on their quality. Ties are allowed and similarly to our assumption in Section 2.4, the ranks are no absolute scores of quality and can only be interpreted relatively among the presented system outputs. An overview of the properties of the corpus are shown in Table 3.1. Detailed statistics per yearly dataset are given in Tables 3.2, 3.3, 3.4 and 3.5.

During the annotation, the systems that produced the translations are kept anonymous from the users. The vast majority of the human annotators are participants of the shared tasks (MT researchers themselves), whereas only for the data of one year, part of the annotations are done by paid workers of Mechanical Turk[1]. Since crowd sourced annotations require additional handling (Snow et al., 2008; Callison-Burch and Chris, 2009) they are excluded from the experiments presented here. The test set, apart from MT outputs, may include reference translations in the comparisons. Given our focus on learning comparisons of MT outputs, judgements related to reference translations are also filtered out.

[1] Amazon Mechanical Turk is a marketplace for hiring temporary workforce to perform work and collect Artificial Intelligence data that require human intelligence. `http://www.mturk.com`

systems	de-en	en-de	es-en	en-es	fr-en	en-fr
2008	235	276	256	183	267	211
2009	382	335	249	217	403	213
2010	441	522	468	246	375	348
2011.combo	147	171	171	177	90	90
2011.newstest	303	399	207	330	249	297
2012	975	1163	923	1166	949	1111
2013	2610	2303	2225	2334	2313	2555
2014	1551	1956	0	0	1401	1642
total	6644	7125	4499	4653	6047	6467

Table 3.3: *WMT corpus statistics: number of sentences for every yearly dataset*

	de-en	en-de	es-en	en-es	fr-en	en-fr
2008	365	426	385	264	434	307
2009	748	745	484	378	784	390
2010	1050	1407	1140	519	837	801
2011.combo	390	441	423	600	300	300
2011.newstest	924	1308	570	1119	708	918
2012	1427	1752	1141	1475	1395	1547
2013	17911	9746	9551	8016	10943	13219
2014	2570	6375	0	0	2698	3410
total	25385	22200	13694	12371	18099	20892

Table 3.4: *WMT corpus statistics: number of HITs for every yearly dataset*

	de-en	en-de	es-en	en-es	fr-en	en-fr
2008	1810	2119	1917	1315	2154	1517
2009	3731	3700	2412	1878	3900	1938
2010	5250	7035	5700	2595	4185	4005
2011.combo	1950	2205	2115	3000	1500	900
2011.newstest	4620	6540	2850	5595	3540	4590
2012	7135	8760	5705	7375	6975	7735
2013	89555	48730	47755	40080	54715	66095
2014	12850	31875	0	0	13490	17050
total	126901	110964	68454	61838	90459	103830

Table 3.5: *WMT corpus statistics: number of ranks for every yearly dataset*

3.1.3 Evaluation set

As explained above, the sets of alternative translations are split randomly and distributed to several human annotators. Because of that, the comparison of two particular translation alternatives may have been evaluated many times by different people, resulting occasionally into contradictory judgments. We choose to not remove any contradictory overlaps upon training, since the learning algorithms, due to their probabilistic nature, may not be affected or even benefit by them.

However, concerning testing, a more robust point of reference is required: a learning method should not be penalised for making decisions on data points that humans anyway disagree. For these purpose, we merge the multiple HITs for the same source sentence into one, so that each system output appears once in the new ranking. The system outputs for this sentence are now ordered based on how many pairwise comparisons for this particular sentence they won. Contradictory pairwise judgments are eliminated through majority voting, i.e. if the same pair of translations has been evaluated by more than two users, this will be replaced by one judgment according to the majority of the users. Cases of equal disagreement are removed from the test set.

3.2 Correlation with human judgments

In this section, the aim is to show that it is possible to construct a ranking mechanism which produces ranks that correlate with those by humans.

Experiment setup The two versions of the ranking mechanism are trained with ML, for all language pairs, learned on ranks produced by human annotators. The mechanism is tested on unseen data and the correlation with the original human ranks is measured. As null hypothesis, it is tested that the machine ranking on the unseen data has zero correlation with the human ranks. The experiment is performed with a cross-validation with 10 folds (Breiman et al., 1984) over the entire data set. The correlation is measured with Kendall's τ. The null hypothesis is tested with a two-tailed test under a confidence level $\alpha = 0.05$.

lang. dir.	adv. ranking mech.		basic ranking mech.	
de-en	0.276	$(4 \cdot 10^{-272})$	0.261	$(1 \cdot 10^{-244})$
en-de	0.165	$(1 \cdot 10^{-28})$	0.151	$(1 \cdot 10^{-38})$
es-en	0.217	$(3 \cdot 10^{-56})$	0.105	$(4 \cdot 10^{-03})$
en-es	0.119	$(4 \cdot 10^{-20})$	0.109	$(8 \cdot 10^{-06})$
fr-en	0.194	$(1 \cdot 10^{-58})$	0.177	$(2 \cdot 10^{-40})$
en-fr	0.209	$(7 \cdot 10^{-141})$	0.200	$(5 \cdot 10^{-112})$

Table 3.6: *Correlation of two versions of the ranking mechanism with human rankings. The correlation is measured with Kendall's τ. In brackets the p-value for the two-tailed test for the null hypothesis of zero correlation, with a statistical significance level $\alpha = 0.05$.*

Results The correlation results are shown in Table 3.6. It holds that every p-value is less than the confidence level $\alpha = 0.05$. This rejects the null hypothesis that the predicted rankings have zero correlation with human rankings, for both versions of the mechanism in all language directions.

3.3 Significance of the ranking mechanism

Apart from showing that the ranking mechanism has some correlation with the human rankings, it is important to show that this correlation is significant against some baselines.

Experiment setup The ranking mechanism performance is compared with two unintelligent rankings: a random ranking and a ranking in the alphabetical order of the systems. It is enough to compare only the the version of the ranking mechanism with the lowest performance. In order to make the comparison in terms of statistical significance, the empirical confidence intervals of Kendall's tau are observed, by applying bootstrap resampling ($\alpha = 0.05$) on the test sets (paragraph 2.7.1.1).

Results The results can be seen in Table 3.7. The unintelligent rankings produce very low correlation scores, including a big amount of ties which result into negative τ values. The basic ranking mechanism is significantly better than all of them.

lang. pair	basic ranking mech.		alphabetical		random	
de-en	0.261	(±0.013)	-0.041	(±0.014)	-0.144	(±0.014)
en-de	0.151	(±0.014)	-0.030	(±0.014)	-0.165	(±0.015)
es-en	0.105	(±0.020)	-0.063	(±0.021)	-0.182	(±0.020)
en-es	0.109	(±0.020)	-0.037	(±0.020)	-0.170	(±0.020)
fr-en	0.177	(±0.017)	-0.024	(±0.017)	-0.175	(±0.017)
en-fr	0.200	(±0.014)	-0.059	(±0.014)	-0.145	(±0.014)

Table 3.7: *Correlation of the basic version of the ranking mechanism with human rankings, compared with a random and an alphabetical ranking. The correlation is measured with Kendall's τ. In brackets the empirical confidence intervals with a statistical significance level $\alpha = 0.05$, based on bootstrap resampling.*

It must be noted that negative values in τ indicate a bad correlation (Section 2.7.1.1), in contrast to other correlation metrics where the absolute value of the correlation score matters. With the confidence intervals considered, the basic version of the ranking mechanism performs significantly better than a random or a fixed alphabetical order of ranks.

3.4 Comparison with other ranking methods

The ranking mechanism is compared with several other state-of-the art methods for ranking. This comparison has been part of the Shared Task on Quality Estimation of the Eighth Workshop on Statistical Machine Translation (WMT13; Bojar et al., 2013b).

Experiment setup Third-party QE ranking systems were not publicly available, so it was not possible to re-train and test these systems in the exact experiment setting as above. In order to still have a fair comparison among the systems, the official results of WMT13 are presented, which introduces a slight variation on the data setting, as compared to the rest of the experiments of this chapter.

The training set uses the WMT evaluation data from the years 2009-2012, whereas the methods were tested with the data from the year 2008. The correlation scores are re-computed to present Kendall's tau similarly to previous experiments (as described in Section 2.7.1.1). The experiment is applied on two language directions, German-English and English-Spanish.

Two variations of our ranking mechanism were submitted as part of the shared task, both using Logistic Regression with SFSS: The basic ranking mechanism as presented above and a preliminary version of it, which excludes the tokenisation features.

Compared ranking methods The CNGL submissions (Biçici, 2013) rank the MT outputs after predicting individual continuous values, using a similarity threshold for each alternative translation. Their model is trained with SVM Regression with a RBF kernel, to predict the F_1 score, computed against reference translations. They use an extensive feature set with Referential Translation Machines (Biçici et al., 2013), whereas feature selection with Partial Least Squares (PLS) is applied in one of the submissions.

The DCU submissions (Almaghout and Specia, 2013), available only for German-English, follow a ranking approach similar to ours, by using a binary Logistic Regression classifier on all pairwise comparisons of the ranking list (Section 3.6), but with a different feature set. The feature set contains six CCG features. A second submission combines the aforementioned CCG features with 80 generic QE features (Specia, 2011).

UPC (Formiga et al., 2013a) also follow a similar ranking approach, based on a binary classifier on all pairwise comparisons. This method uses a Random Forest classifier including ties. Two versions are submitted, based on two different feature sets of about 90 features, including features of semantic analysis.

UMAC (Han et al., 2013) predicts ranking values as a multiple classification problem, using Naïve Bayes and SVM. CMU (Hildebrand and Vogel, 2013), submitted only for English-Spanish, follows a method similar to the one used for n-best list reranking.

Results The results for the comparison on German-English can be seen in Table 3.8. The basic ranking mechanism presented above scores significantly better than all other systems. The basic ranking mechanism for English-Spanish ranks (Table 3.9) is in the second position with $\tau = 0.09$.

system ID	τ
basic ranking mechanism	* 0.31
preliminary ranking mechanism	* 0.28
CNGL SVRPLSF1	0.17
CNGL SVRF1	0.17
DCU CCG	0.15
UPC AQE+LeM+ALGPR+LM	0.11
UPC AQE+SEM+LM	0.11
DCU baseline+CCG	0.00
Baseline Random-ranks-with-ties	-0.12
UMAC EBLEU-I	-0.39
UMAC NB-LPR	-0.49

** In the same position based on confidence intervals of confidence level $\alpha = 0.05$*

Table 3.8: *Comparison of the basic system with other systems of the Task 1.2 of the WMT13 Quality Estimation shared task for German-English (Bojar et al., 2013a) in terms of correlation with human rankings, using metric Kendall's τ*

system ID	τ
CNGL SVRPLSF1	0.16
CNGL SVRF1	* 0.13
basic ranking mechanism	* 0.09
preliminary ranking mechanism	0.04
UPC QQE+LeM+ALGPR+LM	-0.03
UPC AQE+LeM+ALGPR+LM	-0.05
CMU BLEUopt	-0.11
Baseline Random-ranks-with-ties	-0.18
UMAC EBLEU-A	-0.27
UMAC EBLEU-I	-0.34
CMU cls	-0.63

** In the same position based on confidence intervals with confidence level $\alpha = 0.05$*

Table 3.9: *Comparison of the basic system with other systems of the Task 1.2 of the WMT13 Quality Estimation shared task for English-Spanish (Bojar et al., 2013a) in terms of correlation with human rankings, using Kendall's τ*

3.5 Comparison with reference-based evaluation metrics

In this experiment, it is tested whether the ranking mechanism using reference-agnostic machine-learning on sentence-level human rankings can perform better or comparable to state-of-the-art reference-aware automatic metrics.

Experiment setup A set of a sentences translated by many systems is ranked on the sentence-level by both the ranking mechanism and every state-of-the-art automatic metric. Automatic metrics produce scores by measuring differences between the translation and the references, whereas the ranking mechanism applies a statistical model based on quality features acquired on the translation itself. For every metric, the null hypothesis is that the correlation of the metric with human rankings is better than the correlation of the machine-learned ranking.

Here, the comparison is done using the advanced ranking mechanism. The metrics compared are: Bilingual Evaluation Understudy (BLEU) with sentence-level smoothing, METEOR, rgbF, Translation Error Rate (TER) and Word Error Rate (WER) (Section 2.7).

Results The results (Table 3.10) reject the null hypothesis for every comparison. This indicates that even without access to reference translations, the correlation of the advanced ranking mechanism with human judgments is comparable to or higher than the automatic metrics.

lang. pair	ranking	BLEU	METEOR	rgbF	TER	WER
de-en	0.28	-0.22 ‡	0.23 ‡	0.16 ‡	-0.02 ‡	0.15 ‡
en-de	0.16	-0.42 ‡	0.13 ‡	0.10 ‡	-0.09 ‡	-0.15 ‡
es-en	0.22	-0.19 ‡	0.22 ◇	0.16 ‡	-0.02 ‡	0.13 ‡
en-es	0.12	-0.21 ‡	0.12 ◇	0.09 ◇	-0.10 ‡	0.08 ‡
fr-en	0.19	-0.18 ‡	0.20 ◇	0.15 ‡	-0.02 ‡	0.16 ‡
en-fr	0.21	-0.12 ‡	0.18 ◇	0.15 ‡	-0.03 ‡	0.15 ‡

‡: ranking mechanism is significantly better than metric
◇: ranking mechanism is significantly as good as metric

Table 3.10: *Comparison of the best version of the ranking mechanism with state-of-the-art reference-aware automatic metrics concerning correlation with human judgments (Kendall's τ). Statistically significant comparisons based on Confidence Intervals of $\alpha = 0.05$ are indicated.*

The most powerful metric is METEOR, which already includes some tuning of its components on human rankings. The ranking mechanism for de-en and en-de which has had feature engineering particularly for these language pairs, is significantly better than every other system including METEOR. For the rest four language directions (Spanish and

French), where feature engineering from de-en was re-used, the ranking mechanism is significantly as good as METEOR.

Apart from the success of our method, these results suggest that elaborate feature functions and/or machine learning methods may provide more information for the relative quality of translations than scoring methods based on comparisons with a single reference.

3.6 Machine learning methods

As explained earlier, the ranking mechanism is based on a statistical model produced through machine learning. The process of ranking the alternative translations of each sentence is decomposed into pairwise comparisons, allowing the use of a variety of robust pairwise classifiers. In Section 2.5, several machine learning algorithms were outlined to perform ranking as explained in Section 2.4. It is known that the performance among different machine learning algorithms may vary, given the type of the problem and the nature of the data. This leads to the need to investigate how different machine learning algorithms perform for the ranking mechanism. The goal of this comparison is to indicate the most promising methods to be used in further experiments. A detailed investigation of the ranking mechanism with several machine learning methods follows, including an assessment of which methods give the best results.

Experiment setup The investigation of algorithms was run on German-English. Various classification algorithms were trained using the same feature set as the *basic ranking mechanism* presented earlier (Section 3.1.1). The comparative run includes the following basic classifiers: Decision Trees (Section 2.5.6), Gaussian Naïve Bayes (Section 2.5.1), kNN (Section 2.5.2), LDA (Section 2.5.4), Logistic Regression with L2 Regularisation and with Stepwise Feature Set Selection (Section 2.5.3). Additionally, several ensemble classifiers (Section 2.5.7) are build on top of Decision Trees. These include the Adaptive Boosting (AdaBoost), Bagging, Extra Randomized Trees (ExtRa Trees), Gradient Boosting and Random Forest.

Results The results of the comparison of various learning methods can be seen in Table 3.11. According to the basic metric, Kendall's τ, five learning methods share the first statistically significant position, based on the confidence intervals obtained through bootstrap resampling at a confidence level of a=0.05.

The highest score is given by a classifier which ensembles Decision Trees, namely the Gradient Boosting, followed by Logistic Regression with SFSS and the AdaBoost classifier. Logistic Regression with SFSS and LDA outperform all non-boosted learning methods in terms of τ, closely followed by Gaussian Naïve Bayes.

Plain Decision Trees fail to achieve a significant correlation with human ranks. All the other algorithms have some significant correlation with human ranks, as the p-value for the null hypothesis of zero correlation satisfies the confidence level $a = 0.05$. As expected, the

algorithm	τ		p-value	NDCG
Gradient Boosting	‡ 0.265	(± 0.013)	$3 \times 10-^{253}$	0.735
Logistic Regression SFSS	‡ 0.261	(± 0.013)	$1 \times 10-^{244}$	0.730
AdaBoost	‡ 0.259	(± 0.013)	$7 \times 10-^{220}$	0.733
LDA	‡ 0.257	(± 0.013)	$4 \times 10-^{225}$	0.733
Gaussian Naïve Bayes	‡ 0.247	(± 0.013)	$3 \times 10-^{197}$	0.726
Bagging	0.205	(± 0.014)	$1 \times 10-^{108}$	0.716
kNN	0.196	(± 0.013)	$1 \times 10-^{137}$	0.715
Random Forest	0.191	(± 0.013)	$2 \times 10-^{102}$	0.710
Extremely Randomized Trees	0.166	(± 0.013)	$2 \times 10-^{054}$	0.701
Logistic Regression L2	0.129	(± 0.013)	$1 \times 10-^{038}$	0.699
Decision Trees	0.024	(± 0.014)	$1 \times 10-^{001}$	0.639

Table 3.11: *Comparison of various machine learning algorithms on the MT output ranking task for German-English (10-folded cross-validation) in terms of correlation with human rankings as measured with Kendall's tau and Normalised Discounted Cumulative Gain. Kendall's tau includes an empirical confidence interval on bootstrap resampling and the p-value of the two-tailed test for the null hypothesis of zero correlation ($a = 0.05$)*

classifiers which ensemble Decision Trees perform significantly better than the individual Decision Trees.

Logistic Regression with Stepwise Feature Set Selection is chosen for the *basic ranking mechanism* presented in this work, because it combines the good performance with the ability to interpret the statistical model. The highest scoring algorithm, Gradient Boosting, is thereof chosen for the *advanced ranking mechanism*.

3.7 Elimination of ties by weighting pairwise decisions

As explained in Section 2.4.2, every rank is a result of aggregating many pairwise decisions. Nevertheless, the recomposition of a rank from pairwise decisions often results in a tie. This is a result of contradictory pairwise decisions. Yet, it may be that not all aggregated pairwise decisions are of equal importance, and the confidence of the classifier may be a good indicator. Based on this intuition, part of this work is to examine a new way of recomposing ranks, which we call *soft rank recomposition* (Section 2.4.2.2).

In this experiment it is examined whether it is possible to reduce the number of predicted ties by using the "soft rank recomposition", without a negative effect on the results.

Experiment setup In the first part of the experiment, we train an instance of the *basic ranking mechanism* for every language pair, based on a binary classifier of Logistic Regression with Stepwise Feature Selection. Subsequently we use the trained model to predict the results of the pairwise comparisons for the test set. Then, the produced pairwise decisions are converted back to ranks with both "soft" and "hard" recomposition. We measure the impact of the soft recomposition against hard recomposition on the amount of generated ties, and on the correlation with human rankings based on Kendall's τ and NDCG. The training and the evaluation is repeated within a cross-validation of 10 folds and the scores from all folds are averaged.

Results The contribution of the soft recomposition of the ranks (Section 2.4.2.2) can be read off Table 3.12. The ties are measured by the percentage of the pairwise comparisons which are tied. The soft recomposition achieves equal or higher τ correlation coefficients and less ties for all the language pairs. At the same time, on all language pairs but one, the number of the ties is reduced by a percentage of 40-51%. The soft recomposition also has a positive effect on the secondary metric NDCG.

Given the success of this method, soft recomposition is considered as part of the default settings on both the *basic* and the *advanced ranking mechanism*.

lang.	predicted ties			Kendall's τ		NDCG	
direction	hard	soft	diff.	hard	soft	hard	soft
de-en	1.54%	0.79%	-49%	0.253	0.261	0.719	0.730
en-de	0.99%	0.48%	-51%	0.146	0.151	0.719	0.725
es-en	0.00%	0.00%	0%	0.105	0.105	0.737	0.737
en-es	2.34%	1.40%	-40%	0.100	0.109	0.701	0.709
fr-en	2.46%	1.36%	-45%	0.165	0.177	0.733	0.748
en-fr	1.98%	1.08%	-46%	0.191	0.200	0.669	0.683

Table 3.12: *The impact of soft rank recomposition on the basic ranking mechanism, as measured with cross-validation over the entire dataset. The percentage of the predicted ties, Kendall's tau and NDCG are compared.*

3.8 Effect of MT system performance on the amount of ties

As indicated, the percentage of ties varies among language pairs. It is however noteworthy that there can be no direct comparison between different language pairs, since the data for each of them differ a lot. One aspect of these differences, is how the MT systems in each language pair compare to each other in terms of their translation performance. Therefore, we proceed with analysing the contribution of the performance differences between MT systems.

Additionally, the results in Table 3.12 leave open the question why the ranking mechanism for Spanish-English (es-en) does not predict any ties in the first place. As noted earlier, there are different systems participating in every language pair and this is also the case for Spanish-English. Having seen the effect of the performance comparison between different

MT systems, we try to explain why the data for Spanish-English are particularly different. For this purpose we perform a comparison concerning the average document-level BLEU difference of the MT systems that participate in pairwise comparisons.

Experiment setup For this experiment, we gather data from all language directions. For every pair of systems whose segments are compared, we measure the number of predictions that result into ties and the difference of the document-level BLEU score between these two particular systems. The comparative analysis is plotted in Figure 3.1. The system pairs that exhibit the same amount of ties (X axis) are grouped together and their pairwise differences in document-level BLEU scores are averaged among all system pairs with the same number of ties (Y axis).

Results The results in in Table 3.13 indicate that when the BLEU difference between two systems is high in average, pairwise comparisons result in a small amount of ties, whereas the amount of ties gets increased for comparisons between systems which have smaller difference in BLEU. This leads to the conclusion that the closest two MT systems are in terms of translation performance, the hardest it is for the automatic mechanism to distinguish the difference between them.

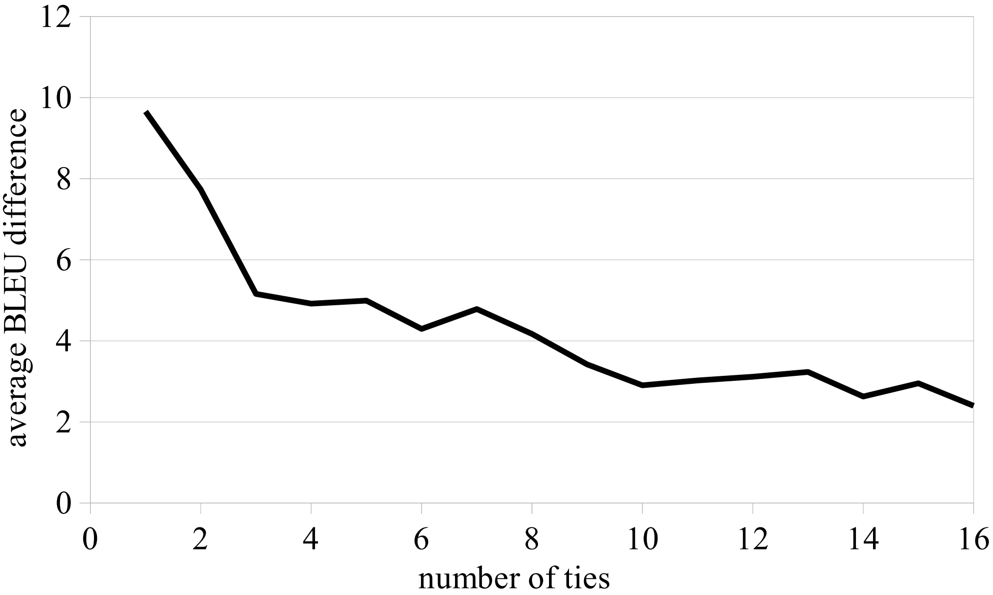

__Figure 3.1:__ The higher the BLEU difference between systems, the less ties appear. Stacked linear graph of the document-level BLEU difference between MT systems participating in pairwise comparisons (Y axis) against the number of ties for the pairwise segment-level comparisons of the respective MT system pairs (X axis) measured on aggregated data from all language pairs

Indeed, the MT systems compared in the case of Spanish-English have the highest document-level BLEU difference than all the other language pairs, reaching 9.38 points BLEU, which is 0.7 points higher than the next BLEU difference which is observed for

lang. pair	de-en	en-de	es-en	en-es	fr-en	en-fr
BLEU difference	7.47	6.23	**9.38**	8.52	8.49	8.65

Table 3.13: *Document-level BLEU score difference for the MT systems participating in the pairwise comparisons in each language pair.*

English-French. This leads to the assumption that one of the reasons for the lack of ties for Spanish-English relies on the high difference of performance between the systems contained in the data of this language pair.

