D3.2: Report on Confidence Measures

Jesús González-Rubio, Philipp Koehn

Distribution: Public

CASMACAT
Cognitive Analysis and Statistical Methods for Advanced Computer Aided Translation

ICT Project 287576 Deliverable D3.2

Project funded by the European Community under the Seventh Framework Programme for Research and Technological Development.
<table>
<thead>
<tr>
<th>Project ref no.</th>
<th>ICT-287576</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project acronym</td>
<td>CASMACAT</td>
</tr>
<tr>
<td>Project full title</td>
<td>Cognitive Analysis and Statistical Methods for Advanced Computer Aided Translation</td>
</tr>
<tr>
<td>Instrument</td>
<td>STREP</td>
</tr>
<tr>
<td>Thematic Priority</td>
<td>ICT-2011.4.2 Language Technologies</td>
</tr>
<tr>
<td>Start date / duration</td>
<td>01 November 2011 / 36 Months</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contractual date of delivery</td>
<td>October 31, 2013</td>
</tr>
<tr>
<td>Actual date of delivery</td>
<td>November 4, 2013</td>
</tr>
<tr>
<td>Date of last update</td>
<td>November 4, 2013</td>
</tr>
<tr>
<td>Deliverable number</td>
<td>D3.2</td>
</tr>
<tr>
<td>Deliverable title</td>
<td>Report on Confidence Measures</td>
</tr>
<tr>
<td>Type</td>
<td>Report</td>
</tr>
<tr>
<td>Status &amp; version</td>
<td>Draft</td>
</tr>
<tr>
<td>Number of pages</td>
<td>64</td>
</tr>
<tr>
<td>Contributing WP(s)</td>
<td>WP7</td>
</tr>
<tr>
<td>WP / Task responsible</td>
<td>UEDIN, UPVLC, CBS, CS</td>
</tr>
<tr>
<td>Other contributors</td>
<td>Herve Saint-Amand, Turan Rustamli</td>
</tr>
<tr>
<td>Internal reviewer</td>
<td>Robin Hill</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Jesús González-Rubio, Philipp Koehn</td>
</tr>
<tr>
<td>EC project officer</td>
<td>Kimmo Rossi</td>
</tr>
</tbody>
</table>

The partners in CASMACAT are:
- University of Edinburgh (UEDIN)
- Copenhagen Business School (CBS)
- Universitat Politècnica de València (UPVLC)
- Celer Soluciones (CS)

For copies of reports, updates on project activities and other CASMACAT related information, contact:

The CASMACAT Project Co-ordinator
Philipp Koehn, University of Edinburgh
10 Crichton Street, Edinburgh, EH8 9AB, United Kingdom
pkoehn@inf.ed.ac.uk
Phone +44 (131) 650-8287 - Fax +44 (131) 650-6626

Copies of reports and other material can also be accessed via the project’s homepage:
http://www.casmacat.eu/

© 2013, The Individual Authors
No part of this document may be reproduced or transmitted in any form, or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission from the copyright owner.
Executive Summary

Contents

1 Overview 4

2 Organization of the Shared Task on Quality Estimation 4

3 Sentence-level Estimate of Post-editing Effort (Task 3.1) 4
   3.1 Introduction 4
   3.2 Development 5
   3.3 Conclusions 5

4 Word-level Confidence Measures (Task 3.2) 6
   4.1 Introduction 6
   4.2 Development 6
   4.3 Conclusions 6

5 Display Multiple Translation Options (Task 3.5) 7
   5.1 Translation Option Array 7
   5.2 Translation Options in Context 8

6 Authoring Assistance (Task 3.6) 10
   6.1 Introduction 10
   6.2 Requirements 10
   6.3 Generation of Paraphrases 12
   6.4 Ranking of Paraphrases 12
      6.4.1 Partial Filters 13
      6.4.2 Score Functions 13
      6.4.3 Filters 13
      6.4.4 Sorters 14
   6.5 Evaluation 14
      6.5.1 Automatic evaluation 14
      6.5.2 Manual Evaluation 16

Attachment A 21
Attachment B 45
Attachment C 55
1 Overview

This deliverable reports on the work carried out in Workpackage 3, which aims at the development of new methods to assist the editing of translations. Such methods will assist human translators to improve machine translation output.

In the reporting period (month 13-24), work was carried out on the following task, as planned:

- Task 3.1: Sentence-level Estimate of Post-editing Work Effort (completed)
- Task 3.2: Word-level Confidence Measures (completed)
- Task 3.5: Display Multiple Translation Options (ongoing into year 3)
- Task 3.6: Authoring Assistance (ongoing into year 3)

2 Organization of the Shared Task on Quality Estimation

The objectives of Task 3.1 and 3.2 have received wider attention in the research community, including the EU-funded research projects MATECAT (our "sister" CAT project) and QT-Launchpad (a research project aimed at quality translation). To collaborate community-wide on the problem of confidence estimation (now called more commonly quality estimation), the three projects organized a shared task at the Workshop on Statistical Machine Translation (WMT), which takes place under the umbrella of the Annual Meeting of the Association for Computation Linguistics (ACL).

Besides participation by submitting systems, our contribution to the organization of the shared task was the provision of training and testing data and contribution to the design of the task definition.

The first field trial of the casmacat workbench in the first year of the project generated a unique data resource that enabled more grounded work on quality estimation. Since we developed the machine translation system that was used in the field trial, we have complete transparency of its models and parameters, enabling the use of so-called "glass box" features. In the field trial, output from the machine translation system was post-edited, giving us information on the post-editing effort, both in terms of edit operations as well as post-editing time. Given this data, we are now able to pose the problem of quality estimation as the task to estimate post-editing time (related to our Task 3.1) and the task to predict which words have to be edited (our Task 3.2).

The shared task is described in detail in the workshop report (Bojar et al., 2013). 14 research teams participated in the shared tasks,

3 Sentence-level Estimate of Post-editing Effort (Task 3.1)

3.1 Introduction

The goal of this task is to develop techniques to estimate the amount of effort required to fix automatic translations. The work for this task is based on our earlier research in sentence-level confidence measures (González-Rubio et al., 2010; 2011; 2012).
3.2 Development

Sentence-level quality estimation (QE) is usually addressed as a regression problem (Blatz et al., 2004; Specia et al., 2009). Given a translation (and other possible sources of information), a set of features is extracted and used to build a model that predicts a quality score. This point of view provides a solid framework within which accurate predictors can be derived. However, several problems arise when applying this approach to predict the quality of natural language sentences. For example, while the concept of translation quality is quite intuitive, the definition of features that reliably account for it has proven to be elusive (Blatz et al., 2004; Callison-Burch et al., 2012). Thus, in practice, feature sets contain a large number of noisy, collinear and ambiguous features that hinder the learning process of the regression models, e.g., due to the “curse of dimensionality” (Bellman, 1961).

In (González-Rubio et al., 2