3.9 Weighting pairwise decisions on various algorithms

The elimination of ties is based on weighting the pairwise decisions by the classification probability prior to re-composing the ranking for each sentence. In a previous experiment (Section 3.7), it has been shown that the elimination of ties has a positive or at least no negative effect, when used for recomposing the ranks predicted with Logistic Regression with Stepwise Feature Set Selection. Since every binary classification algorithm has a different way of computing probabilities, an additional experiment may indicate the applicability of the methods on various algorithms.

Experiment setup The investigation across different algorithms takes place on a similar setup as previously (Section 3.6), based on a 10-folded cross-validation on German-English data. The same settings for each one of the learning methods are also used here.

Results The impact of soft rank recomposition on various learning methods can be seen in Table 3.14. Soft recomposition results into a severe reduction on the number of the predicted ties for most of the algorithms. This does not hold for Gaussian Naïve Bayes (Gaussian NB) and LDA, where a small amount of ties is induced. For LDA in particular, the induced ties do not have a negative effect on the correlation, while for Gaussian Naïve Bayes, the predicted probabilities seem to not be good for weighting pairwise decisions, as their use results in discordant pairs.

For the rest of the algorithms, the correlation with human ranks, as measured by Kendall's τ or NDCG increases or remains in the same level. In the cases of four algorithms, Gradient Boosting, AdaBoost, kNN and Logistic Regression with L2 Regularisation, the few predicted ties are totally eliminated. For algorithms with lower initial performance and bigger amount of predicted ties, such as Bagging, kNN, Random Forest and ExtRa Trees, the gains with soft rank recomposition are more significant.

algorithm	predicted ties			Kendall's τ		NDCG	
	hard	soft	diff.	hard	soft	hard	soft
Gradient Boosting	0.42%	0.00%	-100%	0.260	0.265	0.733	0.735
LogReg SFSS	1.54%	0.79%	-49%	0.253	0.261	0.719	0.730
AdaBoost	0.38%	0.00%	-100%	0.255	0.259	0.732	0.733
LDA	0.15%	0.18%	+19%	0.257	0.257	0.733	0.733
Gaussian NB	0.00%	0.61%	(+62)	0.253	0.247	0.734	0.726
Bagging	6.52%	0.39%	-94%	0.138	0.205	0.681	0.716
kNN	2.56%	0.00%	-100%	0.167	0.196	0.705	0.715
Random Forest	6.37%	0.55%	-91%	0.130	0.191	0.680	0.710
ExtRa Trees	6.75%	0.72%	-89%	0.101	0.166	0.668	0.701
LogReg L2	0.38%	0.00%	-100%	0.132	0.137	0.699	0.700
Decision Trees	11.48%	9.66%	-16%	0.005	0.024	0.631	0.639

Table 3.14: *Impact of soft rank recomposition applied on various binary classifiers, as measured with evaluation over the entire dataset (10-folded cross-validation). The percentage of the predicted ties, Kendall's tau and NDCG are compared.*

3.10 Summary

Machine learning was successfully used as part of a mechanism that is able to perform preference ranking on alternative machine translation outputs. Statistically significant correlation with human judgments was observed and further comparisons indicate that the performance of the mechanism is better than all the baselines and other ranking methods. Most importantly, the ranking mechanism performs significantly better than the state-of-the-art reference-aware automatic metrics, for the language pairs where focused feature engineering took place. The ranking mechanism also beats other metrics in language pairs where the feature engineering from other language pairs was adopted, apart from one reference-aware metric, METEOR, which is on par with the ranking mechanism. This suggests that elaborate features and machine learning may provide more information about relative translation quality than direct comparison with references.

The fact that ranking was decomposed into pairwise decisions allowed the integration of several machine learning algorithms with positive results. The best performing algorithms are Gradient Boosting, Logistic Regression, with Stepwise Feature Set Selection, AdaBoost, LDA and Gaussian Naïve Bayes.

The recomposition of a ranking from pairwise binary classifier decisions faces the problem of creating ties, as a result of unclear and contradictory pairwise decisions, provided also that the test sets have been built with the assumption that there are no ties included. This was solved by weighting the pairwise classification decisions with their prediction probabilities before aggregating them in order to recompose full rankings. This method, referred to as soft rank recomposition, may have different impact to ranking prediction when applied on various algorithms. It has a positive impact when applied to the most of the algorithms investigated, including the three top-performing algorithms. Concerning only two algorithms (LDA and Gaussian Naïve Bayes), soft rank recomposition yielded a small, yet not statistically significant loss on the correlation with human judgments. Finally, it is also shown empirically, that when the difference in the overall performance of the MT systems that are being compared is small, more ties appear, and vice versa.

Chapter 4

Feature Engineering

The supervised machine learning method for ranking (Section 2.5) requires the extraction of features. The features are numerical values reflecting various linguistic or statistical phenomena of the sentence in focus. These phenomena are expected to be related with the quality of the translation and therefore they can be beneficial as factors of the QE model.

The entire spectrum of features considered for the ranking mechanism, including their theoretical background, is described in Section 2.3. Nevertheless, their development and utilisation has been the result of an iterative feature engineering process, aiming to select and use the features in an optimal way, in line with several ML methods. This process has gone through several phases, whereas new features and more advanced statistical and ML methods are gradually employed in every phase.

The aim of this chapter is therefore to describe the feature engineering process by providing detailed information on the steps taken in every phase. Through this process, the effectiveness of several features is examined via statistical analysis and/or hypothesis testing and as a result, several useful conclusions are drawn.

A short overview of the phases followed in each phase can be seen below:

Preliminary ranking mechanism:

1. Create a pool of preliminary features based on grammatical phenomena
2. Investigate contribution of features using feature scoring
3. Select minimal feature sets using human intuition
4. Train with a basic ML method with various feature combinations

Basic ranking mechanism:

1. Augment preliminary feature set with baseline features from *absolute QE*
2. Train with Feature Set Selection embedded in the ML method
3. Investigate contribution of features

Advanced ranking mechanism:

1. Add advanced grammatical and statistical features

2. Select features with the RFECV based on a SVM
3. Train advanced ensemble classifiers

The separation of the development in phases allows us to introduce new feature sets and ML methods between subsequent phases, but also means that the experiment settings vary slightly, e.g. in terms of data or evaluation sets, which also increases gradually from phase to phase. Therefore, not all experiment results are directly comparable between each other, but they are detailed here, due to the useful conclusions that can be drawn through the comparisons within each phase.

4.1 Preliminary ranking mechanism

The aim of the preliminary ranking mechanism is to indicate the usability of grammatical features against shallow methods (such as n-gram scoring). The focus is on devising a minimal feature set, including some grammatical features. The usability of this feature set shall be indicated within a model trained with a simple classifier.

Simple classifiers, such as Naïve Bayes, are useful for a preliminary experiment, since they can be implemented and trained quickly, whereas quick repetitive training allows experimenting with a wide variety of feature combinations in order to confirm several hypotheses. Although there are learning methods able to deal with a broad set of features and the contribution of some features is likely to change, if a more complex method or feature set is used, this preliminary phase aims at highlighting the contributions of particular features by using a basic learning method.

Therefore we restrict the minimal feature set to 5 features per experiment, chosen from a broader pool of 10 features. The pool of 10 features considered in this development phase is shown in Table 4.1.

Count-based: Number of words in the source and target sentences
Language Model: Smoothed n-gram probability of the entire target sentence for a large Language Model of order 5.
Unknown words: words unseen in the training corpus of the LM.
Parsing: PCFG features for both source and target side:

- the number of valid parses in the n-best list of the parser,
- the sentence likelihood, i.e. summing out all parse trees: $P(w)$,
- the joint likelihood of the best tree and its words: $P(t, w)$,
- the averaged likelihood of all trees in the n-best list,
- the number of occurrences of VPs and NPs in the PCFG parses of both the source and the target

Table 4.1: *Features considered for the preliminary feature set.*

In this phase we also derive features from both the source sentence and the translation outputs. As explained in Chapter 2, the learning methods operate based on machine learning instances that represent two MT outputs for the same source sentence. This instance is represented with one binarised feature vector which may contain the features from the source and both feature outputs. Despite the source features being the same for both outputs, we include them as there is still a case that their existence in the feature vector may be useful for measuring adequacy, due to interactions with the target features. This will be examined in more detail in Section 4.2.1.

Experiment setup The machine learning core of the system is built with Naïve Bayes (as explained in Section 2.4.1). The pairwise classifiers for the task are learned using the German-English test-set of the WMT 2008 and 2010 (about 700 source sentences including 3,500 translations or 14,000 pairwise comparisons). For testing, the classifiers are used to perform ranking on a test set of 184 sentences (including about 920 translations or 3,600 pairwise comparisons) which have been kept apart from the 2010 data, after filtering out those that do not contain contradictions among human judgments.

4.1.1 Feature scoring

Simple classification methods are expected to perform better, given a small group of statistically independent features, so we need quick methods to identify features that are expected to perform well. As a quick way of acquiring knowledge about the usability of the features, we performed feature scoring based on a set of scoring methods, including Relief, Information Gain, Gini Index, Relevance and Distance (Section 2.6). For the scoring methods that required discrete feature values (all apart from Relief), prior to scoring the feature values were discretised to 100 values. The scoring was performed on the training part of the WMT 2010 data set.

The feature scoring results are shown in Table 4.2. The top features according to the scoring methods are the LM probability, which scored first for Information Gain, Gini Index and Relevance, the number of unknown words according to the distance scoring and the number of parse trees according to Relief.

The results are very encouraging for the use of grammatical features. The parse likelihood scores very high and gets the second position for most scoring methods, whereas Relief scores the number of the parse trees on the highest position. A general observation is that the source features individually have very low or zero scores for all scoring methods.

4.1.2 Feature observations via model estimation

As explained above, the purpose of the preliminary development phase is to identify a minimal set of innovative features and confirm their applicability using a basic ML method. The experiment is repeated with various feature combinations, based on the observations provided by the feature scoring methods. After repeating the model estimation with various feature sets, conclusions are drawn in three basic directions:

	dist	infgain	gini	rel	relief
target features					
Number of unk	0.013	0.033	0.011	0.133	0.042
Avg n-best tree likelihood	0.011	0.057	0.018	0.181	0.027
LM probability	0.009	0.068	0.022	0.248	0.026
Best tree joined likelihood	0.008	0.058	0.019	0.217	0.012
Sentence likelihood	0.007	0.050	0.017	0.206	0.012
Number of parse trees	0.007	0.028	0.010	0.125	0.043
Number of tokens	0.002	0.011	0.004	0.089	0.017
Number of VPs	0.002	0.007	0.002	0.080	0.018
Number of NPs	0.001	0.007	0.002	0.084	0.023
Number of dots	0.000	0.001	0.000	0.008	0.007
Number of commas	0.000	0.000	0.000	0.017	0.000
source features					
Number of finite verbs	0.000	0.000	0.000	0.000	0.006
Avg n-best tree likelihood	0.000	0.000	0.000	0.000	0.006
Number of NPs	0.000	0.000	0.000	0.000	0.005
Number of parse trees	0.000	0.000	0.000	0.000	0.004
Number of nouns	0.000	0.000	0.000	0.000	0.004
Number of VPs	0.000	0.000	0.000	0.000	0.002
Number of commas	0.000	0.000	0.000	0.000	0.002
Number of PPs	0.000	0.000	0.000	0.000	0.002
Number of tokens	0.000	0.000	0.000	0.000	0.001
Sentence likelihood	0.000	0.000	0.000	0.000	0.001
Best tree joined likelihood	0.000	0.000	0.000	0.000	0.001
Sentence likelihood	0.000	0.000	0.000	0.000	0.001

Table 4.2: *Information gain and Gini Index, Relevance and Distance for preliminary feature set.*

Grammatical features versus LM probability Scoring sentences with their Language Model (LM) probability is a common practice for most SMT systems. The goal of this experiment is to define features beyond the Language Model scoring. In this experiment we test whether grammatical features can be more useful than the score from the Language Model.

We train three QE ranking models using a minimal feature set as a baseline. We measure the performance of the model with that minimal feature set (number of unknown words and sentence length and additionally two augmented versions of this set: one adding the Language Model scores and one adding the grammatical features instead.

The results can be seen in Table 4.3, where the correlation of each setting against human rankings is measured. Adding the LM probability to the minimal feature set yields no improvement on the correlation. However, replacing the LM probability with three gram-

features	τ
unk, len	0.20
unk, len, LM probability	0.19
unk, len, n_{trees}, VPs, summed n-best log-likelihood	0.26
unk, len, n_{trees}, NPs, best-tree log-likelihood	0.23
unk, len, n_{trees}, VPs, avg n-best log-likelihood	0.21
unk, len, n_{trees}, VPs, best-tree log-likelihood	0.25

unk: number of unknown words
len: number of sentence tokens
n_{trees}: number of generated trees in the parsing n-best list

Table 4.3: *Augmenting a minimal feature set with grammatical features achieves better ranking performance than when adding LM probability.*

matical features achieves an improvement of $\tau = 0.07$, which is statistically significant. Therefore, we can conclude that the grammatical features are useful for a minimal ranking mechanism and that they perform better as features than the LM probability.

NPs versus VPs Having as a goal to restrict the size of the minimal feature set to the 5 most useful features, we compare the training of the model with two different settings. The first setting is based on the minimal feature set including the number of unknown words, the number of sentence tokens, the number of n-best parse trees, the joint log-likelihood of the best tree and the number of VPs. The second setting replaces the VPs of the previous one with the NPs.

The result can be seen in the second part of the Table 4.3. Counting VPs is more helpful than counting NPs. Although the improvement between the feature sets with NPs and VPs is relatively small and not conclusive, the feature set including VPs is the only one that yields a significant improvement over the minimal feature set. Therefore we can conclude that using VPs is preferable. In order to adhere to a minimal set of 5 features, we will proceed in using the most successful feature in further experiments.

Variations of parse probability As seen above, three variations of the parse probability can be extracted from the PCFG parsing process of the coarse-to-fine n-best PCFG parsing process (Section 2.3.1). Using the sentence log-likelihood (summing the log-likelihoods from all the parse trees) achieves significantly better performance than just averaging the joint log-likelihood of all the trees in the n-best list. Though, there is a very small difference in the performance between the best-tree log-likelihood and the summed n-best log-likelihood.

4.2 Basic ranking mechanism

Since the preliminary ranking mechanism was created as a proof-of-concept and a means for initial observations, this section is presenting its extensions towards a basic ranking mechanism that can perform significantly better over a bigger amount of data. The preliminary feature set with the grammatical features is augmented with several new state-of-the-art features from related work in absolute (i.e. non-comparative) QE. Apart from the fact that the experiments take place on the full amount of data, a more robust learning method (Logistic Regression) is used, so that it can take better advantage of the multitude of features. Additionally, such a model can be relatively interpretable, leading to useful conclusions about the features.

Experiment setup Whereas the machine learning core of the preliminary system was built with Naïve Bayes, in the basic mechanism we are introducing Logistic Regression. The experiments were performed using the full amount of the available data of WMT2008-2014 for German-English, as detailed in Section 3.1.2. It contains about 4,500-6,600 sentences per language pair translated by various MT systems. The translations of each sentence were grouped randomly into batches of 5 translations, which were manually ranked by annotators.

In order to reduce the complexity of the iterations, the experiments are run on the 6th fold of the 10-folded cross-validation, as that was the fold whose scores were closest to the cross-folded scores during preliminary experiments.

4.2.1 Source/target features and ratios

Related work on absolute QE (e.g. Felice and Specia, 2012), apart from the use of source features, has suggested the introduction of features that are result of the ratio of source against target features. This suggestion is based on the intuition that particular characteristics from the source should be conveyed in the target, in a way that can be seen through the ratios. For this purpose, for every monolingual feature that is applicable to both the source sentence and the target sentences, a new feature is created, as a result of the ratio of the source value to the respective target value.

Since this assumption was originally made for absolute QE, in this experiment, we are examining whether in this preliminary feature set of comparative QE it is preferable to include feature ratios, separate source and target feature values, or only target feature values.

feature set	τ
preliminary (source/target ratios)	0.168
preliminary (source+target)	0.192
preliminary (target)	0.192

Table 4.4: *Comparison of various ways of incorporating source features into the feature set.*

Indeed, the results in Table 4.4 indicate that calculating the ratios of the features lead to considerably lower results than giving the source and target features separately. These observations confirm that during feature engineering it is a good practice to not pre-calculate relations between features prior to the learning process. We can assume that the reason that pre-calculating hurts the performance of the model, is that it forces the learning method to indirectly include a potentially harmful estimator in the model that might have otherwise been assigned a negligible weight during the model estimation.

Additionally, there is no indication of a better prediction when source features complement the target features. This confirms the observation of Section 4.1.1 (also suggested in previous work by Zwarts and Dras, 2008, albeit with more limited experimentation), that the source features do not contain any useful knowledge for the comparison of the translations.

4.2.2 Introducing Logistic Regression and contrastive scoring

In this experiment, some improvements on the preliminary ranking mechanism are tested. Machine learning is improved, as Logistic Regression is introduced as a learning method, since it can handle more efficiently a multitude of features without the assumption of independence that Naïve Bayes relies on.

Additionally, following suggestions from related work in absolute QE (Soricut et al., 2012), a new feature of contrastive scores is introduced. Contrastive evaluation metrics score the n-grams of each produced translation, using the translations by the other competitive systems as multiple pseudo-references. Here, we experiment with contrastive METEOR (as explained in Section 2.3.1.5).

feature set / model	τ
preliminary	0.192
+ Logistic Regression	0.202
+ contrastive METEOR	0.249

confidence interval ($\alpha = 0.05$): τ = 0.014

Table 4.5: *Improvements to the correlation with the human rankings by re-training with Logistic regression and adding contrastive METEOR*

The results of the experiment can be seen in Table 4.5. The usage of Logistic regression results into a considerable improvement on the scores. The addition of contrastive scoring nevertheless introduces a statistically significant improvement of $\tau = 0.047$ over the models trained with the preliminary feature set.

When more than one systems perform the same translation, it may often be the case that they collectively convey more correct information than each of them. This property can be useful for the comparison of the translations. A system output that agrees more with the majority of the other systems, is more likely to be preferred as the best translation, as shown by the positive contribution of the contrastive scoring feature.

4.2.3 Using features from absolute Quality Estimation

The contribution of more features from absolute Quality Estimation is considered in this experiment. This includes the set of the “Vanilla" features that were publicly available as the baseline for the QE Shared Task in WMT12 (Callison-Burch et al., 2012). These features were proven useful for the prediction of absolute values, as compared to the prediction of comparisons or rankings that this work is focusing on.

- number of tokens in the source sentence and the target sentence, average source token length
- LM probability of the source and the target sentence (3-gram)
- ratio of appearances of every target word within the target sentence (type/token ratio)
- average number of translations per source word in the sentence (as given by IBM-1 table thresholded so that $p(t|s) > 0.2$),
- average number of translations per source word in the sentence (as given by IBM-1 table thresholded so that $p(t|s) > 0.01$) weighted by the inverse frequency of each word in the source corpus
- percentage of uni-grams/bi-grams/tri-grams in frequency quartiles 1 (lower frequency words) and 4 (high frequency words) in a corpus of the source language (SMT training corpus)
- percentage of uni-grams in the source sentence seen in a corpus (SMT training corpus)
- number of punctuation marks in the source and target sentence

Table 4.6: *The 17 baseline features from absolute QE (Callison-Burch et al., 2012)*

The baseline feature set includes 17 features, whose description can be seen in Table 4.6. Some of them, such as the count of tokens and the LM probability, coincide with our experiments illustrated above. Several other features are new, including word alignment features based on the IBM model 1, corpus statistics for the source words and counts for the punctuation.[1]

We run the experiment with three settings: one with just the absolute QE features, one with the previously explained feature set and one with a combination of the above. For comparison, we also include the performance of the preliminary feature set.

The ranking correlation scores achieved by the three models can be see in Table 4.7. Both the preliminary feature set and the feature set with the contrastive scores already yield to models significantly better than the absolute QE features. Adding the QE features further improves the correlation by about 7%. It must be noted that due to the different data setup between the development phases of preliminary and the basic ranking mechanism, the numbers appearing in Tables 4.3 and 4.7 are not directly comparable. To allow for a

[1] Punctuation counts were also considered for our preliminary ranking mechanism but they did not qualify for the minimal feature set as they achieved relatively low feature scores on Relief, Information Gain etc.

feature set / model	τ
absolute QE features	0.172
preliminary	0.202
preliminary & contrastive (4.2.2)	0.249
+ absolute QE features	0.266

Table 4.7: *Improvements to the correlation with the human rankings by adding features from absolute QE*

direct comparison in the latter table, the scores corresponding the preliminary feature set have been re-computed to conform with the setup of the basic ranking mechanism.

Further investigation on the extent that absolute QE features are relevant for the purposes of the comparative QE will be discussed upon the coefficient analysis in Section 4.2.5.

4.2.4 Stepwise feature set selection versus L2 Regularisation

Learning methods include several capabilities explicitly for handling features. Two known methods for handling features as part of the learning process are the L2 Regularisation and the Stepwise Feature Set Selection (Sections 2.5.3 and 2.6.3 respectively). Here, we examine the suitability of each of these methods for this feature engineering process.

We train two models with the same settings, one with Logistic Regression with L2 regularisation and one with Stepwise Feature Set Selection. We measure the performance in terms of correlation with human ranking, in order to see which of the two methods is better for the given problem.

Logistic Regression methods	τ	CI ($a = 0.05$)
L2 Regularisation	0.125	± 0.014
Stepwise Feature Set Selection	0.266	± 0.014

Table 4.8: *Comparison between two methods of Logistic Regression in terms of correlation of the produced ranking with the human rankings, both trained on the basic feature set.*

The results of the comparison of the two variations of the learning algorithms can be seen in Table 4.8. The approach of Stepwise Feature Set Selection outperforms the L2 Regularisation with statistical significance. Through further investigation, it can be seen that learning process including L2 Regularisation has lower performance due to the requirement for the normalisation of the features. The normalisation is unable to handle properly the features which contain values equal to the infinite, as is often the case for the parse log-likelihood. In that case, these features may not be represented properly in the model.

4.2.5 Contribution of features

Useful conclusions concerning the contributions of various features can be drawn by looking the estimated beta (β) coefficients of the Logistic Regression model (Section 2.5.3) as a result of the learning process. For every co-efficient, the null hypothesis of it being equal to zero has been rejected with a χ-test. The sign (positive/negative) of the coefficient indicates whether the feature has a positive or a negative contribution the selection of the translation by the humans. Additionally, if the features values have been normalized with their mean and variance, the value of the coefficient may provide indications for the importance of the features on the final decision.

feature name (target sentence)	β
number of unknown words	-0.58
number of tokens	0.50
contrastive METEOR	0.29
number of VPs	0.17
number of n-best trees	-0.17
type/token ratio	-0.14
number of commas	-0.11
sentence parse log-likelihood	0.08
3-gram probability	-0.05
number of dots	0.04

Table 4.9: *The beta coefficients estimated with logistic regression for each feature, in a descending order based on their absolute value. Feature values have been normalized with the mean and the variance of the respective feature.*

The beta coefficients in a model fitted with Logistic Regression can be seen in Table 4.9. An effort to get some useful conclusions from the estimators is given below.

The **number of unknown words** (unk) is a strong indication for an incomplete translation. Although out-of-vocabulary words are not necessary indications for untranslated words, when two translations of the same source have a different amount of unknown words, it is more likely that the one with the most unknown words has failed to translate some of them.

The overall **number of tokens** (len) in the sentence is a positive indication for the better translation. For many translation systems, particularly the statistical ones, it is a common case that some source words are not translated. This is because during the decoding process it is decided that some words/phrases suggested by the translation model reduce dramatically the overall score of the produced sentence, considering also the Language Model. Similar may be the case for syntax-augmented systems (Khalilov and Fonollosa, 2009). Therefore, when a translation has less words than its competitor, it may be the case that a useful word was omitted.

Additional words also occur as a translation error, e.g. when phrases chosen during the decoding of a phrase-based system overlap partially. A special case of this, when the same word is repeated in the generated translation, is handled by the **type/token** ratio, which is given a negative coefficient.

Contrastive METEOR also appears to have a high contribution to the prediction. This confirms the improvement that was shown in a previous experiment, where a possible explanation is given too (Section 4.2.2).

The **number of verb phrases** (VPs) is directly connected with the fluency of the produced translation. The feature value is a result of the parser having tried to analyze the structure of the sentence and identify the existence of one or more verb phrases. Among translation errors, it is more likely that a VP is not formed properly, than having superfluous verb phrases formed by mistake. That may be a reason for the observation that if a translation has more verb phrases than its competitor, it is more likely to be chosen.

Similarly, when the parser analyses a translation, it creates **n-best lists with trees** with alternative analyses of the grammatical structure. Whereas these lists are scored, so that the best parse can be selected, the size of the list can indicate how ambiguous the parse is, as already suggested upon the design on the features (Section 2.3). A translation with fewer n-best trees is a translation with a less ambiguous grammatical structure and this may obviously play a role when comparing translations. The **sentence parse log-likelihood** also has a positive contribution, as another indication of grammaticality (Mutton et al., 2007).

The contribution of **punctuation count** indicates that translation systems often do mistakes with punctuation marks. In particular, it seems more likely to select a translation when it has fewer commas, or when it has more dots. These might be connected with the tension of some systems to erroneously create too many commas or to omit dots, something that obviously is irritating for the readers.

Finally, there is little explanation of the low, albeit negative contribution of the **tri-gram Language Model probability**. Whereas one would expect that a higher model probability would be preferable, this is not the case. One could only assume that this is interacting with some other features, e.g. to give some precedence to the grammatical features against the Language Model.

There can also be conclusions about the features that were not included in the model. It is now possible to identify which of the **absolute QE features** were useful when added to the preliminary feature set. In fact, only the punctuation features, the type/token ratio and the tri-gram probability were assigned a non-zero coefficient, added to the target sentence sentence length, which already existed as a feature in the preliminary feature set. On the contrary, none of the features containing information about the source words or the alignment of target words to source words were deemed of significant importance for the model.

Additionally, we confirm that the **source features** play no role in the comparison between translations (see Sections 4.1.1 and 4.2.1) since these features have been assigned zero coefficients. Despite intuitions why source features can be useful, we may also assume that for the current dataset, the given features do not introduce any useful knowledge about the *relative ability* of the systems to translate these source sentences. Although target-side features are designed to measure fluency as opposed to adequacy, we may here assume that adequacy information is indirectly conveyed through other comparative features, e.g. the count of unknown words, the number of tokens or the contrastive scores.

4.3 Advanced ranking mechanism

The latest, most advanced phase of the ranking mechanism has the goal to improve the performance by including a wide variety of more sophisticated features. The features undergo a more advanced feature selection method, whereas ensemble classifiers are employed to handle the bigger size of the feature set.

Experiment setup The advanced ranking mechanism follows the setup of the basic ranking mechanism, with the difference that the evaluation is done on a full cross validation with 10 folds. The folds have been particularly adjusted, so that multiple evaluations of the same sentence, which exist on the border between the training part and the test part of the fold, remain on the same part. Additionally, all steps of the feature engineering process are now expanded to the language pair English-German, in order to indicate the applicability of the investigated methods.

4.3.1 Feature clean-up

An extended feature set is considered for this part of the ranking mechanism. Additionally to the features of the basic set, presented in the previous section, it includes all possible generated black-box features from the categories *tree label counts, CFG rules, alignment of tree nodes, language checking* and *IBM-1 model scores* and some additional *count-based and position features*. A detailed list of the features is given in Section 2.3.1.

Some of the above feature categories refer to an open class of features. This means that the amount of features is not pre-defined, but it depends to the annotation of the given text. Additionally, since many of these features refer to aligned tree nodes or CFG rules, the final number of features may be very high. Indeed, for the feature extraction we used a sample with the first 11,114 sentences, containing 55,544 translations. For these sentences, the feature generation process resulted into approximately 154,657 features. This is far beyond the capability of the machine learning methods, given the size of the corpus.

Nevertheless, many of these feature are sparse, since they depend on the appearance of a grammatical phenomenon and a vast majority of them appears in relatively few sentences. Indicatively, among the above features, there are 22,970 aligned tree nodes between English and German. Out of these, only 26 tree node labels are aligned in more than 10,000 sentences, 46 of them in more than 5,000 sentences, whereas 197 tree node labels are aligned in more than 1,000 sentences.

Based on the above, we proceed by choosing only a subset of the entire feature set, by selecting the most dense features from the important categories. For each feature, we vary the density threshold to get a representative sample. In particular, from the open class features we gave priority to classes which are highly relevant to grammaticality, and in particular VPs and NPs:

- CFG features including VPs and derivatives with more than 20,000 occurrences (5 features)
- CFG features including NPs and derivatives with more than 20,000 occurrences (5 features)
- CFG alignment features including VPs and derivatives with more than 10,000 occurrences (5 features)
- CFG alignment features including NPs and derivatives with more than 30,000 occurrences (5 features)
- CFG position features including VPs and derivatives with more than 24,000 occurrences (5 features)

Additionally we included language correction features that have more than 1,000 occurrences (4 features). From the rest of the features, where there is no clear intuition about their contribution on grammaticality etc., the density threshold was set to include the ones that are more dense, albeit limiting the overall number of features to a few hundreds. After some repetitive probing, the threshold was set to chose the ones that have at least 51,000 occurrences. This includes:

- 50 features from absolute QE, as provided by QuEst
- 8 features about the position of unknown words in the sentence
- the Language Model probability
- the IBM1-model score on both directions
- the recall, precision and overall score of contrastive-METEOR, and the score of contrastive BLEU
- count and position statistics for VPs, NPs, NNs, dots, commas
- maximum and average height of an S, a NP or a VP node in a tree
- parsing statistics (log-likelihood, number of trees) from both Berkeley parser and BitPar

This selection process resulted to a set of 139 features. It must be noted that the above intuition-based selection process is done mostly for acquiring an adequate and compact set as a step for advancing the experiments. There is no evidence whatsoever, that the size or the content of the reduced feature set is by any means optimal, as compared to other possible subsets from the original 154,657 feature set.

4.3.2 Adding the advanced features to the basic ranking mechanism

The first experiment with the advanced feature set concerns its application on the exact setup of the basic mechanism. We train two models for each language pair, using Logistic

set	τ	NDCG
basic	0.261	0.730
advanced	0.110	0.680

Confidence interval ($\alpha = 0.05$): $\tau = 0.014$

Table 4.10: *Comparison of the ranking performance of the basic and the advanced feature set by using Logistic Regression, for German-English*

Regression with Stepwise Feature Set Selection: one model is trained with the basic feature set, whereas an advanced model is trained with the entire set of 139 features described in the previous section. The goal is to examine what is the effect of the advanced feature set on the performance of the ranking.

The results of the application of the advanced feature set can be seen in Table 4.10. The ranking performance of the advanced feature set is significantly worse than the basic feature set, using the particular learning method. This may indicate the inability of this learning method to handle this amount of features. Therefore, the next step is to experiment with feature selection methods and other learning algorithms.

4.3.3 Recursive Feature Elimination and Gradient Boosting

Related work and preliminary experiments have shown that reducing the number of features by feature selection can result in better learning capabilities. Since Stepwise Feature Set Selection did not perform adequately for the augmented set, we investigate the possibility of using RFECV, using a SVM with a linear kernel. Additionally to the RFECV, we introduce an advanced learning method: a Gradient Boosting classifier that operates as an ensemble of Decision Trees.

Similar to previous experiments, the feature selection is applied on a sub-set of the training part of the 6th fold of the full dataset. Since RFECV with SVM does not scale well for a big amount of data, we performed stratified sub-sampling to select a smaller amount of sentences for tuning the feature selection. After several tests on various percentages, 2.5% of the original sentences were used for German-English, whereas 5% of the original sentences were used for English-German. The selected feature set was used to train the ranking model on the entire data-set and to evaluate its performance with 10-folded cross-validation.

The optimal feature set for German-English contains 41 features, whereas the English-German one contains 56 features.

The results of using RFECV and Gradient Boosting can be seen in Table 4.11. When it comes to using the advanced feature set, Gradient Boosting achieves significantly better performance than Logistic Regression. RFECV improves significantly the performance of the Logistic Regression on the advanced feature set, but it still does not reach the performance of the same algorithm with the basic set.

lang.	method	set	τ	NDCG
de-en	Logistic Regression	basic	0.261	0.730
		RFECV	0.181	0.716
		full	0.110	0.680
	Gradient Boosting	basic	0.265	0.736
		RFECV	0.276	0.739
		full	0.280	0.742
en-de	Logistic Regression	basic	0.151	0.725
		RFECV	0.020	0.696
		full	0.034	0.703
	Gradient Boosting	basic	0.138	0.723
		RFECV	0.174	0.731
		full	0.170	0.733

Confidence interval ($\alpha = 0.05$) $\tau = 0.014$

Table 4.11: *Comparison of the performance of the full feature set and the result of the RFECV with Logistic Regression and Gradient Boosting in terms of correlation with the human judgments.*

When it comes to Gradient Boosting, there is a negligible difference between using the RFECV and the full feature set. Although the usage of RFECV did not manage to improve the performance, it is interesting to conclude that the number of features (139) was reduced to less than the half, but the performance of the ranking remained the same. Reducing the amount of features can be of interest in an application environment, since it also reduces the requirements for computation and time.

The above observation can also be seen in Figure 4.1, which depicts the progressive increase in the classification quality, as more features are added into the model. It becomes apparent, that the performance reaches already high levels with an amount of about 25 features, whereas after few fluctuations it enters a plateau, where more features do not have any significant implications to the model.

4.3.4 Observations from Recursive Feature Elimination

Although the model of Gradient Boosting cannot be easily interpreted, we can use the results of the RFECV to get some observations about the most important features. This is based on the previous observation, that the models built on RFECV feature sets have little performance difference than the full feature set.

The features that prevailed in the RFECV can be seen in Table 4.12. Although it is not clear what is the exact contribution of each feature, based on the top features after the RFECV we can note some observations:

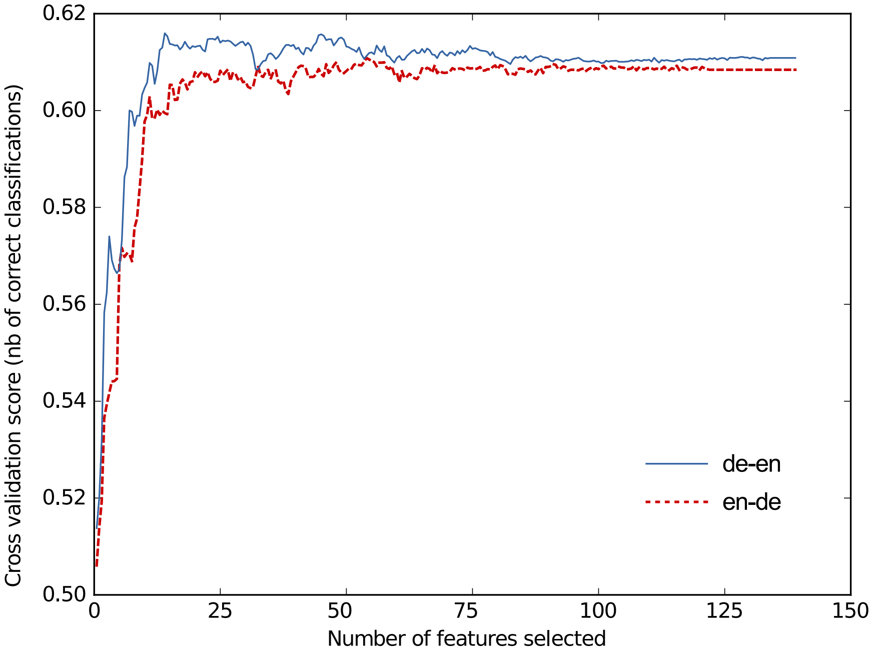

Figure 4.1: *Plot of the Recurcive Feature Elimination with Cross-Validation that was performed in order to find an optimal feature subset*

Comparison between basic and advanced feature sets During the implementation of the preliminary and the basic ranking mechanism, simple features containing source information were ruled out (Sections 4.1.1, 4.2.1 and 4.2.5). However, during the feature selection for the advanced feature set, it became apparent that a few advanced features that do include source information are among the top features. These features are derived from alignment of grammatical structures between the source and the target. For German-English, these are the statistics of the source-target alignment of the simplest CFG rule for a sentence node (S→NP-VP), whereas for English-German it is the count of target noun phrases aligned with source noun phrases. The existence of these particular node alignments among the top ranked ones is considered reasonable, given the grammatical operation of these nodes and their regular occurrence. Additionally, it indicates that although very simple and generic features based on source information may be of little use, targeted features that capture translation adequacy on particular structures can still be of high relevance for comparing two translations. The IBM model 1 features on both directions also appear to be satisfactory source-related features on both language directions.

Finally, it is worth noting that some single features from the basic ranking mechanism have been replaced by a multitude of more specific features with the same or similar functionality. For example, the number of verb phrases in overall has been replaced with the number of verb phrases within specific rules. This can be also attributed to the abilities of the advanced learning method which can make better use of a larger amount of partially overlapping features.

Comparison between language pairs Although the feature elimination process was run separately for each language direction, there is some similarity between the language pairs, with regard to the features that were automatically selected.

However, some noteworthy differences can be seen at the grammatical features. The ones selected for English-German indicate the importance of the position of the verb phrases and the prepositional phrases in the sentence, obviously justified by the positional requirements of the German grammar. This is in contrast to the feature set for German-English, which contains no features referring to the position of verb phrases or prepositional phrases.

Meanwhile, nouns and noun phrases are represented by a relatively big amount of features on both language directions. For the direction into English, some CFG rules have separate features, such as the noun phrases that contain a determiner and a noun, the verb phrases containing an infinitive and the prepositional phrases introduced with the preposition "in".

Comparison between parsers Parsing probabilities for the advanced feature set were produced with both BitPar and Berkeley Parser. During the RFECV, only the ones from BitPar deemed more significant for the model, whereas the ones from Berkeley Parser were eliminated. Nevertheless, Berkeley Parser delivers a more simplified tree, which is more suitable for count and alignment of nodes.

4.4 Summary

In this chapter we presented the feature engineering in three phases. The preliminary phase identified a basic set of features trained with Naïve Bayes, in order to predict rankings that have a significant correlation with human rankings. It was also shown that grammatical features can be more useful than the ones from a Language Model.

In the second phase (basic) a significant improvement over the preliminary mechanism was achieved by adding contrastive scores and other features from absolute QE in a Logistic Regression model. We confirmed that basic source features do not contribute to the translation comparisons and that Logistic Regression handles better this feature set with Stepwise Feature Selection than with L2 Regularisation. Finally we examined the model coefficients to get some intuition about the contribution of each feature, showing that the most dominant features are the number of unknown words, the number of tokens and the contrastive scores, including some grammatical features, such as the number of verb phrases and parse trees.

In the last phase (advanced), a wide variety of grammatical, alignment and position features are introduced in two language directions, whereas an improvement is achieved when trained with a Gradient Boosting classifier. Feature selection with Recursive Feature Elimination has managed to reduce the amount of features to a third of the original set, without any significant drop to the performance. Few features derived from the alignment of grammatical structures between the source and the target are among the most dominant advanced features, whereas only for translating into German, the position of the verb phrases plays a role.

ranking mechanism	prel.	basic	advanced	
language pair	de-en	de-en	de-en	en-de
Parse probabilities				
n-best parses (Berkeley, count)	+	+	+	+
probability (BitPar, best tree)			+	+
probability (BitPar, worst tree)			+	+
probability (Berkeley, summed all trees)	+	+		
probability (BitPar, mean, std of n-best)			+	+
Tree nodes				
nouns (count)				+
nouns (average position)			+	+
nouns (std of positions)			+	
NPs (count)			+	+
NPs (average position)			+	
NPs (std of positions)			+	+
VPs (count)	+	+		
VPs (std of positions)				+
VPs (average and max tree height)			+	+
PPs (count, std of positions)				+
CFG rules				
NP→DT-NN (count)			+	
PP→IN-NP (count)			+	
VP→TO-VP (count)			+	
S→VP (position from the sentence end)				+
VP→VP (position from the sentence end)				+
Aligned CFG rules and nodes				
S→NP-VP (counts, tree depth, position)			+	
NP (counts)				+
Contrastive scores			+	
contrastive BLEU			+	+
contrastive METEOR recall			+	
contrastive METEOR frag. penalty			+	+
contrastive METEOR overall score		+	+	+
IBM model 1				
IBM model 1 scores (both directions)			+	+
avg translations per source word (threshold p>0.2; p>0.01 weighted by the inv. freq)				+

ranking mechanism	prel.	basic	advanced	
language pair	de-en	de-en	de-en	en-de
Target language model				
smoothed 5-gram probability			+	+
smoothed 3-gram probability		+		+
3-gram perplexity				+
unknown words (count)	+	+		+
unknown words (average, first and last position)			+	+
unknown words (std of positions)				+
unknown words (average, first position norm.)			+	
Language correction				
total number of errors			+	+
space after comma or parenthesis			+	+
uppercase sentence start				+
Count-based features				
tokens (count)	+	+		+
tokens (source/target ratio)			+	+
tokens (target/source ratio)				+
type/token ratio		+		+
characters (average per word)			+	
commas (count)		+	+	+
commas (average, std of positions)				+
commas (target-source difference)				+
dots (count)		+		+
dots (average position)			+	
numbers (percentage in target)			+	
numbers (target-source difference normalized)				+
tokens inc. chars other than “a-z” (count, ratio)			+	+

Table 4.12: *The features of the advanced feature set as selected after RFECV for the language pairs German-English and English-German. Additionally the features of the preliminary and the basic feature set.*

Chapter 5

Comparative Quality Estimation for System Combination

In the previous chapters, the focus has been on the performance and the functionality of the method as a full ranking mechanism. Apart from ranking per se, we investigate a possible application: the possibility to use ranking for sentence selection in order to produce a combination on the output of various systems.

The system combination presented here is a combination with relatively low granularity, compared to advanced lattice-based combination methods, since the combination unit is the entire sentence. Nevertheless, as the decisions are taken given a full sentence, there is the possibility to take advantage of informative features that depend on a full syntactic structure, e.g. grammatical features, which could not be extracted from sentence chunks. Secondly, the combination provided here is rather hybrid, in the sense that it combines two different machine translation technologies: Rule-based Machine Translation (RBMT) and Statistical Machine Translation (SMT) and variations of them. A selection method that takes into consideration the full sentence can be therefore beneficial, in order to profit from the long distance transfer rules employed by the RBMT system. Additionally, the two types of systems may be complementary to each other in many other ways.

We hereby apply the ranking mechanism on the output of three distinct systems and we show that it can successfully produce a system combination that performs better than any of its components. This chapter is structured as follows: First, we explain the setting of the experiment and the preparation of the components of the system combination (Section 5.1). In Section 5.2, we measure the performance of the selection mechanism. In Section 5.3 we list 6 different feature sets and examine their performance, the contribution of each feature and the impact they have on the proportion of the sentences chosen from every system. Finally, a comparison of various learning methods is given in Section 5.4.

5.1 Experiment setup

A ranking mechanism is trained on sentence-level ranking labels on the MT outputs of the training set, as described in Section 2.1.1. The trained ranking mechanism is applied later on the output of three diverse systems in order to perform selection. For every sentence, the three alternative translation outputs are ranked by the ranking mechanism and the output with the best rank is fed to the combined output. The three systems will be thereof referred to as *MT components* and the ranking mechanism, as applied for the selection, as a *selection mechanism*. The three exact systems are not seen in the training set of the selection mechanism, although variations or previous versions of them may exist. The setup for each of the three components is presented later (Sections 5.1.1, 5.1.2 and 5.1.3).

In this experiment we test the following hypotheses:

- The selection yields better translations than any individual MT component. To evaluate the performance of the selection mechanism, the combined output, including all the sentences of the document, is scored with an automatic metric against the reference (Section 2.7.2). In order to judge whether the use of the selection mechanism led to an improvement, the scores on the combined output are compared with the respective scores on the individual output of each of the three MT components and with those achieved with a random selection. The hypothesis is confirmed with a two-tailed significance test, based on a bootstrap resampling of 1000 samples (Section 5.2).

- The selection leads into similar performance improvements even if MT outputs from an inferior component are included. For this purpose we use the selection mechanism on two variations: one variation performed selection between all three components (trained-3) and another variation between the two best components (trained-2). Then, we measure whether there is a difference in the results of the selection between the two variations.

- The machine learning setup that yields the best performance is similar to the one of the generic ranking experiment (Section 3). We therefore train and test the selection mechanism with many combinations of feature sets (Section 5.3) and machine learning methods (Section 5.4) and compare their performance. Additionally, we test whether the use of a different parser for the PCFG features may lead to considerable differences to the quality and the setup.

This experiment is based on the ranking method, as presented at the experiments in Chapter 3. As compared to those experiments, the current one has some differences at the setup, particularly concerning the usage of the data:

- The development and test sets originate from the IT domain and the SMT components include IT domain-specific corpora in their training and tuning datasets. This was chosen in order to check whether the selection mechanism can generalize by covering a domain other than the generic ones. The development and test sets originate from the QTLeap corpus[1], a small domain-specific dataset containing user questions to an IT help-desk, translated by professional translators for the purpose of being used in MT evaluation. As this dataset is divided in several non-overlapping batches, each containing 1000 questions, we used the first batch for tuning of the SMT system and the second batch for testing the entire pipeline. For reducing the complexity, we perform the analysis for German-English, assuming that the results for the other language pairs will be similar.

- Instead of ranking and combining many systems, as explained above, we use 3 MT systems that derive from two different machine translation technologies (RBMT and SMT). This way we show that the selection mechanism is capable of a hybrid combination, i.e. able to properly select the positive aspects of both MT types.
- The rank labels that are used for training the ranking mechanism are not produced by humans, but they are instead generated after scoring the MT outputs against the reference translations using automatic metrics. This is done due to the lack of test set where a fixed amount of participating systems is consistently evaluated by humans (the WMT ranking evaluation contains arbitrary rankings over different combinations of systems for every sentence).

However, given the fact that the feature engineering, the machine learning methods and the training material for training the ranking model is the same as the generic ranking experiments, we consider that the modifications included here do not harm the overall consistency, but instead complement the original

5.1.1 SMT component

The SMT component is a phrase-based statistical system with baseline settings as per the Shared Task on Translation on the IT-domain, within the First Conference on Machine Translation (WMT16; Bojar et al., 2016a).

Training was performed on a large amount of parallel corpora of a generic domain including political speeches and news, and a technical domain including text from software interfaces and documentation. The generic domain corpora include Commoncrawl (Smith et al., 2013), Europarl ver. 7 (Koehn, 2005), MultiUN (Eisele and Chen, 2010) and NewsCrawl, as appeared in Bojar et al. (2013a), similar to the setting of the state-of-the-art system (Durrani et al., 2013). The domain-specific corpora originate from the localisation files of the Chromium Browser, the Drupal content management system, the LibreOffice User Interface and the Linux distribution Ubuntu Saucy. Additionally, the terminology provided by the Text Foundation and the parallel localisation text of their website was included as a parallel corpus. The size of the corpora that take part in the training data is shown in Table 5.1.

The translation table was trained on a concatenation of the generic and technical data, filtering out the sentences longer than 80 words. Alignments were produced by Giza++ or

[1] http://www.qtleap.eu

corpus	entries	words
Commoncrawl (parallel)	2.4M	53.6M
Europarl (parallel)	1.9M	50.1M
MultiUN (parallel)	167.6K	5.8M
News Crawl (parallel)	201.3K	5.1M
Europarl (mono)	2.2M	54.0M
News (mono)	89M	1.7B
Chromium browser	6.3K	55.1K
Drupal	4.7K	57.4K
LibreOffice help	46.8K	1.1M
LibreOffice UI	35.6K	143.7K
Ubuntu Saucy	182.9K	1.6M

Table 5.1: *Size of training corpora used for the SMT component*

mGiza with "grow-diag-final-and" symmetrisation and "msd-bidirectional-fe" reordering (Koehn et al., 2003). The tuning was run with MERT (Och, 2003).

The **Language Model** (LM) was trained using the target side of the technical corpora, Europarl and the monolingual News Crawl from the years 2007 to 2013 (Callison-Burch et al., 2007). In order to adapt the Language Model to the particular domain (Schwenk and Koehn, 2008), one separate LM was trained for each corpus and then all LMs were interpolated on the domain-specific tuning set. The localisation and terminology corpora of Text Foundation were not included as they mostly consisted of uni-grams and low ordered n-grams.

Preprocessing followed the state-of-the-art preprocessing techniques for phrase-based SMT. The text was tokenised, truecased (Lita et al., 2003; Koehn et al., 2008) and German compounds were split Koehn et al. (2008) prior to the training and the decoding. The MT output was de-tokenised and de-truecased afterwards. A few regular expressions were added to the tokeniser, so that URLs are not tokenised before being translated. Normalisation of punctuation was also included, mainly in order to fix several issues with variable typography on quotes. The phrase-based SMT system was trained with Moses (Koehn et al., 2007) using EMS (Koehn, 2010), whereas the Language Models were trained with SRILM (Stolcke, 2002) and queried with KenLM (Heafield, 2011).

5.1.2 Rule-based component

Lucy Translator was used as a RBMT system. It is known as a system that is based on rules developed through many years by professional linguists (Alonso and Thurmair, 2003). The system has shown state-of-the-art performance in the past, particularly for German.

The translations used in the current experiments were obtained by using the translation engine of Lucy EX version 6.11 Label 20 Build 79 and the lingware lexicons version LC 7.10.

5.1.3 Serial system combination of Rule-based and Statistical Machine Translation

As an approximation of automatic post-editing of the RBMT system, a serial RBMT→SMT system combination is used, similar to the one described in Simard et al. (2007). The first stage is the translation of the source language part of the original training corpus of the SMT component above by the RBMT system. The second stage consists of training a phrase-based SMT system with the RBMT output as the source-side of the parallel training data and the target language of the original corpus as its target side. Later, the test set is first translated by the transfer-based system and then the obtained translation is translated by the SMT system.

source:	In the Insert menu, select Table
SMT:	In Menü "Einfügen", Tabelle auswählen.
RBMT:	Wählen Sie im Einsatz-Menü Tabelle aus.
RBMT→SMT:	Wählen Sie im Einfügen Menü Tabelle aus.
reference:	Wählen Sie im Einfügen Menü die Tabelle aus.

Figure 5.1: *Example of the operation of the serial combination of RBMT and SMT as compared to its components.*

In previous experiments involving German, this method could outperform Moses trained on a large parallel corpus (Avramidis et al., 2015b). The example in Figure 5.1 illustrates how the statistical post-editing operates. While the original SMT output used the right terminology ("Menü Einfügen" – "insert menu"), the instruction includes an imperative in an informal way, which can be considered impolite. In contrast, the output of the transfer-based system employs the rule of forming an imperative in the polite way, yet mistranslating the menu type. The serial combination RBMT→SMT produces a correct translation.

5.1.4 Selection mechanism

The training of the selection mechanism is based on the MT outputs of all systems that participated in the evaluation of the WMT shared tasks for German-English (years 2008-2014), as described in Section 3.1.2. As mentioned earlier, the development and test sets originate from the QTLeap corpus.

The ranking labels used for training this experiment are produced by scoring competitive MT outputs against the reference. We experiment on several metrics for constructing the ranking labels, in particular smoothed BLEU, METEOR and rgbF. We choose the label type which maximizes if possible all automatic scores on our development set, including document-level BLEU, METEOR, rgbF and TER.

We test all suggested feature sets with various ML methods. The binary classifiers are wrapped into rankers using the "soft pairwise recomposition" (Section 2.4.2.2) to avoid ties between the systems.

5.2 Performance of the selection mechanism

The optimal model is based on a feature set that includes target-language features derived from the Language Model, the IBM model 1, PCFG parsing and CFG rules (Section 5.3) and trained with a Gradient Boosting classifier (Section 5.4). The optimal ranking labels are produced with rgbF 2.7.2, measured against the reference translations.

The automatic document-level scoring of the combination produced by the sentence-level selection mechanism and its three components, based on BLEU, METEOR, TER and rgbF is given in Table 5.2. SMT is the component with the highest scores for all three measures, RBMT comes second (with the exception of TER where it has a non-significant difference with RBMT→SMT), whereas RBMT→SMT has the lowest scores.

system	BLEU	METEOR	TER	rgbF
SMT	43.1	40.4	32.2	48.6
RBMT	40.8	38.7	36.4	46.5
RBMT→SMT	39.7	38.3	36.1	46.0
random selection	39.5	38.6	34.5	45.5
trained-3	‡ 44.9	40.6	† 31.5	‡ 49.6
trained-2	‡ 44.6	‡ 40.8	† 31.6	‡ 49.8

statistically significant comparison with SMT:
‡: $a < 0.05$
†: $a < 0.10$

Table 5.2: *Performance of the selection mechanism compared to its three components, as measured by four automatic reference-based evaluation metrics.*

The selection mechanism when applied on all three systems (trained-3) achieves an improvement of 1.8 points BLEU, 0.2 points METEOR, 0.5 points TER and 1.0 point rgbF. The improvement is statistically significant for the BLEU, TER and rgbF scores, when compared to all three components and the random selection.

The performance of the selection mechanism when applied on the two best system outputs (trained-2) is similar with the one applied on all three systems (trained-3) and there is no significant difference between the two settings for most of the metrics (with the exception of METEOR which is slightly higher for trained-2). This indicates that the selection mechanism leads into performance improvements even with different participating components and is not negatively affected by a component whose overall output is inferior than the one by the others.

5.3 Feature sets

This section goes into details about how the best feature set is chosen, after testing the performance of the selection mechanism with various feature sets. Since the setting for system combination has some differences to the one of the ranking mechanism shown in the previous chapters, we examine the performance of the selection mechanism after retraining it with a small variety of feature sets, similar to the ones seen before (Chapter 4).

5.3.1 Definition of feature sets

We experimented with several feature sets that performed well in previous experiments, including some modifications. In order to make the experiment feasible, not all feature sets presented in Chapter 4 were used for the experimentation. The feature sets used are detailed in Table 5.3 whereas a small description is following:

Feature set 1a is a the first version of the *basic feature set* from Chapter 4, as described in Section 4.2.2. This set was presented in Avramidis (2012a), as an extension of a feature set which has performed well as a metric in WMT11 metrics task (Avramidis et al., 2011). It includes the number of tokens, count of unknown tokens (unk), count of alternative parse trees, count of verb phrases, the PCFG parse probability given by Berkeley Parser and a contrastive METEOR.

Feature set 1b augments Feature set 1a, by replacing Berkeley Parser with BitPar and including two additional parsing features over the parser's n-best list: the standard deviation of the probabilities in the parsing n-best list and a binary feature on whether the probability of the best tree is higher than the standard deviation over the mean of the probabilities in the n-best list.

Feature set 2a is the augmented version of the *basic feature set* from Chapter 4 (Section 4.2.3) that combines the feature set 1a with the baseline feature set of WMT (Specia et al., 2013; Bojar et al., 2013a), although most of the source-based features got assigned a zero coefficient (Section 4.2.5). This feature set got the best result in the Quality Estimation task of the Eight Workshop for Statistical Machine Translation (WMT13) for ranking in German-English (Avramidis and Popović, 2013, as shown in Section 3.4).

Feature set 2b is similar to feature set 2a, but the Berkeley Parser probability is replaced with the probability produced with BitPar.

Feature set 3a is a subset of the *advanced feature set* from Chapter 4 (Section 4.3). It does not include the contrastive METEOR score but it introduces 18 more features. In particular, it includes the n-gram probability of order 5, two features for the IBM model 1 sentence scores in both directions and two features from language correction rules. The PCFG parse feature subset was augmented with the number of unaligned nodes between source and target, the overall tree depth, the depth of the tree and the depth of first (min) and last (max) sentence nodes. Finally, it includes the position of several verb-related CFG rules (e.g. S→VP, VP→VP etc.) measured from the beginning and the end of the sentence.

feature	feature set	1a	1b	2a	2b	3a	3b
category	feature name						
counts	total tokens	+	+	+	+	+	+
	unk	+	+	+	+	+	+
	unique/total tokens			+	+	+	+
	commas, dots			+	+	+	+
LM	LM 5gram prob					+	+
	LM 3gram prob			+	+		
	high freq source uni-grams			(+)	(+)		
	low/high freq source bi-grams			(+)	(+)		
	low/high freq source tri-grams			(+)	(+)		
	percentage of known source 1grams			(+)	(+)		
	contrastive METEOR	+	+	+	+	+	
IBM-1	score, inverted score					+	+
	translations per src token (p>0.2)			(+)	(+)		
	translations per src token (p>0.01 weighted by the inv. freq of src token)			(+)	(+)		
PCFG	n-best trees	+	+	+	+	+	+
	Berkeley prob	+		+		+	+
	BitPar prob		+		+		
	standard deviation of n-best prob		+				
	low prob as compared to n-best		+				
	VPs	+	+	+	+	+	+
	unaligned src/target nodes					+	+
	tree depth					+	+
	min depth of S					+	+
	max depth of S					(+)	(+)
CFG position (sentence start)	S→VP					+	+
	VP→VP					+	+
	VP→VVINF					(+)	(+)
	VP→VB					+	+
	VP→VBZ					+	+
	VP→VBG					+	+
CFG position (sentence end)	VP→VZ					(+)	(+)
	VP→VP					+	+
	VP→VVINF					(+)	(+)
	S→VVPP					(+)	(+)
Lang. correction	unpaired brackets					(+)	(+)
	compound suggestions					(+)	(+)

(+): Feature was assigned a zero coefficient during the machine learning process

Table 5.3: *List of the feature sets investigated for the system combination*

Feature set 3b is a variation of feature set 3a by removing the contrastive METEOR features. It was developed in an effort to re-evaluate the contribution of the contrastive METEOR score as a feature for sentence-level system combination. It is noteworthy that this feature bounds the Quality Estimation of every single output to the competitive outputs. Since the feature was proven useful given a context of more than five competitive outputs, in the current experiment which provides only two competitive outputs its contribution needs to be re-evaluated.

5.3.2 System combination performance per feature set

In this experiment, we train several systems with the feature sets explained above, using various learning methods. Then we compare their performance in combining the three MT systems. The automatic scores on the results of the system combination are shown in Table 5.4.

The best score is achieved by the model trained on the feature set 3b using Gradient Boosting. The system combination by this model achieves better performance according to all three automatic metrics measured here. The use of the same machine learning method and a very similar feature set also shows the highest performance in the generic ranking experiments (Section 3.6). Similar to the observations there, the simple feature sets were optimally trained with linear methods, whereas the more complicated feature sets required ensemble classifiers, indicating that when additional features are included, the problem has non-linear characteristics. The observed similarity of machine learning setup (features, algorithm) between this chapter's experiment and the generic ranking experiment, indicates that the method is not easily affected by changes in the domain or the ranking labels.

Based on BLEU and rgbF, most feature variations (with the exception feature set 3a) achieve an improvement over the best MT component, but the difference with METEOR is tighter, where only feature set 3b yields a small improvement.

feature set	BLEU	METEOR	rgbF	algorithm
1a	44.5	40.4	49.6	Logistic Regression (SFSS)
1b	44.1	40.3	49.5	Logistic Regression (L2)
2a	43.7	40.2	48.9	Logistic Regression (L2)
2b	43.8	40.2	49.2	LDA
3a	42.4	39.9	48.2	AdaBoost
3b	44.9	40.6	49.6	Gradient Boosting
SMT	43.1	40.4	48.6	

Table 5.4: *Comparison of the 6 most successful feature sets, all trained over the ranking labels produced with rgbF. The last column indicates the learning method that gave the best performance for the given feature set.*

The only difference as compared to previous experiments refers to the feature of Contrastive METEOR which has a negative contribution to the feature set 3a, as it can be observed after comparing the performance of features 3a and 3b. We should recall that this feature appears to be useful for the ranking experiment (Section 4.3). There, a larger amount of MT outputs (5 to 12) are ranked for every sentence and therefore contrastive METEOR is computed based on a wider basis of competitive translations which act as pseudo-references. On the contrary, the inclusion of the feature in this feature set leads to a significant drop on the scores, potentially indicating that the calculation of contrastive METEOR on two out of the these three particular systems is not indicative of the quality of the third output.

Finally, there does not seem to be any significant difference in the contribution of the parsing features, if another parser is used to produce these features. This can be seen through the comparison of feature sets 1b and 2b (produced with BitPar) with feature sets 1a and 2a respectively (produced with Berkeley Parser), where there is only a minor difference in the scores.

5.3.3 Contribution of features

The contribution of features to the binary selection model can be seen by examining the beta coefficients of the best Logistic Regression model (Table 5.5).

The Language Model is the feature contributing the most to the sentence selection. This can be contrasted to the observations for the generic ranking model (Section 4.2.5), where the Language Model has a negative or no contribution. As explained above, this can be justified by that fact that these are two different problems applied on different data.

The rest of the feature coefficients are relatively similar to generic ranking experiments, with the contribution of the PCFG parse probability also having a bigger contribution. Moreover, the features of the positions of verb-related CFG rules are also remarkable. It appears that the models favours positioning subordinate verb phrases closed to the end of the sentence (as seen by VP→VP, VP→VB, VP→VP-end) with the exception of the verb phrases that contain a verb gerund (VBG) that should be closer to the beginning. These observations are justified, as they generally follow the positional requirements of the German language.

5.3.4 System proportions per feature set

Here we analyse the proportion of sentences that the selection mechanism chose from each MT component. The proportions of the best combination and their graph are shown in Table 5.6. SMT provides almost half of the sentences, namely 49.1%. The rule-based system 22.8% of them, whereas the RBMT→SMT contributes with a 23.5%.

Meanwhile, it is noteworthy that different feature sets favour different system proportions. The proportions per feature set can be seen in Figure 5.2. Apart from the feature set 3b and 1a which got the best scores, having the biggest percentage originating from SMT, the

feature	beta
LM 5gram prob	2.01
total tokens	1.19
Berkeley prob	0.54
IBM1-score inverted	0.50
unk	-0.47
max depth of S	-0.47
VPs	0.44
unique/total tokens	-0.41
position VP→VP	0.34
commas	-0.31
position VP→VB	0.24
unaligned src/target nodes	-0.23
dots	0.22
tree depth	0.20
position VP→VP (end)	-0.11
position S→VP	0.11
IBM1-score	0.10
position VP→VBG	-0.07
n-best trees	0.01
position VP-VBZ	-2×10^{-03}
min depth of S	0.00
position VP→VZ (end)	0.00
position VP→VVINF (end)	0.00
position VP→VVINF	0.00
position S→VVPP (end)	0.00
unpaired brackets	0.00
compound suggestions	0.00

Table 5.5: *Beta coefficients of the best feature set as learned with Logistic Regression*

rule-based system is the one with the most selected sentences for 4 out of the 6 feature sets. The RBMT→SMT had the highest proportion only for one feature set which had very low scores.

We may attribute the difference of the proportions between feature set 3b and the rest of the feature sets to some of the most outstanding individual features they consist of. In particular, grammatical features, used as the main fluency indicators for feature sets 1a, 1b, 2a and 2b, are very related to the translation methodology of the rule-based system. On the contrary, feature set 3b adds three more features, a Language Model score and two features from word alignment, that are similar to the respective components of a SMT system and which may explain why it favours SMT more.

MT component	proportion
SMT	49.1%
RBMT	22.8%
RBMT→SMT	28.1%

Table 5.6: *Proportion of each MT component as selected by the selection mechanism*

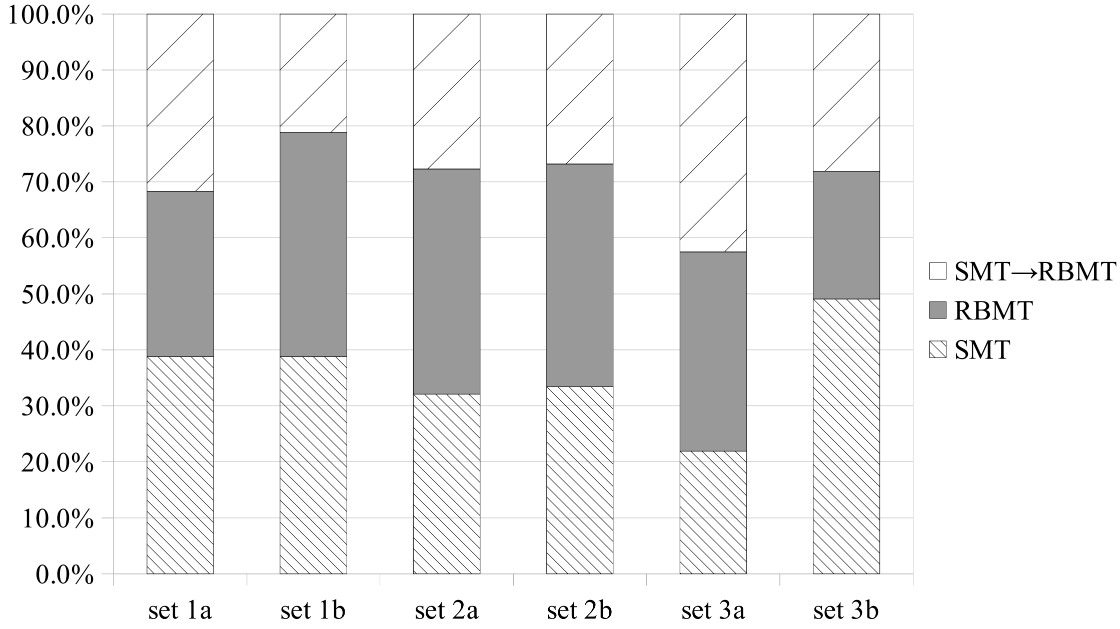

Figure 5.2: *Sentence proportions from each component per feature set*

5.4 Machine Learning

We train a ranking mechanism with all feature sets detailed above, by employing many machine learning methods, including a Bagging Classifier, Decision Trees, ExtRa Trees, Gaussian Naïve Bayes, k-neighbors, SVMs with a linear kernel, Logistic Regression, LDA, Quadratic Discriminant Analysis, Random Forests. Additionally, AdaBoost and Gradient Boosting, two classifiers ensembled over Decision Trees (Section 2.5) are used.

The automatic scores on the sentence selection, using these classification methods are reported in Table 5.7. For every classification method, it is indicated which feature set and which training label performed best.

The best performance is given by Gradient Boosting, followed by Logistic Regression and AdaBoost. In the fourth position comes LDA. It is known that LDA seems to be more appropriate method than Logistic Regression when the covariates follow the normal distribution, but their performance gets very close when the sample size is large (Pohar et al., 2004). This is very similar to our case, for the feature set 1a.

algorithm	BLEU	METEOR	rgbF	feat.set	label
Gradient Boosting	44.9	40.6	49.6	3b	rgbF
Logistic Regression	44.5	40.4	49.6	1a	rgbF
AdaBoost	44.5	40.5	49.3	3b	rgbF
LDA	44.0	40.2	49.3	1a	rgbF
Gaussian Naïve Bayes	43.9	40.4	49.0	1b	rgbF
Bagging	43.4	39.7	48.4	3b	rgbF
Linear SVM	43.1	40.4	48.6	3a	meteor
Quadratic Discr. Analysis	43.0	39.8	48.4	2a	rgbF
Random Forest	43.0	39.5	48.3	3b	rgbF
Decision Tree	42.7	39.6	47.9	1b	rgbF
ExtRa Trees	42.7	39.5	47.9	3b	rgbF
kNN	42.2	39.3	47.4	3b	rgbF

Table 5.7: *Comparison of the performance by various machine learning methods on sentence selection for system combination. For each learning method, we give the automatic scores against the reference translations, the proportion of the 3 system components and the best performing feature set and training label*

Quite a few methods are not able to produce a system combination better than the best component. SVM with a linear kernel, Quadratic Discriminant Analysis, Random Forest, Decision Tree, ExtRa Trees and kNN achieve scores that are equal or lower than the SMT component.

5.5 Summary

In this Chapter the ranking mechanism was applied for system combination. The available MT outputs for every sentence of the test set were ranked with the ranking mechanism and the MT output with the highest rank was selected for the output of the system combination. The system combination was successful in producing a combined output that has higher automatic metric scores than any of its components. Several configurations of feature sets and machine learning methods were tested and the best results were given with an augmented syntax-aware feature set, trained with Gradient Boosting, whereas the training ranking labels were produced with rgbF score measured against the reference translations.

Chapter 6

Translation errors and decoding events

This Chapter investigates situations in the decoding process of phrase-based SMT that can be associated with particular errors on the output of the translation. A set of translations post-edited by professional translators is used to automatically identify errors based on edit distance. Binary classifiers predicting the existence of an error in the sentence are fitted with Logistic Regression, using features from the decoding search graph. Models are fitted for two common error types and three language pairs, in both directions. The statistically significant coefficients of the logistic function are used to indicate parts of the decoding process that are related to the particular errors. Observations related to the coefficients are reported and assumptions for the underlying phenomena are given.

An overview of the related work is given in the Introduction of the dissertation (Section 1.2.6), whereas the contents of the Chapter are structured as follows: In Section 6.1 we provide an overview of the methods employed, namely the statistical modelling of the errors and the automatic annotation of the translation errors. Some more details about the dataset and the training of the models are given in Section 6.2, whereas a technical description about the experiment are given in Section 6.3. The results are shown in Section 6.4, namely the performance of the models and the coefficients estimated for reordering errors (Section 6.4.2.1) and missing words (Section 6.4.2.2). The chapter ends with some discussion about the efficiency of the approach, directions for further work (Section 6.5) and a summary (Section 6.6).

6.1 Methods

6.1.1 Statistical modelling of errors

This experiment uses ML in order to fit a statistical model, associating properties and events of the decoding process with the existence of particular errors of a phrase-based statistical

MT system. Such a model optimizes a function f:

$$Y = f(X) = \beta \circ X \tag{6.1}$$

where:

- X is the independent variable, i.e. a feature vector representing properties and events of the decoding process and has been extracted from the decoding search graph
- Y is the dependent variable, i.e a value predicting the existence of errors
- β is a weight vector estimated by ML to minimize the error of the function $\circ$ given samples of X and Y. This vector contains coefficients for each one of the features. Given a well-fitted model and a relevant statistical function, these coefficients can indicate the importance of each feature.

Our aim is to use the β coefficients in order to explain the behavior of the decoding process, with regard to the errors. The exact formulation of the logistic function $\circ$ is given in Section 2.5.3. The optimal amount of features is selected with Stepwise Feature Set Selection (Section 2.6).

In order to assess the contribution of individual predictors in a given model, we examine their significance by calculating a p-value for each of them. This represents the probability for the null hypothesis, that the beta coefficient of the feature differs from zero. The significance of each coefficient is computed based on the Wald test, following the χ^2 distribution. The Wald test uses the ratio of the square of the regression coefficient to the square of the standard error of the coefficient (Harrell, 2001; Menard, 2002). When the p-value of the Wald test is below the confidence level, the null hypothesis is rejected and the tested coefficient is considered statistically significant.

6.1.2 Automatic annotation of translation errors

Our intention is to not train the model using a complex quality metric such as BLEU or WER as a dependent variable, since this would increase complexity by capturing many issues in just one number. Instead, we choose a more fine-grained approach, by focusing onto specific type of errors that occur often in machine translation output.

The existence of an error on the MT output is used as a training label for the supervised ML algorithm. The types of the errors are identified as a subset of commonly observed error categories (Vilar et al., 2006). In order to detect the errors on the translation output, we follow the automatic error detection method by Popović and Ney (2011), which has shown to correlate well with human error annotation. This method automatically detects errors based on the edit distance of the produced translation against a reference human translation. An example of how errors are detected can be seen in Figure 6.1. Although this previous work defines 5 error types, for our purpose we focus on *missing words* and *re-ordering errors*, since our data give sufficient amounts for training a statistical model on these error categories.

source:	Überraschenderweise zeigte sich, dass die neuen Räte in Bezug auf diese neuen Begriffe etwas im Dunkeln tappen.
translation:	Surprisingly, showed that the new councils in relation to these new concepts slightly in the dark.
post-editing:	Surprisingly, [miss:it] [lex:seems] that the new [lex:councillors] [miss:are] [reorder: slightly in the dark] in relation to these new concepts.

Figure 6.1: *Example of the results found with the automatic error detection process. One can see 2 missing words and a wrong reordering of a batch of 4 words (plus some lexical errors, which are not discussed in this work)*

6.2 Data and models

The model features are extracted from the decoding process (glass-box features) in order to represent various stages of the decoding algorithm of the translation system. This extraction (detailed in Section 2.3.2) results into one feature vector for every sentence. Then, every sentence is labeled with a binary value per error category. This label indicates that one or more errors from this particular error category occur in the sentence. For example, if at least one reordering error occurs in the sentence, the label for the re-ordering error category is set to 1, whereas if no re-ordering error occurs, the label is set to 0. For every sentence, the feature vector and the binary label form a training instance for the prediction model of the respective error category.

For the purpose of our analysis we train one Logistic Regression model per error category (2 categories) and language direction (6 language directions), practically resulting in 12 models. Additionally we train 2 more models (one model per category) with data from all the language pairs in order to reach conclusions for error categories whose occurrence is common for all language directions. Some of these conclusions would not be easily drawn from language-specific models due to data sparsity.

The experiment is based on data from WMT11 (Callison-Burch et al., 2011), augmented with a small amount of data of WMT10 (Callison-Burch et al., 2010) and technical documentation of mechanical engineering equipment, as provided by a translation agency. It focuses on the translation directions from German (de) to English (en), French (fr)) and Spanish (es), and backwards. The number of sentences originating from each data source per language direction is shown in Table 6.1. The sentences are given to professional translators, with the instructions to perform as few changes as possible in order to correct the translations. A major part of the dataset originating from non-confidential (i.e. non-customer) sources is freely available in the TaraXŰ corpus (Avramidis et al., 2014a). The size of the corpus and the number of post-edited (p.e) sentences can be seen in Table 6.2.

Minimal post-editing is considered to be ideal for automatic error detection. In contrast, reference translations may contain severe alterations to the structure of the sentence, misleading the automatic error detection. Nevertheless, as it can be seen in Table 6.2, the num-

	de-en	de-fr	de-es	en-de	fr-de	es-de
WMT10	118	30	33	14	74	0
WMT11	952	952	80	1087	977	101
customer	741	0	430	0	0	830
total	1811	982	543	1101	1051	931

Table 6.1: *Number of sentences from various sources per language pair*

lang	sentences		reordering err.		missing words	
	total	p.e	total	p.e	total	p.e
de-en	1811	1139	1043	474	1079	570
en-de	1101	315	891	232	671	151
de-fr	982	198	819	157	597	80
fr-de	1051	122	851	88	691	76
de-es	543	543	288	288	322	322
es-de	931	931	345	345	333	333
all	6419	3248	4237	1584	3693	1532

Table 6.2: *The size of the corpus per error category and language pair. p.e. indicates the number of sentences that were minimally post-edited by professional translators*

ber of solely post-edited sentences for certain error types and language directions is too small for machine learning, leading to a severely sparse set of training data and therefore weak models.

Consequently, as it was not technically possible to acquire post-editions on a larger amount of data, we perform error-detection on a mixture of post-editions and reference translations, with the hope that this increases the quality of the statistical models. Preliminary experiments confirmed the positive effect, as the precision and recall was increased on most models (even up to 29%) when adding errors detected against reference translations to the ones detected against post-edited output. Despite some obvious drawbacks, this move is also motivated by the fact that the original experiments that showed the accuracy of the automatic error detection (Popović, 2011b) were also performed against reference translations. In order to keep the focus on the main scientific goals of the chapter we won't present these preliminary experiments along with the main experiments.

6.3 Experiment setup

As a statistical phrase-based system we train one Moses (Koehn et al., 2006) system per language direction, using Europarl (Koehn, 2005) and News Commentary corpora. The News Commentary was used only for German-English, English-German due to lack of parallel text for the other language pairs. The settings of the systems follow the baseline of the Shared Task of WMT11 (Callison-Burch et al., 2011), including a truecaser (Wang et al., 2006) for all language pairs and a compound splitter (Koehn and Knight, 2003) for the models translating from German. The system was tuned with MERT (Och, 2003) on the news test set from WMT07 (Callison-Burch et al., 2007). The decoding features are extracted from Moses' verbose output of level 2. and the target Language Model has an order of 5. The translation errors on the MT output are automatically annotated with Hjerson (Popović, 2011c).

6.4 Results

6.4.1 Model performance

A necessary step is to check how well each model fits the data, since a well-fitted model is required for drawing conclusions. For this purpose we perform cross-fold validation with 10 folds. The precision and recall scores are shown in Table 6.3. Precision indicates the ratio of the predicted sentences that contain an error, whereas recall indicates the ratio of the sentences that have an error and its existence is successfully predicted.

	all		de-en		de-fr		de-es		en-de		fr-de		es-de	
	prec	rec	prec	rec	prec	rec	prec	rec	prec	rec	prec	rec	prec	rec
reord.	.70	.79	.83	.81	.88	.96	.85	.82	.85	.93	.87	.92	.77	.70
miss.	.85	.88	.75	.76	.67	.79	.80	.82	.67	.78	.70	.86	.64	.52

Table 6.3: *Precision (prec) and recall (rec) of the logistic regression model, measured with cross-validation. There is one model per combination of language pair and error category; plus one model trained on data from all language pairs per error category. High precision and recall indicate that the model is well-fit.*

The model predicting the existence of reordering errors has the highest precision and recall on all individual language pairs and achieves a generally high precision of about 83-87% (apart from Spanish-German). On the contrary, the model of predicting missing words seems most successful on the dataset combining all language pairs, with a precision of 85% and a recall of 88%. This may be an indication that reordering errors are affected more than missing words by phenomena which are specific to the language direction.

6.4.2 Analysing coefficients

We proceed to the analysis by considering the beta coefficients of the fitted logistic function (function 2.15) for each feature. The feature coefficients given by the fitted logistic function vary per error category and language pair/direction. This is understandable, given the fact that the translation systems and the test sentences are relatively heterogeneous among language pairs and direction.

The interpretation of the feature coefficients may also vary. The most clear indication is the positive or negative sign of each coefficient. Additionally, one has to bear in mind that several features result as a mathematical function or as a byproduct of other features; thus, when they all occur in a logistic function, explaining the contribution of one of them should often consider the existence of other that are related to it, a task which is not always straightforward. Finally, numerical values by themselves cannot clearly indicate whether an observed tendency is the cause of the error or the symptom of another underlying issue. Knowledge or investigation of the decoding process may therefore be useful.

In order to lead to useful conclusions, we show the feature coefficients that seem to have a statistically significant relation to known functionality of the decoding process for several language pairs in both directions. Conclusions are detailed per error category, based on Tables 6.4 and 6.5, which include the most significant and indicative beta coefficients. Coefficients not appearing in specific cells of the table are equal to zero, have a non-significant value, or have been eliminated by the Stepwise Feature Set Selection. For a few language pairs we also include coefficients with p-values for a more loose confidence level (up to $\alpha = 0.30$), when the same coefficient is statistically significant for other language directions.

6.4.2.1 Reordering errors

The coefficients for the reordering errors are shown in Table 6.4. First, the **sentence-level ratio of unknown words** (unk-per-tokens) and the standard deviation of the **position of an unknown token** in the translated sentence (unk-pos-std) have a positive effect towards the creation of a reordering error. Unknown words may interrupt the reordering blocks that are used by the reordering process. A high standard deviation suggests that unknown tokens are scattered in distant places along the sentence; being "unknown" they cannot be captured by the lexical reordering model and it is therefore likely that they cause an erroneous phrase order in the parts of the sentence where they occur. These observations are confirmed for the multilingual model with a zero p-value and additionally for three of the individual language-specific models.

More reordering errors tend to occur when **the source sentence is longer** (total-source-words), since longer sentences tend to contain more complicated grammatical phenomena which require reordering (Birch et al., 2009) particularly when German is involved. Besides, it is known that the complexity of the beam search increases with the sentence sentence length (Koehn et al., 2003; Zens et al., 2004).

The **length of phrase segments** chosen during the decoding has some correlation with reordering errors. In particular there are more chances for reordering errors to occur when:

- the target phrases are longer in average (tgt-avg-phrase-len),
- the source phrases are shorter (in particular when the length range between the shortest and the longest source phrase is lower; src-phrase-len-min/max) and

	all	de-en	de-fr	de-es	en-de	fr-de	es-de
unk-per-tokens	2.08					5.98	3.03
unk-pos-std	0.19	0.22			0.17	0.25	
total-source-words	0.23						0.37
src-phrase-len-avg				-3.51			
src-phrase-len-max					-0.50	-0.36	-1.45
src-phrase-len-min	-0.78						
src-phrase-len-std					3.08	1.44	3.43
src-phrase-pos-min					-0.44		
src-tgt-diff-avg		-2.53				2.03	
src-tgt-diff-max	0.27	0.14	0.24		0.83	0.47	
src-tgt-diff-min	-0.26			-0.60	-0.50	-0.84	
src-tgt-diff-std					-1.66	-2.39	
tgt-avg-phrase-len	1.03				0.70		1.57
best-trans-total	0.01				0.01		
distortion	-0.05	-0.06		-0.21	-0.09	-0.06	-0.15
mono-backward	-0.08	-0.16		-0.25			-0.13
swap-backward	-0.19			-0.48		-0.21	-0.30
other-backward	-0.13	-0.17		0.20			-0.54
other-forward	-0.08	-0.14			-0.11		-0.38
mono-forward						0.22	
LM 5gram	-0.01		-0.02	-0.03			
phrase prob.	-0.04					-0.11	
lexical weighting						-0.04	
inv. phrase prob.	0.03	0.03					
inv. lex. weighting							
phrase penalty				-0.48			-0.59
time-calculate-lm		6.60					
time-other-hyp-score-ratio			-4.72				
time-other-hyp-score				23.47			
time-collect-opt-ratio					-4.63		

Table 6.4: *Indicative beta coefficients for the features affecting the existence of reordering errors.*

- the target phrases are longer than the source phrases (indicated by the minimum and the maximum difference between target and source phrase length; src-tgt-diff-min/max).

This observation refers to the cases when the decoder chooses to translate a source phrase with a relatively longer target phrase. It is noteworthy that this is mostly confirmed for the language directions into German (src-tgt-diff-*), whereas there is the opposite observation for translating from German: the prediction model for German-English has none of these features activated, apart from a relevant feature with a *negative* coefficient (src-tgt-diff-avg). One possible assumption would attribute this observation to the known length difference between German and English (Gries and Wulff, 2012), which is often relevant to constructions that require long-distance reordering. For example, the German present perfect tense which translates the English past simple typically requires one more token for every verb on the German side.

Observations concerning the scoring of the best translation by the inherent probabilistic models of the decoding process include:

- The **scores of the distortion and the lexical reordering model** (backward/forward) have a negative correlation to the probability of re-ordering errors, as it is expected (for the combined model and 5 different language directions)
- For **German-French** and **German-Spanish**, the **LM score** has a negative correlation with the probability of re-ordering errors, confirming that for these target languages the Language Model can successfully indicate a proper word order.

Finally, concerning the distribution of the decoding time:

- For **German-English**, reordering errors occur when the **calculation of the Language Model** takes longer time (time-calculate-lm). This is particularly detectable in this language direction, because the English Language Model is significantly larger than the other Language Models of the other target languages.
- For **German-French** and **German-Spanish** there are more reordering errors when the decoder spends time to calculate the score of hypotheses other than the one selected (time-other-hyp-score), whereas
- for **English-German** there are more reordering errors when the decoder spends proportionally more time to collect options (time-collect-opt-ratio). Both latter observations refer to cases where the search space expands a lot, possibly due to a long and complex source sentence bearing a lot of similarly uncertain translation alternatives.

6.4.2.2 Missing words

The beta coefficients of the model predicting the existence of missing words are shown in Table 6.5.

	all	de-en	de-fr	de-es	en-de	fr-de	es-de
pC-avg	-1.14			-2.21	-5.10	-5.18	
pC-low			-0.31	-0.61	-0.12		
pC-low-avg	-0.67					-5.15	-4.93
pC-min-pos	0.40	0.57		1.06			1.15
pC-std	2.11	1.14	3.82	9.46			7.05
c-avg	0.52	0.66			1.44	2.66	2.14
c-low		0.11	0.19	0.28			
c-low-avg	0.94						
c-std	1.62					8.43	
alt-c-low	0.01						
total-source-words	-0.10				-0.10		-0.29
src-phrase-len-avg	0.70	1.23	1.56				1.74
src-phrase-len-min	-0.48	-0.88	-0.86			-1.10	-0.95
src-tgt-diff-std	0.41	0.87	0.91			1.06	0.57
unk	-1.60	-0.18		-2.81			
unk-pos-last	1.22		3.97				
time-calculate-lm	0.98	2.58					
time-search	0.18		0.77		1.30	0.66	
time-translation		2.98					0.24
total-transopt	0.01	0.01			0.01		
total-hypotheses	0.01			0.01			
hyp-recombined	0.01	0.01	0.01	0.01	0.01		0.01
lexical weighting	0.03	0.02		0.15	0.01	0.03	
inv. lex. weighting	0.01		0.02			0.01	

Table 6.5: *Indicative beta coefficients for the features related to missing words.*

The **probability scores** for the phrases of the best translation have some correlation with missing words. In particular it is more likely for words to be missing when:

- the phrase probabilities are lower in average (pC-avg) and the future-cost estimates are higher in average (c-avg),
- the translated sentence contains less phrases with significantly lower phrase probability (pC-low and pC-low-avg) and more phrases with significantly lower future cost estimate (c-low and c-low-avg),
- the phrase with the lowest probability appears later in the sentence (pC-min-pos),
- there is a high standard deviation of probability (pC-std) and future cost estimate (c-std) among the phrases.
- there is a higher number of alternative hypotheses with a low future cost estimate, i.e. significantly lower than the mean of all hypotheses (alt-c-low).

Concerning the scores that form the overall probability, there are higher chances that words are missing, when the inverse phrase translation probability $\phi(f|e)$ of the best translation is lower and its direct and inverse lexical weighting $lex(e|f)$ and $lex(f|e)$ are higher.

The **sentence and phrase length** also have some effect to the existence of reordering errors. Contrary to the reordering errors, the longer the source sentence is, the less possible it is for missing words to occur. There may be also words missing when the source sentences phrases are longer, namely when the average source length (src-phrase-len-avg) is higher and the length of the shortest source phrase (src-phrase-len-min) is lower. A positive impact to having words missing is also given by the standard deviation of the length difference between respective source and target phrases (src-tgt-diff-std), i.e. when the length difference between aligned source and target phrases varies a lot.

The number of **unknown words** in the sentence has a negative correlation to the probability of having missing words (unk). Moreover, the later an unknown word occurs in the sentence, the more likely it is for words to be missing (unk-pos-last).

Finally, it is confirmed that the **search time** (time-search) or the total **translation time** (time-translation) is longer when words are missing. The decoder collects more translation options (total-transopt), creates more hypotheses (total-hypotheses) and recombines more hypotheses (hyp-recombined). Similar to reordering errors, the calculation of the Language Model also takes longer.

Some of the above observations can be explained as following: Missing words often occur when the decoder prefers a target phrase that translates a relatively longer span of the source sentence, although the target phrase does not contain the translation for one of the words of the source (Guthmann, 2006). The fact that the word is not covered can be attributed to the fact that the chosen phrase alignment had a much higher probability than the correct one.

The issue is illustrated by an example given in Figure 6.2, where we compare the dominant translation with the correct translation through constrained decoding. The German verb of the subordinate clause in the source sentence needs to be reordered and positioned before the object. Nevertheless, although the correct phrase gets good backwards reordering and Language Model scores, in the dominant translation it gets superseded by another phrase that covers a longer span and has much higher phrase table scores (Table 6.6), although it is totally missing the verb.

By deliberately adding an unknown token before the subordinate verb (add-unk), or replacing the object of the sentence by an unknown token (replace), the dominant alignment cannot be used and therefore the correct reordering has the chance to occur. Unfortunately, alignment models tend to learn phrase alignments with missing subordinate verbs as more probable, due to the extremely long distance of the reordering. But very often in such cases, unknown tokens (e.g. foreign names) occur as the object of these verbs, having a positive contribution to a correct translation.

	dominant	constrained	+add unk	replace
distortion	**0**	-10	-12	-6
word penalty	-30	-30	-31	-28
unknown word penalty	0	0	-100	-100
mono-backward reord.	-8.02	**-6.86**	-6.86	-6.86
swap-backward	**0.00**	-0.85	0.00	0.00
other-backward	**0.00**	-0.48	-1.97	-1.56
other-forward	-6.59	**-6.48**	-7.80	-6.65
swap-forward	**0.00**	-1.61	0.00	0.00
mono-forward	**0.00**	-1.03	-0.88	-0.88
LM 5gram	-126.49	**-125.52**	-133.18	-127.85
inverse phrase translation prob.	**-18.45**	-20.75	-20.75	-21.60
inverse lexical weighting	**-40.19**	-43.65	-43.65	-41.47
direct phrase translation prob.	**-25.26**	-26.78	-26.78	-26.52
direct lexical weighting	**-39.31**	-39.68	-39.68	-33.91
phrase penalty	**15.00**	16.00	16.00	16.00
final score	**-26.94**	-27.98	-129.36	-128.34

Table 6.6: *The individual model scores for the alternative decoding processes indicated in Figure 6.2. The correct translation (constrained) is losing from the dominant one due to the high phrase-table scores)*

source:	der amerikanische Präsident Barack Obama kommt für 26 Stunden nach Oslo , Norwegen , um hier als vierter US @-@ Präsident in der Geschichte den Frieden Nobel Preise entgegenzunehmen				
dominant:	in the history of		the Nobel Peace		prize .
pC	-0.59		-0.42		-0.81
c	-2.61		-2.64		-2.71
constrained:	in history	to receive the Nobel Peace Prize			.
pC	-0.52	-1.33	-0.51		-0.14
c	-2.25	-3.11	-2.72		-0.78
add unk.:	in history	to receive the Nobel Peace Prize		UNK	.
pC	-0.52	-1.33	-0.51	0.00	-0.14
c	-2.25	-3.11	-2.72	0.04	-0.78
replace:	in history	to receive	the	UNK	.
pC	-0.52	-1.33	-0.57	0.00	-0.14
c	-2.25	-3.11	-1.36	0.04	-0.78

Figure 6.2: *The calculated log-probabilities (pC) and future cost estimates (c) of the last phrases of a sentence where in the* dominant *translation a word ends up missing. The issue does not occur when an unknown word is added. Scores with alternative decodings with the correct output are also given for comparison.*

6.5 Discussion and further work

The experiments of this chapter are only a first step towards understanding the qualitative impact of common cases in the decoding process and further assessing the usability of glass-box features in the field of QE. The numerical results lead us to several observations and we attempt to provide intuitive explanations and assumptions. Since there can be several improvements to the direction of drawing safer and more useful conclusions, we are giving here some concerns and ideas for future work.

The chosen method is employed to examine feature contributions in a specific model. Nevertheless, there is a concern that these are not necessarily generalized across different models. Moreover, although the decoding features are indeed related to the translation performance, there are concerns that the logistic relationship between decoding features and specific translation errors is so broad, that the statistical relationship is hard to be captured by simple binary classification approaches. Future efforts should therefore examine other machine learning methods, also considering the possibility to model the amount and/or the exact location of errors.

Further work could extend this effort by including a wider range of error categories that describe better the requirements for a translation correct output. Instead of automatically detecting errors on post-edited output, a possible extension could consider modelling error types assigned by humans. Additionally, the analysis of features can be extended in order to cover other types of machine translation, such as hierarchical phrase-based translation, rule-based translation and neural translation.

An obvious application of this analysis would be incorporating the findings into the decoding process, in order to improve it, e.g. by adding features into the decoding engine, so that directly indicate factors that cause errors, or by improving issues on the alignment process.

6.6 Summary

We provided statistical evidence on how internals of the phrase-based SMT decoding process correlate with the existence of two common error types. The existence of the errors in a sentence was modelled over some decoding process features with logistic regression, which resulted into several models with satisfactory precision and recall values. By grouping the observations by error type, we showed important features (representing stages of the decoding process) that are common for several language pairs and directions at the same time. Indications and tendencies were observed, based on the statistically significant coefficients.

Reordering errors have positive correlation to unknown tokens, particularly when they are spread in the source sentence. There is also a positive correlation to the source sentence sentence length and when the target phrases are longer than the source phrases they translate. The latter is attributed to cases where German structures are typically longer than their translations (e.g. past tense).

Missings word have a negative correlation with the source sentence length and a positive correlation with the length of the spans that decoder uses from the source sentence. Among others, it is remarkable that the unknown tokens have a negative correlation to the chance for missing words. Manual investigation shows that an unknown token near the end of the sentence may prevent the decoder choosing an erroneously aligned phrase that omits the German subordinate verb.

Chapter 7

Open source software for Quality Estimation

In this Chapter we present Qualitative[1], an open source software for Quality Estimation, mainly developed for Comparative Quality Estimation and automatic sentence-level translation ranking. The software implements a pipeline annotating the translations with black-box features and applying the machine learning algorithm in order to rank data based on pre-trained models or train and evaluate new models. Optionally, the pipeline can also fetch translations from external MT engines and perqform on-the-fly sentence-level system combination.

The feature generation includes support for Language Models, PCFG parsing by two parsers, language checking tools and various other pre-processors and feature generators. The code follows the principles of object-oriented programming to allow modularity and extensibility, whereas it integrates 25 state-of-the-art external tools into one single Python program, through an interoperable framework with 9 different integration approaches. The tool can operate by processing both batch-files and single sentences. An XML-RPC interface is provided for hooking up with web-services. A comparison of the tool with similar tools from was given earlier, as part of the section on related work (Chapter 1.2.7).

This Chapter is structured as follows. First, we give a short introduction with the available execution modes and the framework for organising training and experiments (Sections 7.1 and 7.2). The basic architecture of the software and the main packages are outlined in Section 7.3, whereas the integration of external components is described in Section 7.4. Finally, some directions for further work and a summary are given in Sections 7.5 and 7.6.

[1] Publicly available at `http://www.github.com/lefterav/qualitative`

7.1 Execution modes

There are several ways the program can be executed. In particular, the basic functionality is provided through the following modes:

- **Command-line interaction**: It allows the user to interact with the sentence selection on the commandline, given a configuration file and a pre-trained selection model. Then the user can sequentially type the source sentence and the respective translations one by one.
- **Batch mode**: This serves for cases where multiple sentences with their respective translations need to be processed in a row (e.g. for training new models or translating full documents).
- **XML-RPC interface**: This instantiates an XML-RPC server awaiting translation requests for one sentence at a time. The server responds to the command `rank`, having as parameters the source and any number of respective translations. It is useful for binding to web-applications.

7.2 Experiment management

Experimenting over the training of new models is organised by using the Python ExpSuite (Rückstieß and Schmidhuber, 2011). This allows the exhaustive exploration of several experimental settings in parallel. We have forked and extended its functionality to provide out-of-the-box parallelised **cross-validation** for any given dataset. Additionally, the split training and test-sets of the cross-validation are **cached** in a common directory, so that they can be re-used for different experimental settings which require the same original dataset. The experiments can be resumed from the step they were left, in case of any unexpected interruption.

The experiment pipeline keeps a structured log of every step of the experiment, which may include the results of the evaluation, but also details about the machine learning process (e.g. the beta coefficients of a log-linear model, or weights of a linear model). The trained models are also dumped in external files, so that they can be re-used later. After all iterations and cross-validation folds of the experiment are concluded, a script allows for creating a comma-separated table that compares all experimental settings against a desired set of metrics.

7.3 Core module

The core of the program is based on Comparative Quality Estimation. The given sentence is first processed within a pipeline of modules. These modules perform text pre-processing and various NLP analysis processes to generate features that indicate the quality of the translation. The generated features are consequently fed to a machine learning algorithm, which employs a statistical model for putting the translations in an order of preference. This statistical model has been previously trained on human-annotated data.

In principle, there can be various combinations of features and machine learning algorithms, depending on what performs best given various properties of the QE task. Pretrained models are provided that follow the settings most successful experiments for several language pairs, as outlined in the previous chapters.

7.3.1 Basic architecture

In our approach, the main core of the program is written in Python. Python has been chosen because it offers the flexibility of dynamic programming, which allows for quick and relatively easy experimentation in many NLP tasks. Functionality from several powerful scientific and machine learning toolkits are available through imported libraries. Additionally, a Python script can be connected to real user applications, either through a web server (e.g. Django), or by offering its functionality via a socket service. This choice offers a flexible framework for both experimentation and practice, although it has got its own limitations (e.g. processing cannot be distributed in many computational machines without additional engineering).

The program is internally structured around several packages:

- **data processing unit** is able to read and write XML and text-based files containing annotated translations. It loads the given data in the memory via the respective data structures or stores any results into new files,
- **preprocessing** performs the necessary string normalisation tasks for the languages at hand (e.g. tokenisation, compound splitting, truecasing etc.),
- **machine translation** sends source sentences to MT engines and receives their translation along with translation meta-information,
- **feature generation** sends the source sentences and their translations to the feature generators and receives vectors of numerical features,
- **machine learning** serves for the communication with machine learning toolkits for two functions: training and testing. During training, it sends a batch of vectors, each one with a golden label and it receives a model. During testing, the model is loaded and given a vector, the predicted label is returned and
- **evaluation** implements several metrics for measuring ranking and translation performance.

For each of the above packages, the commands are organized so that they form a specific interface as a principle of internal *modularity*. This way, the same functionality can be implemented by different classes. For example, every feature generator class has to implement

```xml
<?xml version="1.0" encoding="utf-8"?>
<jcml>
  <judgedsentence langsrc="en" id="11" langtgt="de">
    <src>Go to System Preferences</src>
    <tgt system="system_1" berkeley-loglikelihood="-84.908"
    berkeley-n="19" rank="2">
      Gehen Sie zu Systemeinstellungen
    </tgt>
    <tgt system="system_2" berkeley-loglikelihood="-74.656"
    berkeley-n="5" rank="3">
      Sprung zu Systemeinstellungen
    </tgt>
    <ref berkeley-loglikelihood="-83.355" berkeley-n="18" >
      Gehen Sie zu Systemeinstellungen
    </ref>
  </judgedsentence>
</jcml>
```

***Figure** 7.1: Sample JCML file, containing a source sentence, the reference and two translations with Berkeley Parser scores and human ranks*

at least one function that receives a source sentence and its translations and returns a vector of numerical features.

7.3.2 Data Processing

The majority for the read/write processing of the software is done in a special XML format, the JCML format, which stands for *Judged Corpus Markup Language*. It is a simple XML format that has been devised so that it allows dynamic feature lists but at the same time it can be inspected manually. Reading and writing occurs incrementally to avoid memory-related issues, given the big volume of some data sets. There are also several scripts that allow the conversion from and to other common formats. A sample of such a file can be seen in Figure 7.1.

7.3.3 Machine translation

One of the basic applications of the automatic ranking is the possibility to combine different systems on the sentence level. Such a method is often referred to as a case of *hybrid* MT when it combines different types of systems (e.g. statistical and rule-based). This package handles the communication with translation engines by connecting to remote APIs. It currently supports Moses (Koehn et al., 2006), Lucy (Alonso and Thurmair, 2003), as well as MT-Monkey (Tamchyna et al., 2013) for accessing deployed server installations. The communication with the engines allows fetching translations and glass-box features (translation probability, unknown words etc.), when these are made available by the engine.

Additionally, some techniques of hybridisation are included, such as (a) serial post-editing of a RBMT system with SMT (Section 5.1.3), (b) forcing a RBMT system to fetch domain-specific terms from a domain-specific SMT system and finally (c) a SMT system using an alternative decoding path through a phrase table whose terms have been annotated through Word-Sense Disambiguation (WSD), as in Avramidis et al. (2016a).

The machine translation interface, apart from being a step of the QE pipeline, it can also act as a standalone application, or as a web-service pluggable via XML-RPC.

7.3.4 Feature generation

As explained above, the generation of features is a crucial step for acquiring quality hints regarding the processed sentence. For this purpose, this package provides a set of modules, thereof called *feature generators*. These are classes which process the text of the sentences and return numerical values that describe some aspects of quality.

The functionality of the feature generators includes:

- scoring with LMs, PCFG parsing with two parsers, contrasting scores of several reference-aware metrics such as BLEU, WER, METEOR and Hjerson,
- language correction software such as LanguageTool and Acrolinx IQ, IBM-1 probabilities, as well as token counts.
- **word alignment** based on the IBM-1 model (Brown et al., 1993), allowing to derive the count of aligned PCFG tree spans, nodes and leaves between the source and the target sentence. Whereas this generates hundreds of sparse features, the most prominent of them are expected to help isolate systems that fail to translate grammatically important chunks of the source,
- relative and absolute **position** of every PCFG tag within the sentence, with the goal to capture wrong positioning of grammatical chunks in languages where this is important (e.g. German),
- a re-implementation of the **WMT baseline features** (Callison-Burch et al., 2012) (a.k.a. QuEst features) in Python, including the average number of translations per source word in the segment as given by IBM-1 model with probabilities thresholded in different ways, and the average number of occurrences of the target word within the target segment,
- integration of SRILM and KenLM (Heafield, 2011) which provides efficient of scores from Language Models
- a wrapper for the TreeTagger (Schmid, 1994), which acquires the necessary POS tags for Hjerson (Popović, 2011c)

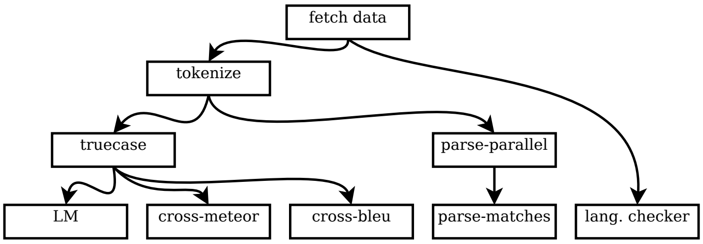

***Figure** 7.2: Sample pipeline of feature generators for one of the most successful feature sets.*

- a connector to the XML-RPC of MoodWSD (Weissenborn et al., 2015), an external software for WSD.

Features generators can run in parallel (i.e. when processed in batch mode for training). For slower tasks (e.g. parsing), it is additionally possible to split a batch in smaller parts and process them in parallel. A sample parallelised feature generator pipeline can be seen in Figure 7.2.

7.3.5 Machine Learning

A transparent and modular internal interface allows for integration of several external ML toolkits. The integration of every ML toolkit should extend an abstract class named `Ranker`. This should implement some basic functions, such as training on a batch of sentences, or producing the ranking given one source sentence and its translations. The implementation of every ML toolkit is also responsible of converting the given sentence data and its features to the data structures understood by the toolkit. Binary classifiers, where available, are wrapped to provide a ranker's functionality.

Currently the following toolkits and learning methods are supported:

- Orange (Demšar et al., 2004) with kNN, Logistic Regression with Stepwise Feature Set Selection or L2 Regularisation and C45 trees,
- Scikit-learn (Pedregosa et al., 2011) with SVM with Grid parameter optimisation over cross-validation, Decision Trees, Gaussian Naïve Bayes, LDA and Quadratic Discriminant Analysis, Bagging, AdaBoost and Gradient Boosting and feature selection methods such as RFECV
- MLpython[2] with listwise ranking methods, such as ListNet (Cao et al., 2007).

[2] MLpython is described at `http://www.dmi.usherb.ca/~larocheh/mlpython/`

7.3.6 Evaluation

The evaluation phase is the last part of the experiment process, as the trained models are tested against gold-sets and need to be evaluated accordingly. The evaluation phase offers a wide range of ranking metrics, including Kendall's tau, inverse-weighed Kendall's τ and its theoretical p-values and confidence intervals.

A set of ranking metrics originating from Information Retrieval include the NDCG, First Answer Reciprocal Rank (FARR; Radev et al., 2002), Mean Reciprocal Rank (MRR; Radev et al., 2002) and Expected Reciprocal Rank (ERR; Chapelle et al., 2009). Since the mathematical formula for the computation of the Expected Reciprocal Rank is computed in exponential time, we use the simplified computation suggested by Chapelle et al. (2009), which is outlined in algorithm 1. The algorithm reduces the computational perplexity by calculating the relevance grades g_i only once for each rank i. This is used during the loop for calculating the relevance probability R_i and gradually augmenting the ERR value.

Algorithm 1: Linear computation of Expected Reciprocal Rank

```
foreach $i$ in $[0, n]$ do $g_i \leftarrow$ RelevanceGrade($i$);
$p \leftarrow 1, ERR \leftarrow 0.$
for $r \leftarrow 1$ to $n$ do
    $R \leftarrow$ RelevanceProb($g_r$)
    $ERR \leftarrow ERR + p \times R/r$
    $p \leftarrow p \times (1 - R)$
return $ERR$
```

Finally, the evaluation phase includes automatic metric scores (BLEU, METEOR, TER, WER, Hjerson) for the performance of the system combination and its components against the reference translation.

7.4 Connecting external components

In order to ensure state-of-the-art functionality for all the underlying NLP tasks, the software integrates a multitude of external tools into one single Python program, through an interoperable framework. Here we present 9 approaches taken to connect 25 external components, originally developed in various programming languages. There are two main categories of communicating with external software components, based on whether the execution of the external software is controlled from within our Python code, which we will call the "host", or whether it is run as a remote service. The full design of the integration is shown in Figure 7.3

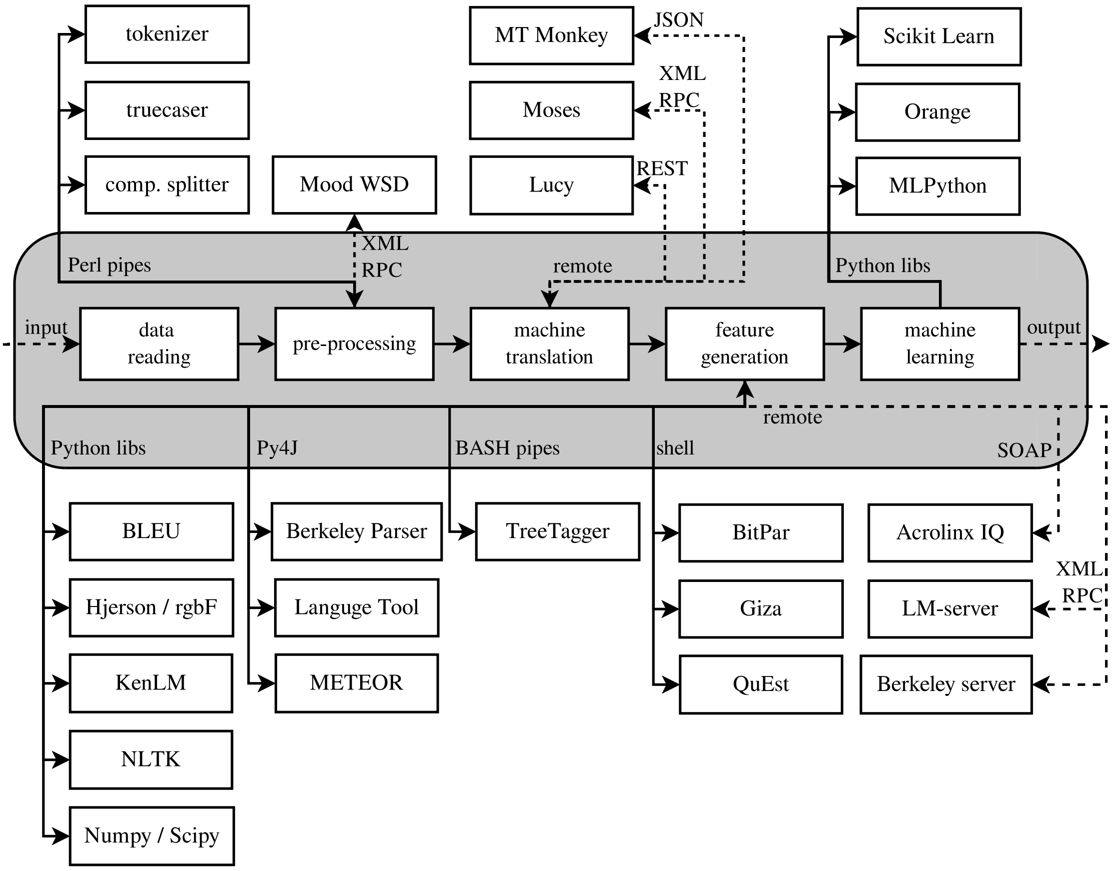

***Figure** 7.3: Full diagram of the components that have been integrated into the application*

7.4.1 Inherent integration

In these functions, the execution of the external software is encapsulated into the host. The goal is to keep the external tool running in the background so that it can receive requests from the host. It gets automatically unloaded when the host program is finished. The part of the host program or the code which handles the specifities of the communication is referred to as a "connector".

7.4.1.1 Native Python libraries

Many pieces of Python open-source software already offer their functionality in openly available libraries. This is the easiest and most efficient type of integration, as all of the public functions of the included software can be directly called from within our host Python code. The software served by this method includes:

- BLEU (Papineni et al., 2001), Levenshtein Distance (Levenshtein, 1966) and RgbF (Popović, 2012b) for MT evaluation scores
- Hjerson (Popović, 2011c) for automatic detection of MT errors
- KenLM (Heafield, 2011) for language modelling
- MLPython[3], Orange (Demšar et al., 2004) and Scikit-learn (Pedregosa et al., 2011) for machine learning functions.
- NLTK (Loper and Bird, 2002) for several simple NLP tasks
- NumPy (Van Der Walt et al., 2011) for memory-efficient handling of numerical arrays and SciPy (Oliphant, 2007) for scientific (e.g. complex mathematical or statistical) functions.

7.4.1.2 Java programs

Py4j[4] was chosen as a solution to integrate functionality from open-source Java programs into Python. The Java Virtual Machine (JVM) starts in the background including the required Java Packages (jar) in the classpath. Then, a Py4j gateway connects with the JVM via a socket and makes all public classes and functions loadable and callable from within Python. Python types are automatically converted to Java types and vice versa. If the processes are thread-safe on the Java side, they can be also parallelised in several Python threads.

This method is used to connect with:

- Berkeley Parser (Petrov et al., 2006) for parsing with Probabilistic Context Free Grammars (PCFG),
- Language Tool (Naber, 2003; Miłkowski, 2012) for rule-based language checking and

[3] http://www.dmi.usherb.ca/~larocheh/mlPython/

[4] http://www.py4j.org

- METEOR (Lavie and Agarwal, 2007; Denkowski and Lavie, 2014) for MT evaluation scoring.

This method is efficient and allows wide access and parametrisation to the functionality of the external Java program. Nevertheless, it also requires good knowledge to its internal structure, e.g. via a Java API documentation or by reading the Java source code. This is needed because the imported objects, functions and variables have to be treated in Python the same way they would do in Java. Additionally, the host needs to know or maintain a knowledge of the system socket where the JVM operates, which makes it complicated to run many hosts on the same JVM. In a few cases, parts of the source code had to be modified and be re-build, since not all required functions were declared as public, which is a major requirement.

7.4.1.3 SWIG

Simplified Wrapper and Interface Generator (SWIG) allows wrapping C++ code as a Python library. Creating such a connector allows to parse C/C++ interfaces and generate the 'glue code' for Python to call into the C/C++ code. In our program we have not developed such a connector, but we have experimented with SWIG-SRILM (Madnani, 2009), an existing wrapper around SRILM (Stolcke, 2002).

7.4.1.4 Pipes

An external commandline-based software is launched by the host as a sub-process in the background. The standard input, the standard output and the error output can be captured within a Python object (a *pipe*). Therefore, a program-specific connector needs to be written. It should be aware of the commandline behaviour of the software and simulate that through the pipe. The sub-process is treated as a black-box, i.e. no access to particular internal functions is possible.

For example, a standard tokeniser from the Moses scripts would read from the standard input all characters, waiting for and "end of line". Once the "end of line" is received, the tokenisation takes place and the tokenised string is returned through the standard output.

This approach is mainly used for Perl scripts and C++ programs but can be adapted for any commandline application. Such software includes:

- Moses scripts for pre-processing and post-processing, such as punctuation normaliser, tokeniser, compound splitter (Koehn and Knight, 2003), truecaser (Och et al., 2003), de-truecaser, de-tokeniser etc. Although re-implementations for most of these exist in Python and therefore could be directly included in our code, one may still require to stick to the original Moses Perl scripts, if they want to re-use pre-trained Moses translation models or acquire results comparable with other scientific works that use these state-of-the-art Perl scripts.

- TreeTagger for POS tagging (Schmid, 1994) integrated via the TreeTaggerWrapper (Pointal, 2015).

The advantage of this method is that it can be adapted for many programs without requiring knowledge of their internal coding or functioning, while it still allows loading a tool into memory once and sending individual requests when the host program needs it. The disadvantage is that the only way of interaction is through the standard input and output, which offer no flexibility for parametrisation or passing more complex types. Additionally, reading standard output often requires excessive use of regular expressions to understand some complex output, which would otherwise be intended for the visual understanding of the user. Unexpected errors and exceptions are hard to capture, too. We should also mention that some tools only work with input and output files (batch mode) and do not support per-request communication with standard input and output. Finally, serious deficiencies have been noted concerning the buffering support of the pipes, which may prevent data from being transferred through the standard input/output.

7.4.1.5 Shell with external files

The data to process is written by the host on a temporary file. The external program is launched once, asked to process the given temporary file as an input and write its output in another temporary file, which consequently gets read by the host. This is the last resort for having the host communicate with external tools, since loading the entire program per request and writing external files is not efficient for single sentences and is useful only for processing batches of requests. We also noticed that some programs of this kind do not allow many instances to be run in parallel (e.g. because they require an exclusive lock on some internal files, whose location is often non-parametrisable).

We used this method for aligning sentences with Giza++ (Och and Ney, 2003), acquiring baseline features from QuEst and doing PCFG parsing with BitPar (Schmid, 2004) with the help of a wrapper (van Cranenburgh, 2010). This method was useful only for experiments that did not require parallelisation and single requests.

7.4.2 Integrating functionality as a remote service

An additional possibility of integrating an external tool is by sending requests to it as a remote service. In this case, the external tool must provide a server which initially loads the program and implements a network protocol of requests and responses. It waits until a request is received from the host, in order to run the required functions. The result of the functions is then sent with a corresponding response.

Four such protocols and the respective tools we have used are:

- **JSON**: with MT-Monkey (Tamchyna et al., 2013), which acts as a hub and a load balancer for fetching translations from several MT engines
- **SOAP**: with Acrolinx IQ (Siegel, 2011) for language checking

- **REST**: with the Lucy rule-based MT system (Alonso and Thurmair, 2003).
- **XML-RPC**: with Moses (Koehn et al., 2006) for Statistical MT, with LM-server (Madnani, 2009) for LM scoring, with our own XML-RPC wrapper of Berkeley Parser and with Mood (Weissenborn et al., 2015), a Word Sense Disambiguation analyzer.

Such an integration is straightforward if the tool already provides a protocol interface, since the protocols allow for easy mapping of function and variable types across many different programming languages. This solution is based on a network connection, so it is also desirable when one needs to distribute different computationally or memory intensive modules to many computational servers. Nevertheless, a network communication may be considerably slower due to the network overhead. Additionally, starting and stopping remote services cannot be easily controlled by the host, unlike the encapsulation described in the previous section.

7.5 Further work

Some parts of the pipeline have been also used for other types of QE, such as quality score prediction for single outputs and error prediction (Avramidis, 2012b; Avramidis and Popović, 2014). Small extensions to provide abstract interfaces for different QE problems, including classification and regressions, would be desirable.

We are also aware that the glass-box feature integration requires extensions to support most MT engines, although this faces the barrier that not all glass-box features are easily available.

Finally, despite the huge potential for machine learning, the processing of large amount of data in linear files leads into bottlenecks, in case they must be analyzed or processed selectively. We plan to support more effective data types (e.g. JSON). A possible solution would include the implementation of smart databases and other data-effective techniques.

7.6 Summary

We developed an open source software for Comparative Quality Estimation. We presented the main modes of operation (included local and server mode) and the framework for training and experimentation. Then, we outlined the basic architecture and described the functionality of the main packages. An additional emphasis was given to the methods for integrating external state-of-the-art software, originally written in other languages. In that context, 9 different methods were employed in order to integrate 25 external tools. ‘

Chapter 8

Conclusions and further work

8.1 Conclusions

Comparative Quality Estimation is a distinct type of Quality Estimation, motivated by the possibility to improve the objectivity of translation comparisons, while reducing the requirements for the ground truth by favouring ordinality against cardinality (Chapter 1). The ranking mechanism is defined as a pipeline of the feature generation function and the ranker, following the principles of a typical supervised machine learning problem (Chapter 2),

It is shown that such a ranking mechanism can predict rankings which correlate significantly with human rankings and are significantly better than random or alphabetical rankings (Chapter 3). Additionally, the rankings predicted by the ranking mechanism compete with state-of-the-art reference-aware metrics on sentence-level ranking, as it performs significantly better than the automatic metrics BLEU, METEOR, TER and rgbF, for the language pairs where focused feature engineering took place. The ranking mechanism also beats all other metrics in language pairs where the feature engineering from other language pairs was adopted, apart from one automatic metric, METEOR, which is significantly as good as the ranking mechanism. This suggests that elaborate features and machine learning may provide more information about relative translation quality than direct comparison with references.

When compared with other methods of automatic ranking (pairwise classification, individual continuous values with similarity threshold, a multiple classifier and a method similar to n-best reranking) the ranking mechanism is significantly better than any of them for the language pair where particular feature engineering took place, whereas it got the second position for a second language pair.

The fact that ranking is decomposed into pairwise decisions allows the integration of several machine learning algorithms with positive results. The best ranking models are achieved with Gradient Boosting and AdaBoost, Logistic Regression and Linear Discriminat Analysis. A considerable amount of ties was eliminated thanks to the weighting of the pairwise decisions with the classification confidence.

Iterative feature engineering (Chapter 4) indicates that for German-English, information from the parsing of the target sentences can be more valuable as features than the LM probability. The most valuable features for our Logistic Regression model are the number of the unknown words, the number of tokens, the contrastive METEOR score, the number of VPs and the number of the alternative n-best parse trees of the target translations (Section 4.2.5). Superficial source features are not useful for the purpose of Comparative QE. Nevertheless features from the alignment of particular tree nodes between the source and the target sentence and alignment scores (IBM-Model 1) appear to be important (Section 4.3.4). Additionally, the choice of the target-side grammatical features depends on the language direction. Indicatively, for English-German, features indicating the position of verb phrases in the sentence play a bigger role, whereas for German-English, features such as the count of subordinate infinitives and prepositional phrases seems more important (Section 4.3.4).

Feature Selection can improve the performance of the ranking mechanism. For a model trained with Logistic Regression, Stepwise Feature Set Selection performs significantly better than L2 Regularisation. For a model trained with Gradient Boosting, Recursive Feature Elimination can reduce the amount of features to about a third, without any negative implication to the ranking performance (Section 4.3.3).

The ranking mechanism can be used for hybrid system combination. Using pairwise comparisons to combine three different systems from 2 different MT technologies (SMT and RBMT) leads to a combination that performs significantly better than its components, as measured by automatic metrics (Chapter 5).

The appearance of common error types can be explained with empirical analysis on the MT output (Chapter 6). The correlation of the existence of two common MT errors with features from the phrase-based decoding process can lead to conclusions about events in the decoding process that caused the respective errors. Reordering errors have a higher probability to appear when there are proportionally more unknown tokens and they are spread more along the sentence. They also occur more when there are longer source sentences and when the target phrases are longer than the source phrases they translate, possibly due to grammatical phenomena with length imbalance (such as German past tenses) known to have high reordering requirements. There is a higher probability of words to be missing when the source sentences are shorter and when the decoder chooses longer spans from the source sentence. Additionally, the number of unknown words in the source has a negative correlation with the number of missing words. Finally, it is shown that unknown words, when they occur as the object of subordinate verbs, may prevent the verb omission by interrupting long verb-less spans originating from erroneous alignments.

8.2 Further work

The goal of this line of research remains to bring the research on Quality Estimation closer to solving common problems in practice. There are reports that QE has been integrated in several professional and industrial translation pipelines as an *additional evaluation or selection component* on top of the existing translation systems (Martins et al., 2016). Nevertheless, the operation of QE is clearly supplementing missing capabilities of the underlying translation systems. So an obvious further goal would be the *integration of QE into the overall design and development cycle* of the systems, or even as an inherent component of the system architecture itself. Efforts have been done, but no optimal solution has been shown (Hopkins and May, 2011; Burchardt et al., 2013; Aranberri et al., 2016). Additionally, little progress has been made with regards to using QE as a *tool for evaluation*, despite the positive results over the last 7 years (few of which also included in this thesis). This can be of course seen in parallel with the yet persisting use of BLEU by the research community, although its flaws have been underlined and improved metrics have been available (Bojar et al., 2016b).

We may assume that the observed situation derives from the fact that the engineering overhead of configuring and running QE is often not justified by the potential profit. The overhead itself resorts to many aspects of QE that are language-specific or system-oriented, often requiring fine-grained work for feature engineering and other necessary steps.

After the experiments of this thesis were concluded, Neural MT (Bahdanau et al., 2014) emerged, leading to a significant change for the landscape of the field. Despite the obvious improvements (Junczys-Dowmunt et al., 2016), qualitative analysis (Bentivogli et al., 2016) reveals that errors are still apparent, indicating that the role of QE may be relevant in several sensitive applications. However, the overall spectrum of the MT errors is undergoing a shift as the MT output starts becoming less distinguishable from human translations (Wu et al., 2016). The generated sentences are clearly more fluent, obeying grammatical and syntactic rules better than statistical systems. Given enough data, a QE model would require an enhanced machine learning and feature engineering power in order to cope with this new spectrum of errors and better differentiate quality.

The most crucial change is nevertheless foreseen on the conceptual level. The new learning method has successfully replaced scattered modules and ad-hock solutions with one single self-contained concept; a unified concept that is considered as one of its main advantages. In that sense, the exact form of QE is being at stake: Will QE still be useful as an independent module, or will neural MT models eventually be capable of inherently performing QE on their output? Can the findings of QE, or even QE itself, be integrated in the learning method in an effective way?

Keeping these aspects in consideration, we think that further work should evaluate and redefine the role of QE given the entry of Neural MT and investigate the use of approaches using deep learning for the purposes of QE and vice versa.

Bibliography

Agresti, A. (1996). *An introduction to categorical data analysis*, volume 135. Wiley New York.

Akiba, Y., Imamura, K., and Sumita, E. (2001). Using multiple edit distances to automatically rank machine translation output. In *Proceedings of the MT Summit VIII*, pages 15–20, Santiago de Compostela, Spain. International Association for Machine Translation.

Akiba, Y., Watanabe, T., and Sumita, E. (2002). Using Language and Translation Models to Select the Best Among Outputs from Multiple MT Systems. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Almaghout, H. and Specia, L. (2013). A CCG-based Quality Estimation Metric for Statistical Machine Translation. In *Proceedings of the Machine Translation Summit XIV*, pages 223–230, Nice, France. International Association for Machine Translation.

Alonso, J. A. and Thurmair, G. (2003). The comprendium translator system. In *Proceedings of the Ninth Machine Translation Summit*, New Orleans, USA. International Association for Machine Translation.

Ament, K. (1998). Babelfish: Real-Time Machine Translation on the Internet. *Intercom*.

Aranberri, N., Avramidis, E., Burchardt, A., Klejch, O., Popel, M., and Popović, M. (2016). Tools and Guidelines for Principled Machine Translation Development. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1877–1882, Portoroz, Slovenia. European Language Resources Association.

Avramidis, E. (2011). DFKI System Combination with Sentence Ranking at ML4HMT-2011. In *Proceedings of the International Workshop on Using Linguistic Information for Hybrid Machine Translation (LIHMT 2011) and of the Shared Task on Applying Machine Learning Techniques to Optimising the Division of Labour in Hybrid Machine Translation (M. Sha*. Center for Language and Speech Technologies and Applications (TALP), Technical University of Catalonia.

Avramidis, E. (2012a). Comparative Quality Estimation: Automatic Sentence-Level Ranking of Multiple Machine Translation Outputs. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 115–132, Mumbai, India. The COLING 2012 Organizing Committee.

Avramidis, E. (2012b). Quality estimation for Machine Translation output using linguistic analysis and decoding features. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 84–90, Montréal, Canada. Association for Computational Linguistics.

Avramidis, E. (2013a). RankEval: Open Tool for Evaluation of Machine-Learned Ranking. *The Prague Bulletin of Mathematical Linguistics*, 100:63–72.

Avramidis, E. (2013b). Sentence-level ranking with quality estimation. *Machine Translation*, 27(3-4):239–256.

Avramidis, E. (2014). Efforts on Machine Learning over Human-mediated Translation Edit Rate. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 302–306, Baltimore, Maryland, USA. Association for Computational Linguistics.

Avramidis, E. (2016a). Interoperability in MT Quality Estimation or wrapping useful stuff in various ways. In *Proceedings of the LREC 2016 Workshop on Translation Evaluation: From Fragmented Tools and Data Sets to an Integrated Ecosystem*, pages 1–6, Portoroz, Slovenia. European Language Resources Association.

Avramidis, E. (2016b). Qualitative: Python Tool for MT Quality Estimation Supporting Server Mode and Hybrid MT. *The Prague Bulletin of Mathematical Linguistics*, 106:147–158.

Avramidis, E. (2017a). Comparative Quality Estimation for Machine Translation: Observations on machine learning and features. *Proceedings of the 20th Annual Conference of the European Association for Machine Translation, The Prague Bulletin of Mathematical Linguistics*, (108):307–318.

Avramidis, E. (2017b). QE::GUI - A Graphical User Interface for Quality Estimation. *The Prague Bulletin of Mathematical Linguistics*, 109:51–60.

Avramidis, E. (2017c). Sentence-level quality estimation by predicting HTER as a multi-component metric. In *Second Conference on Machine Translation*, pages 534–539, Copenhagen, Denmark. Association for Computational Linguistics.

Avramidis, E., Burchardt, A., Hunsicker, S., Popović, M., Tscherwinka, C., Torres, D. V., and Uszkoreit, H. (2014a). The taraXÜ Corpus of Human-Annotated Machine Translations. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2679–2682. European Language Resources Association.

Avramidis, E., Burchardt, A., Macketanz, V., and Srivastava, A. (2016a). DFKI's system for WMT16 IT-domain task, including analysis of systematic errors. In *Proceedings of the First Conference on Machine Translation*, pages 415–422, Berlin, Germany. Association for Computational Linguistics.

Avramidis, E., Burchardt, A., Popović, M., and Uszkoreit, H. (2015a). Towards Deeper MT - A Hybrid System for German. In *Proceedings of the 1st Deep Machine Translation Workshop (DMTW-2015)*, pages 12–19, Prague, Czech Republic. Charles University in Prague, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics.

Avramidis, E., Costa-Jussà, M. R., Federmann, C., van Genabith, J., Melero, M., and Pecina, P. (2012). A Richly Annotated, Multilingual Parallel Corpus for Hybrid Machine Translation. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 2189–2193, Istanbul, Turkey. European Language Resources Association.

Avramidis, E., Macketanz, V., Burchardt, A., Helcl, J., and Uszkoreit, H. (2016b). Deeper Machine Translation and Evaluation for German. In Hajic, J., van Noord, G., and Branco, A., editors, *Proceedings of the 2nd Deep Machine Translation Workshop. Deep Machine Translation Workshop (DMTW)*, pages 29–38, Lisbon, Portugal. Charles University Prague, Charles University, Prague.

Avramidis, E., Macketanz, V., Lommel, A., and Uszkoreit, H. (2018). Fine-grained evaluation of Quality Estimation for Machine translation based on a linguistically-motivated Test Suite. In *Proceedings of the First Workshop on Translation Quality Estimation and Automatic Post-Editing*, pages 243–248, Boston, MA, USA.

Avramidis, E. and Popović, M. (2013). Machine learning methods for comparative and time-oriented Quality Estimation of Machine Translation output. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 329–336, Sofia, Bulgaria. Association for Computational Linguistics.

Avramidis, E. and Popović, M. (2014). Correlating decoding events with errors in Statistical Machine Translation. In *Proceedings of the 11th International Conference on Natural Language Processing (ICON2014)*, pages 20–29, Goa, India. International Institute of Information Technology, Natural Language Processing Association, India.

Avramidis, E., Popović, M., and Burchardt, A. (2015b). DFKI's experimental hybrid MT system for WMT 2015. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 66–73, Lisbon, Portugal. Association for Computational Linguistics.

Avramidis, E., Popović, M., Vilar, D., and Burchardt, A. (2011). Evaluate with Confidence Estimation: Machine ranking of translation outputs using grammatical features. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 65–70, Edinburgh, Scotland. Association for Computational Linguistics.

Avramidis, E., Poustka, L., and Schmeier, S. (2014b). Qualitative: Open source Python tool for Quality Estimation over multiple Machine Translation outputs. *The Prague Bulletin of Mathematical Linguistics*, 102:5–16.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *Computer Research Repository*, abs/1409.0.

Banerjee, S. and Lavie, A. (2005). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgements. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, MI. Association for Computational Linguistics.

Beck, D., Shah, K., and Specia, L. (2014). SHEF-Lite 2.0: Sparse Multi-task Gaussian Processes for Translation Quality Estimation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 307–312, Baltimore, Maryland, USA. Association for Computational Linguistics.

Beck, D., Specia, L., and Cohn, T. (2013). Reducing Annotation Effort for Quality Estimation via Active Learning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 543–548, Sofia, Bulgaria. Association for Computational Linguistics.

Bentivogli, L., Bisazza, A., Cettolo, M., and Federico, M. (2016). Neural versus Phrase-Based Machine Translation Quality: a Case Study. In *The 2016 Conference on Empirical Methods on Natural Language Processing*, Austin, Texas, USA. Association for Computational Linguistics.

Berka, J., Bojar, O., Fishel, M., Popović, M., and Zeman, D. (2012). Automatic MT Error Analysis: Hjerson Helping Addicter. In *Proceedings of the Eight International Conference on Language Resources and Evaluation*, pages 2158–2163, Istanbul, Turkey. European Language Resources Association.

Bernth, A. (1999). A confidence index for machine translation. In *Proceedings of 8th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-99)*, pages 120–127.

Besacier, L. and Lecouteux, B. (2013). LIG System for WMT13 QE Task : Investigating the Usefulness of Features in Word Confidence Estimation for MT. *WMT-2013: 8th Workshop on Statistical Machine Translation*, pages 386–391.

Biçici, E. (2013). Referential Translation Machines for Quality Estimation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 343–351, Sofia, Bulgaria. Association for Computational Linguistics.

Biçici, E., Groves, D., and Van Genabith, J. (2013). Predicting sentence translation quality using extrinsic and language independent features. *Machine Translation*, 27(3-4):171–192.

Biçici, E. and Way, A. (2014). Referential Translation Machines for Predicting Translation Quality. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 313–321, Baltimore, Maryland, USA. Association for Computational Linguistics.

Birch, A., Blunsom, P., and Osborne, M. (2009). A Quantitative Analysis of Reordering Phenomena. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 197–205, Athens, Greece. Association for Computational Linguistics.

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 20th international conference on Computational Linguistics (COLING 2004)*, pages 315–321, Stroudsburg, PA, USA. Association for Computational Linguistics.

Blatz, J., Fitzgerald, E., Foster, G., Gandrabur, S., Goutte, C., Kulesza, A., Sanchis, A., Ueffing, N., Goutte, C., Sanchis, A., and Ueffing, N. (2003). Confidence estimation for statistical machine translation. In Rollins, M., editor, *Johns Hopkins Summer Workshop Final Reports*, pages 1–108, Baltimore, Maryland, USA. Johns Hopkins University, Yale University Press.

Bojar, O., Buck, C., Callison-Burch, C., Federmann, C., Haddow, B., Koehn, P., Monz, C., Post, M., Soricut, R., and Specia, L. (2013a). Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 12–58, Sofia, Bulgaria. Association for Computational Linguistics.

Bojar, O., Buck, C., Callison-Burch, C., Haddow, B., Koehn, P., Monz, C., Post, M., Saint-Amand, H., Soricut, R., and Specia, L., editors (2013b). *Proceedings of the Eighth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Sofia, Bulgaria.

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. Association for Computational Linguistics.

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. (2016a). Findings of the 2016 Conference on Machine Translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Bojar, O., Chatterjee, R., Federmann, C., Haddow, B., Huck, M., Hokamp, C., Koehn, P., Logacheva, V., Monz, C., Negri, M., Post, M., Scarton, C., Specia, L., and Turchi, M. (2015). Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal. Association for Computational Linguistics.

Bojar, O., Graham, Y., Kamran, A., and Stanojević, M. (2016b). Results of the WMT16 Metrics Shared Task. In *Proceedings of the First Conference on Machine Translation*, pages 199–231, Berlin, Germany. Association for Computational Linguistics.

Brants, S., Dipper, S., Eisenberg, P., Hansen-Schirra, S., König, E., Lezius, W., Rohrer, C., Smith, G., and Uszkoreit, H. (2004). TIGER: Linguistic Interpretation of a German Corpus. *Research on Language and Computation*, 2(4):597–620.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, 24(421):123–140.

Breiman, L. (2001). Random Forests. *Machine Learning*, 1(45):5–32.

Breiman, L., Friedman, J., Stone, C., and Olshen, R. A. (1984). *Classification and Regression Trees*. Chapman and Hall/CRC.

Brown, P., Cocke, P., Pietra, S. D., Pietra, V. D., Jelinek, F., Lafferty, J., Mercer, R., and Roossin, P. S. (1990). A statistical approach to machine translation. *Computational linguistics*, 16(2)(2):79–85.

Brown, P., Pietra, S. A. D., Pietra, V. J. D., and Mercer, R. L. (1993). The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics - Special issue on using large corpora*, 19(2):263–311.

Burchardt, A., Tscherwinka, C., Avramidis, E., and Uszkoreit, H. (2013). Machine translation at work. In *Studies in Computational Intelligence*, volume 458, pages 241–261. Springer, Berlin.

Callison-Burch and Chris (2009). Fast, cheap, and creative: evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, volume 1, pages 286–295, Singapore. Association for Computational Linguistics.

Callison-Burch, C. and Flournoy, R. S. (2001). A Program for Automatically Selecting the Best Output from Multiple Machine Translation Engines. In *Proceedings of the Machine Translation Summit VIII*, pages 63–66, Proceedings of the Machine Translation Summit VIII. European Association for Machine Translation.

Callison-Burch, C., Fordyce, C., Koehn, P., Monz, C., and Schroeder, J. (2007). (Meta-) evaluation of machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 136–158, Prague, Czech Republic. Association for Computational Linguistics.

Callison-Burch, C., Fordyce, C., Koehn, P., Monz, C., and Schroeder, J. (2008). Further Meta-Evaluation of Machine Translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 70–106, Columbus, Ohio. Association for Computational Linguistics.

Callison-Burch, C., Koehn, P., Monz, C., Peterson, K., Przybocki, M., and Zaidan, O. (2010). Findings of the 2010 Joint Workshop on Statistical Machine Translation and Metrics for Machine Translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics*, pages 17–53, Uppsala, Sweden. Association for Computational Linguistics.

Callison-Burch, C., Koehn, P., Monz, C., Post, M., Soricut, R., and Specia, L. (2012). Findings of the 2012 Workshop on Statistical Machine Translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada. Association for Computational Linguistics.

Callison-Burch, C., Koehn, P., Monz, C., and Schroeder, J. (2009). Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28, Athens, Greece. Association for Computational Linguistics.

Callison-Burch, C., Koehn, P., Monz, C., and Zaidan, O. (2011). Findings of the 2011 Workshop on Statistical Machine Translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 22–64, Edinburgh, Scotland. Association for Computational Linguistics.

Cameron, A. (1998). *Regression analysis of count data*. Cambridge University Press, Cambridge UK; New York NY USA.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. In *The 24th Annual International Conference on Machine Learning*, pages 129–136, Corvallis, OR, USA. Association for Computing Machinery.

Cavar, D., Küssner, U., and Tidhar, D. (2000). From Off-line Evaluation to On-line Selection. pages 599–612. Springer Verlag, Berlin, Germany.

Chapelle, O. and Chang, Y. (2011). Yahoo! Learning to Rank Challenge Overview. *Journal of Machine Learning Research-Proceedings Track*, 14:1–24.

Chapelle, O., Metlzer, D., Zhang, Y., and Grinspan, P. (2009). Expected reciprocal rank for graded relevance. In *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09*, page 621, New York, USA. ACM Press.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270, Ann Arbor, Michigan, USA. Association for Computational Linguistics.

Christensen, D. (2005). Fast algorithms for the calculation of Kendall's τ. *Computational Statistics*, 20(1):51–62.

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association*, 74(368):829–836.

Coomans, D. and Massart, D. (1982). Alternative k-nearest neighbour rules in supervised pattern recognition. *Analytica Chimica Acta*, (138):15–27.

Corston-Oliver, S., Gamon, M., and Brockett, C. (2001). A machine learning approach to the automatic evaluation of machine translation. *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics - ACL '01*, pages 148–155.

de Souza, J. G. C., González-Rubio, J., Buck, C., Turchi, M., and Negri, M. (2014a). FBK-UPV-UEdin participation in the WMT14 Quality Estimation shared-task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 322–328, Baltimore, Maryland, USA. Association for Computational Linguistics.

de Souza, J. G. C., Turchi, M., and Negri, M. (2014b). Machine Translation Quality Estimation Across Domains. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 409–420, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Demšar, J., Zupan, B., Leban, G., and Curk, T. (2004). Orange: From Experimental Machine Learning to Interactive Data Mining. In *Principles of Data Mining and Knowledge Discovery*, pages 537–539.

Denkowski, M. and Lavie, A. (2014). Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

Duh, K. (2008). Ranking vs. regression in machine translation evaluation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, number June, pages 191–194, Columbus, Ohio, USA. Association for Computational Linguistics.

Durrani, N., Haddow, B., Heafield, K., and Koehn, P. (2013). Edinburgh's Machine Translation Systems for European Language Pairs. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 114–121, Sofia, Bulgaria. Association for Computational Linguistics.

Eisele, A. and Chen, Y. (2010). MultiUN: A Multilingual Corpus from United Nation Documents. In *Proceedings of the Seventh conference on International Language Resources and Evaluation. International Conference on Language Resources and Evaluation (LREC-2010), May 19-21, La Valletta, Malta*, pages 2868–2872. European Language Resources Association.

Federmann, C. (2013). Multi-Engine Machine Translation as a Lifelong Machine Learning Problem. In *Proceedings of the AAAI 2013 Spring Symposium on Lifelong Machine Learning*, Stanford, CA, USA. AAAI Press.

Federmann, C., Avramidis, E., Costa-Jussà, M. R., van Genabith, J., Melero, M., Pecina, P., and Ruiz Costa-jussà, M. (2012). The ML4HMT Workshop on Optimising the Division of Labour in Hybrid Machine Translation. In *Proceedings of the 8th ELRA Conference on Language Resources and Evaluation*, pages 3430–3435, Istanbul, Turkey. European Language Resources Association.

Felice, M. and Specia, L. (2012). Linguistic Features for Quality Estimation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 96–103, Montr{é}al, Canada. Association for Computational Linguistics.

Formiga, L., Gonzàlez, M., Barrón-Cedeño, A., Fonollosa, J. A. R., Màrquez, L., Gonz, M., and Barr, A. (2013a). The TALP-UPC Approach to System Selection: Asiya Features and Pairwise Classification Using Random Forests. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 359–364, Sofia, Bulgaria. Association for Computational Linguistics.

Formiga, L., Màrquez, L., and Pujantel, J. (2013b). Real-life Translation Quality Estimation for MT System Selection. In *Proceedings of MT Summit XIV*, pages 69–76, Nice, France.

Freund, Y. and Schapire, R. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. *Computational learning theory*, 55:119–139.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232.

Gamon, M., Aue, A., and Smets, M. (2005). Sentence-level MT evaluation without reference translations : Beyond language modeling. In *Proceedings of the 10th Annual Conference of the European Association for Machine Translation (EAMT 2005)*, number 2001, pages 103–111, Budapest, Hungary.

Gandrabur, S. and Foster, G. (2003). Confidence estimation for translation prediction. In *Proceedings of the seventh conference on Natural language learning (HLT-NAACL 2003)*, pages 95–102. Association for Computational Linguistics.

Garvin, D. (1984). What Does "Product Quality" Really Mean? *Sloan management review*, page 25.

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

Giménez, J. and Marquez, L. (2010). An Open Toolkit for Automatic Machine Translation (Meta-) Evaluation. *The Prague Bulletin of Mathematical Linguistics*, 94:77–86.

Goodstadt, L. (2010). Ruffus: a lightweight Python library for computational pipelines. *Bioinformatics*, 26(21):2778–2779.

Gries, S. T. and Wulff, S. (2012). Regression analysis in translation studies. In Oakes, M. P. and Ji, M., editors, *Quantitative methods in corpus-based translation studies: A practical guide to descriptive translation research*, chapter 2, pages 35–52. John Benjamins Publishing.

Guthmann, N. (2006). *Exploiting Linguistically-Enriched Models for Phrase-Based Statistical Machine Translation*. PhD thesis, University of Edinburgh.

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.

Guzmán, F. and Vogel, S. (2012). Understanding the Performance of Statistical MT Systems: A Linear Regression Framework. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 1029–1044, Mumbai, India. The COLING 2012 Organizing Committee.

Han, A. L.-F., Lu, Y., Wong, D. F., Chao, L. S., He, L., and Xing, J. (2013). Quality Estimation for Machine Translation Using the Joint Method of Evaluation Criteria and Statistical Modeling. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 365–372, Sofia, Bulgaria. Association for Computational Linguistics.

Hardmeier, C., Nivre, J., and Tiedemann, J. (2012). Tree Kernels for Machine Translation Quality Estimation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 109–113, Montréal, Canada.

Harrell, F. E. (2001). *Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis*. Springer.

Harris, K., Burchardt, A., Rehm, G., and Specia, L. (2016). Technology Landscape for Quality Evaluation: Combining the Needs of Research and Industry. In *Proceedings of the LREC 2016 Workshop on Translation Evaluation: From Fragmented Tools and Data Sets to an Integrated Ecosystem*, Portoroz, Slovenia. European Language Resources Association.

Hartung, J., Knapp, G., and Sinha, B. K. (2008). *Statistical Meta-Analysis with Applications*. Wiley.

He, Y., Ma, Y., Genabith, J. V., Way, A., van Genabith, J., and Way, A. (2010). Bridging SMT and TM with translation recommendation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 622–630, Uppsala, Sweden. Association for Computational Linguistics.

Heafield, K. (2011). KenLM : Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.

Hearst, M. A., Dumais, S. T., Osman, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28.

Herbrich, R., Graepel, T., and Obermayer, K. (1999). Support Vector Learning for Ordinal Regression. In *Ninth International Conference on Artificial Neural Networks*, pages 97–102, Edinburgh, UK.

Hildebrand, S. and Vogel, S. (2013). MT Quality Estimation: The CMU System for WMT'13. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 373–379, Sofia, Bulgaria. Association for Computational Linguistics.

Hopkins, M. and May, J. (2011). Tuning as ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland. Association for Computational Linguistics Morristown, NJ, USA.

Hosmer, D. (1989). *Applied logistic regression*. Wiley, New York [u.a.], 8th edition.

Huang, L. and Chiang, D. (2007). Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, volume 45, page 144, Prague, Czech Republic. Association for Computational Linguistics.

Hüllermeier, E., Fürnkranz, J., Cheng, W., and Brinker, K. (2008). Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16-17):1897–1916.

Hunt, E., Martin, J., and Stone, P. (1966). Experiments in Induction. Academic Press, New York.

Hutchins, J. (2003). Machine translation and computer-based translation tools: What's available and how it's used. *A New Spectrum of Translation Studies*.

Hutchins, J. and Somers, H. (1992). *An Introduction to Machine Translation*. Academic Press, Cambridge.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446.

Junczys-Dowmunt, M., Dwojak, T., and Hoang, H. (2016). Is Neural Machine Translation Ready for Deployment? A Case Study on 30 Translation Directions. *Computer Research Repository*, abs/1610.0.

Kaki, S., Yamada, S., and Sumita, E. (1999). Scoring multiple translations using character N-gram. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS'99)*, Beijing, China.

Kaljahi, Z., Samad, R., Foster, J., Rubino, R., Roturier, J., and Hollowood, F. (2013). Parser Accuracy in Quality Estimation of Machine Translation: A Tree Kernel Approach. *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1092–1096.

Kendall, M. G. (1938). A New Measure of Rank Correlation. *Biometrika*, 30(1-2):81–93.

Khalilov, M. and Fonollosa, J. A. R. (2009). N-gram-based statistical machine translation versus syntax augmented machine translation: comparison and system combination. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computation Linguistic*, pages 424–432, Athens, Greece. Association for Computational Linguistics.

Khedr, A. M. (2008). Learning k-Nearest Neighbors Classifier from Distributed Data. *Computing and Informatics*, 27(3):355–376.

Kim, H., Lee, J.-H., and Na, S.-H. (2017). Predictor-Estimator using Multilevel Task Learning with Stack Propagation for Neural Quality Estimation. In *Proceedings of the Second Conference on Machine Translation*, volume 2, pages 562–568, Copenhagen, Denmark. Association for Computational Linguistics.

Klejch, O., Avramidis, E., Burchardt, A., and Popel, M. (2015). MT-ComparEval: Graphical evaluation interface for Machine Translation development. *The Prague Bulletin of Mathematical Linguistics*, 104(1):63–74.

Knight, W. R. (1966). A computer method for calculating Kendalls tau with ungrouped data. *Journal of the American Statistical Association*, 61(314):436–439.

Koby, G. S., Fields, P., Hague, D., Lommel, A., and Melby, A. (2014). Defining Translation Quality. *Revista Tradumàtica*, pages 413–420.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the tenth Machine Translation Summit*, volume 5, pages 79–86, Phuket, Thailand.

Koehn, P. (2010). An Experimental Management System. *The Prague Bulletin of Mathematical Linguistics*, 94:87–96.

Koehn, P., Arun, A., and Hoang, H. (2008). Towards better Machine Translation Quality for the German-English Language Pairs. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 139–142, Columbus, Ohio. Association for Computational Linguistics.

Koehn, P., Birch, A., and Steinberger, R. (2009). 462 Machine Translation Systems for Europe. In *Proceedings of the twelfth Machine Translation Summit*, pages 65–72, Ontario, Canada.

Koehn, P. and Hoang, H. (2007). Factored Translation Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876. Association for Computational Linguistics.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens Richard a nd 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*, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Koehn, P. and Knight, K. (2003). Empirical Methods for Compound Splitting. In *Proceedings of the 10th Annual Conference of the European Association for Machine Translation*, volume 1, page 8, Budapest, Hungary. European Association for Machine Translation.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54.

Koehn, P., Shen, W., Federico, M., Bertoldi, N., Callison-Burch, C., Cowan, B., Dyer, C., Hoang, H., Bojar, O., Zens, R., Constantin, A., Herbst, E., and Moran, C. (2006). Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Kononenko, I. (1994). Estimating attributes: analysis and extensions of RELIEF. In *Proceedings of the European conference on machine learning on Machine Learning*, pages 171–182, Secaucus, NJ, USA. Springer-Verlag New York, Inc.

Kübler, S. (2008). The PaGe 2008 Shared Task on Parsing German. In *Proceedings of the Workshop on Parsing German*, pages 55–63, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lapata, M., Court, R., Street, P., and Sheffield, S. (2003). Probabilistic Text Structuring : Experiments with Sentence Ordering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, number July, pages 545–552, Sapporo, Japan. Association for Computational Linguistics.

Lavie, A. and Agarwal, A. (2007). METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic. Association for Computational Linguistics.

Levenshtein, V. (1966). Binary Codes Capable of Correcting Deletions and Insertions and Reversals. *Soviet Physics Doklady*, 10(8):707–710.

Li, H., Kumaran, A., Pervouchine, V., and Zhang, M. (2009). Report of NEWS 2009 Machine Transliteration Shared Task. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 1–18, Suntec, Singapore. Association for Computational Linguistics.

Lin, K. H.-Y., Yang, C., and Chen, H.-H. (2007). What emotions do news articles trigger in their readers? In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '07*, page 733, New York, New York, USA. ACM Press.

Lita, L. V., Ittycheriah, A., Roukos, S., and Kambhatla, N. (2003). tRuEcasIng. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 152–159, Sapporo, Japan. Association for Computational Linguistics.

Logacheva, V., Blain, F., and Specia, L. (2016a). USFD's Phrase-level Quality Estimation Systems. In *Proceedings of the First Conference on Machine Translation*, pages 800–805, Berlin, Germany. Association for Computational Linguistics.

Logacheva, V., Hokamp, C., and Specia, L. (2015). Data enhancement and selection strategies for the word-level Quality Estimation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 330–335, Lisbon, Portugal. Association for Computational Linguistics.

Logacheva, V., Hokamp, C., and Specia, L. (2016b). MARMOT: A Toolkit for Translation Quality Estimation at the Word Level. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 16)*, Paris, France. European Language Resources Association.

Lommel, A., Burchardt, A., Popović, M., Harris, K., Avramidis, E., and Uszkoreit, H. (2014). Using a new analytic measure for the annotation and analysis of MT errors on real data. In *Proceedings of the 17th Annual Conference of the European Association for Machine Translation*, pages 165–172. Croatian Language Technologies Society, European Association for Machine Translation.

Loper, E. and Bird, S. (2002). NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, volume 1, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics.

Machacek, M. and Bojar, O. (2014). Results of the WMT14 Metrics Shared Task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 293–301, Baltimore, Maryland, USA. Association for Computational Linguistics.

Macketanz, V., Avramidis, E., Burchardt, A., Helcl, J., and Srivastava, A. (2017). Machine Translation: Phrase-Based, Rule-Based and Neural Approaches with Linguistic Evaluation. *Cybernetics and Information Technologies (cait)*, 17(2):28–43.

Madnani, N. (2009). Source Code: Querying and Serving N-gram Language Models with Python. *The Python Papers Source Codes*.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Martins, A., Astudillo, R., Hokamp, C., and Kepler, F. (2016). Unbabel's Participation in the WMT16 Word-Level Translation Quality Estimation Shared Task. In *Proceedings of the First Conference on Machine Translation*, pages 806–811, Berlin, Germany. Association for Computational Linguistics.

Martins, A., Kepler, F., and Monteiro, J. (2017). Unbabel's Participation in the WMT17 Translation Quality Estimation Shared Task. In *Proceedings of the Second Conference on Machine Translation*, volume 2, pages 569–574, Copenhagen, Denmark. Association for Computational Linguistics.

McLachlan, G. (1992). *Discriminant Analysis and Statistical Pattern Recognition*. Wiley.

Melamed, I. D. (1997). A Word-to-Word Model of Translational Equivalence. In *Proceedings of the 35th Annual Conference\ of the Association for Computational Linguistics (ACL 97)*, Madrid, Spain.

Menard, S. (2002). *Applied logistic regression analysis*, volume 106. Sage.

Miłkowski, M. (2012). Translation Quality Checking in LanguageTool. In Pęzik, P., editor, *Corpus Data across Languages and Disciplines*, Corpus Data across Languages and Disciplines, pages 213–223. Peter Lang, Frankfurt am Main, Berlin, Bern, Bruxelles, New York, Oxford, Wien.

Miller, A. (2002). *Subset Selection in Regression*. Chapman & Hall, London, 2nd edition.

Moreau, E. and Rubino, R. (2013). An Approach Using Style Classification Features for Quality Estimation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 429–434, Sofia, Bulgaria. Association for Computational Linguistics.

Mutton, A., Dras, M., Wan, S., and Dale, R. (2007). GLEU: Automatic Evaluation of Sentence-Level Fluency. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 344–351.

Naber, D. (2003). A rule-based style and grammar checker. Technical report, Bielefeld University, Bielefeld, Germany.

Nießen, S., Och, F. J., Leusch, G., and Ney, H. (2000). An evaluation tool for machine translation: Fast evaluation for MT research. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, pages 39–45, Athens, Greece. European Language Resources Association.

Nitta, Y. (1986). Idiosyncratic gap: A tough problem to structure-bound Machine Translation. In *Proceedings of the 11th coference on Computational linguistics*, page 107, Morristown, NJ, USA. Association for Computational Linguistics.

Nomoto, T. (2003). Predictive Models of Performance in Multi-Engine Machine Translation. In *Proceedings of the tenth Machine Translation Summit*, pages 269–276, Phuket, Thailand. International Association for Machine Translation.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167, Morristown, NJ, USA. Association for Computational Linguistics.

Och, F. J., Gildea, D., Khudanpur, S., Sarkar, A., Yamada, K., Fraser, A., Kumar, S., Shen, L., Smith, D., Eng, K., Jain, V., Jin, Z., and Radev, D. (2004). A Smorgasbord of Features for Statistical Machine Translation. In *HLT-NAACL 2004: Main Proceedings*, pages 161–168, Boston, Massachusetts, USA. Association for Computational Linguistics.

Och, F. J., Gildea, D., Khudanpur, S., Sarkar, A., Yamada, K., Fraser, A., Kumar, S., Shen, L., Smith, D., Eng, K., and Others (2003). Syntax for statistical machine translation. Technical report, Center for Language and Speech Processing, Baltimore, Maryland, USA.

Och, F. J. and Ney, H. (2003). A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

Och, F. J., Tillmann, C., and Ney, H. (1999). Improved alignment models for statistical machine translation. *Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28.

Oliphant, T. E. (2007). Python for Scientific Computing. *Computing in Science & Engineering*, 9(3):10–20.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2001). BLEU: a Method for Automatic Evaluation of Machine Translation. IBM Research Report RC22176(W0109-022), IBM.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Parton, K., Tetreault, J., Madnani, N., and Chodorow, M. (2011). E-rating Machine Translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 108–115, Edinburgh, Scotland. Association for Computational Linguistics.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia. Association for Computational Linguistics.

Petrov, S. and Klein, D. (2007). Improved inference for unlexicalized parsing. In *Proceedings of the 2007 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 404–411, Rochester, New York, USA. Association for Computational Linguistics.

Petrov, S. and Klein, D. (2008). Parsing German with Latent Variable Grammars. In *Proceedings of the Workshop on Parsing German*, pages 33–39, Stroudsburg, PA, USA. Association for Computational Linguistics.

Pighin, D., González, M., and Màrquez, L. (2012). The UPC Submission to the WMT 2012 Shared Task on Quality Estimation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 127–132, Montréal, Canada. Association for Computational Linguistics.

Pighin, D. and Màrquez, L. (2011). Automatic Projection of Semantic Structures: an Application to Pairwise Translation Ranking. In *Proceedings of Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-5)*, number June, pages 1–9, Portland, Oregon , USA. Association for Computational Linguistics.

Pohar, M., Blas, M., and Turk, S. (2004). Comparison of Logistic Regression and Linear Discriminant Analysis: A Simulation Study. *Metodološki zvezki*, 1(1):143–161.

Pointal, L. (2015). TreeTagger Wrapper. Technical report.

Popović, M. (2011a). Evaluation without references: IBM 1 scores as evaluation metrics. *Computational Linguistics*, (1993):99–103.

Popović, M. (2011b). From human to automatic error classification for machine translation output. In *Proceedings of the 15th International Conference of the European Association for Machine Translation*, pages 265–272, Leuven, Belgium. European Association for Machine Translation.

Popović, M. (2011c). Hjerson: An Open Source Tool for Automatic Error Classification of Machine Translation Output. *The Prague Bulletin of Mathematical Linguistics*, 96(-1):59–68.

Popović, M. (2012a). Morpheme- and POS -based IBM 1 scores and language model scores for translation quality estimation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 133–137, Montréal, Canada.

Popović, M. (2012b). rgbF: An Open Source Tool for n-gram Based Automatic Evaluation of Machine Translation Output. *The Prague Bulletin of Mathematical Linguistics*, 98(98):99–108.

Popović, M., Avramidis, E., Burchardt, A., Vilar, D., and Uszkoreit, H. (2013). What can we learn about the selection mechanism for post-editing? In *Proceedings of the fourteenth Machine Translation Summit*, pages 11–17, Nice, France. International Association for Machine Translation.

Popović, M., Lommel, A., Burchardt, A., Avramidis, E., and Uszkoreit, H. (2014). Relations between different types of post-editing operations, cognitive effort and temporal effort. In *Proceedings of the 17th Annual Conference of the European Association for Machine Translation*, pages 191–198, Dubrovnik, Croatia. European Association for Machine Translation.

Popović, M. and Ney, H. (2011). Towards Automatic Error Analysis of Machine Translation Output. *Computational Linguistics*, 37(4).

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, 1(1):81–106.

Quirk, C. (2004). Training a Sentence-Level Machine Translation Confidence Measure. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, volume 4, pages 825–828, Lisbon, Portugal. European Language Resources Association.

Radev, D., Qi, H., Wu, H., and Fan, W. (2002). Evaluating web-based question answering systems. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, Las Palmas, Spain. European Language Resources Association.

Robnik-Šikonja, M. and Kononenko, I. (2003). Theoretical and Empirical Analysis of ReliefF and RReliefF. *Machine Learning*, 53(1-2):23–69.

Rückstieß, T. and Schmidhuber, J. (2011). A Python Experiment Suite. *The Python Papers*, 6(1):2.

Russell, B. and Gillespie, D. (2016). Measuring the behavioral impact of machine translation quality improvements with A/B testing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2295–2299, Austin, Texas. Association for Computational Linguistics.

Ryan, J. P. (1989). SYSTRAN: A Machine Translation System to Meet User Needs. In *Proceedings of the Machine Translation Summit*, page 116, Tokyo, Japan. IOS Press.

Sánchez-Martínez, F. (2011). Choosing the best machine translation system to translate a sentence by using only source-language information. In *Proceedings of the 15th Annual Conference of the European Association for Machine Translation*, pages 97–104, Leuve, Belgium. European Association for Machine Translation.

Scarton, C. and Specia, L. (2014a). Document-level translation quality estimation: exploring discourse and pseudo-references. In *Proceedings of the17th International Conference of the European Association for Machine Translation*, pages 101 – 108, Dubrovnik, Croatia. European Association for Machine Translation.

Scarton, C. and Specia, L. (2014b). Exploring Consensus in Machine Translation for Quality Estimation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 342–347, Baltimore, Maryland, USA. Association for Computational Linguistics.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK. UCL Press.

Schmid, H. (2004). Efficient Parsing of Highly Ambiguous Context-Free Grammars with Bit Vectors. In *Proceedings of the 20th international conference on Computational Linguistics*, pages 162–169, Stroudsburg, PA, USA. Association for Computational Linguistics.

Schmid, H. (2006). Trace Prediction and Recovery with Unlexicalized PCFGs and Slash Features. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 177–184, Stroudsburg, PA, USA. Association for Computational Linguistics.

Schwenk, H. and Koehn, P. (2008). Large and diverse language models for statistical machine translation. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, pages 661–666, Hyderabad, India. Asian Federation of Natural Language Processing.

Servan, C., Le, N.-T., Luong, N. Q., Lecouteux, B., and Besacier, L. (2015). An Open Source Toolkit for Word-level Confidence Estimation in Machine Translation. In *Proceedings of the 12th International Workshop on Spoken Language Translation (IWSLT'15)*, Da Nang, Vietnam.

Shah, K., Avramidis, E., Biçici, E., and Specia, L. (2013). QuEst: Design, Implementation and Extensions of a Framework for Machine Translation Quality Estimation. *The Prague Bulletin of Mathematical Linguistics*, 100:19–30.

Shah, K. and Specia, L. (2014). Quality estimation for translation selection. In *Proceedings of the 17th Annual Conference of the European Association for Machine Translation*, pages 109–116, Dubrovnik, Croatia. Citeseer, European Association for Machine Translation.

Siegel, M. (2011). Autorenunterstützung für die Maschinelle Übersetzung. In *Multilingual Resources and Multilingual Applications: Proceedings of the Conference of the German Society for Computational Linguistics and Language Technology (GSCL)*, Hamburg.

Simard, M., Ueffing, N., Isabelle, P., and Kuhn, R. (2007). Rule-Based Translation with Statistical Phrase-Based Post-Editing. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 203–206, Prague, Czech Republic. Association for Computational Linguistics.

Smith, J. R., Saint-Amand, H., Plamada, M., Koehn, P., Callison-Burch, C., and Lopez, A. (2013). Dirt Cheap Web-Scale Parallel Text from the Common Crawl. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1374–1383, Sofia, Bulgaria. Association for Computational Linguistics.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., Makhoul, J., Weischedel, R., Makhoul, J., and Weischedel, R. (2006). A Study of Translation Error Rate with Targeted Human Annotation. In *Proceedings of the 7th biennial conference of the Association for Machine Translation in the Americas*, pages 223–231, Cambridge, MA, USA. International Association for Machine Translation.

Snow, R., O'Connor, B., Jurafsky, D., Ng, A. Y., Connor, B. O., Jurafsky, D., Ng, A. Y., Labs, D., and St, C. (2008). Cheap and Fast — But is it Good ? Evaluating Non-Expert Annotations for Natural Language Tasks. *Computational Linguistics*, (October):254–263.

Soricut, R., Bach, N., and Wang, Z. (2012). The SDL Language Weaver Systems in the WMT12 Quality Estimation Shared Task. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 145–151, Montréal, Canada. Association for Computational Linguistics.

Soricut, R. and Echihabi, A. (2010). TrustRank: Inducing Trust in Automatic Translations via Ranking. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 612–621, Uppsala, Sweden. Association for Computational Linguistics.

Specia, L. (2010). Combining Confidence Estimation and Reference-based Metrics for Segment-level MT Evaluation. In *The Ninth Conference of the Association for Machine Translation in the Americas*. Association for Machine Translation in the Americas.

Specia, L. (2011). Exploiting Objective Annotations for Measuring Translation Post-editing Effort. *Machine Translation*, (May).

Specia, L., Cancedda, N., and Dymetman, M. (2010a). A Dataset for Assessing Machine Translation Evaluation Metrics. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, pages 3375–3378, Malta. European Language Resources Association.

Specia, L. and Farzindar, A. (2010). Estimating Machine Translation Post-Editing Effort with HTER. In *Proceedings of the AMTA2010 Workshop for Bringing MT to the User - MT Research and the Translation Industry*, pages 33–41. Association for Machine Translation in the Americas.

Specia, L., Paetzold, G., and Scarton, C. (2015). Multi-level Translation Quality Prediction with QuEst++. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 115–120, Beijing, China. Association for Computational Linguistics and The Asian Federation of Natural Language Processing.

Specia, L., Raj, D., and Turchi, M. (2010b). Machine translation evaluation versus quality estimation. *Machine Translation*, 24(1):39–50.

Specia, L., Shah, K., de Souza, J. G. C., and Cohn, T. (2013). QuEst - A translation quality estimation framework. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 79–84, Sofia, Bulgaria. Association for Computational Linguistics.

Specia, L., Turchi, M., Cancedda, N., Dymetman, M., and Cristianini, N. (2009a). Estimating the Sentence-Level Quality of Machine Translation Systems. In *Proceedings of the 13th Annual Conference of the European Association for Machine Translation*, pages 28–35, Barcelona, Spain. European Association for Machine Translation.

Specia, L., Turchi, M., Wang, Z., Shawe-Taylor, J., and Saunders, C. (2009b). Improving the Confidence of Machine Translation Quality Estimates. In *Proceedings of the Machine Translation Summit XII*, Ottawa, Canada. International Association for Machine Translation.

Srivastava, A., Macketanz, V., Burchardt, A., and Avramidis, E. (2016). Towards Deeper MT: Parallel Treebanks, Entity Linking, and Linguistic Evaluation. In *Proceedings of the Workshop on Deep Language Processing for Quality Machine Translation (DeepLP4QMT)*, page 34, Sofia, Bulgaria. Institute of Information and Communication Technologies, Bulgarian Academy of Sciences.

Stanojevic, M. and Sima'an, K. (2014). BEER: BEtter Evaluation as Ranking. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 414–419, Baltimore, Maryland, USA. Association for Computational Linguistics.

Stolcke, A. (2002). SRILM — an Extensible Language Modeling Toolkit. In *7th International Conference on Spoken Language Processing (INTERSPEECH 2002)*, volume 2, pages 901–904, Denver, Colorado. International Speech Communication Association.

Sumita, E., Akiba, Y., and Imamura, K. (2002). Reliability measures for translation quality. In *7th International Conference on Spoken Language Processing (INTERSPEECH 2002)*, Denver, Colorado, USA. International Speech Communication Association.

Tamchyna, A., Dušek, O., Rosa, R., and Pecina, P. (2013). MTMonkey: A Scalable Infrastructure for a Machine Translation Web Service. *The Prague Bulletin of Mathematical Linguistics*, 100:31–40.

Taulé, M., Martí, A., and Recasens, M. (2008). AnCora: Multilevel Annotated Corpora for Catalan and Spanish. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, pages 96–101, Marrakech, Morocco. European Language Resources Association.

Telljohann, H., Hinrichs, E., Kübler, S., Kübler, R., and Tübingen, U. (2004). The Tüba-D/Z Treebank: Annotating German with a Context-Free Backbone. In *In Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, pages 2229–2235, Lisbon, Portugal. European Language Resources Association.

Tidhar, D. and Küssner, U. (2000). Learning to Select a Good Translation. In *Proceedings of the 18th Conference on Computational Linguistics*, volume 2, pages 843–849, Saarbrücken, Germany. Association for Computational Linguistics.

Ueffing, N. and Ney, H. (2005). Application of word-level confidence measures in interactive statistical machine translation. In *Proceedings of the 10th EAMT Conference on Practical applications of machine translation*, pages 262–270, Budapest, Hungary. European Association for Machine Translation.

van Cranenburgh, A. (2010). Enriching Data-Oriented Parsing by blending morphology and syntax. Technical report, University of Amsterdam, Amsterdam.

Van Der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, 13(2):22–30.

Vilar, D., Xu, J., D'Haro, L. F., and Ney, H. (2006). Error Analysis of Machine Translation Output. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC'06)*, pages 697–702, Genoa, Italy. European Language Resources Association.

Vogel, S., Och, F., Tillmann, C., Nießen, S., Sawaf, H., and Ney, H. (2000a). Statistical Methods for Machine Translation. In Wahlster, W., editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 377–393. Springer Verlag: Berlin, Heidelberg, New York.

Vogel, S., Och, F. J., and Ney, H. (2000b). The Statistical Translation Module in the Verbmobil System. In *Proceedings of the 5th Conference on Natural Language Processing (KONVENS-2000)*, pages 291–293, Ilmenau, Germany. VDE Verlag.

Wagner, J. and Foster, J. (2009). The effect of correcting grammatical errors on parse probabilities. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 176–179, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wang, W., Knight, K., and Marcu, D. (2006). Capitalizing Machine Translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 1–8, New York. Association for Computational Linguistics.

Wang, Y., Wang, L., Li, Y., He, D., Liu, T.-Y., and Chen, W. (2013). A Theoretical Analysis of NDCG Type Ranking Measures. In *Proceedings of the 26th Annual Conference on Learning Theory*, pages 1–30, Princeton, NJ, USA. Proceedings of Machine Learning Research.

Weissenborn, D., Hennig, L., Xu, F., and Uszkoreit, H. (2015). Multi-Objective Optimization for the Joint Disambiguation of Nouns and Named Entities. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International*

Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, pages 596–605, Beijing, China. Association for Computer Linguistics.

Wong, S.-M. J. and Dras, M. (2010). Parser features for sentence grammaticality classification. In *Proceedings of the 2010 Australasian Language Technology Association Workshop*, pages 67–75, Melbourne, Australia. Australasian Language Technology Association.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *Computer Research Repository*, abs/1609.0.

Yasuda, K., Sugaya, F., Takezawa, T., Yamamoto, S., and Yanagida, M. (2002). Automatic machine translation selection scheme to output the best result. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, pages 525–528, Las Palmas, Spain. European Language Resources Association.

Ye, Y., Zhou, M., and Lin, C.-Y. (2007). Sentence level machine translation evaluation as a ranking problem: one step aside from BLEU. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 240–247, Prague, Czech Republic. Association for Computational Linguistics.

Zens, R., Ney, H., Watanabe, T., and Sumita, E. (2004). Reordering Constraints for Phrase-Based Statistical Machine Translation. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 205–211, Geneva, Switzerland. Association for Computational Linguistics.

Zhang, M., Jiang, H., Aw, A., Sun, J., Li, S., and Tan, C. L. (2007). A tree-to-tree alignment-based model for statistical machine translation. In *Proceedings of the Ninth Machine Translation Summit*, pages 535–542, Copenhagen, Denmark. International Association for Machine Translation.

Zwarts, S. and Dras, M. (2008). Choosing the Right Translation: A Syntactically Informed Classification Approach. In *Proceedings of the 22nd International Conference on Computational Linguistics*, volume 1, pages 1153–1160, Stroudsburg, PA, USA. Association for Computational Linguistics.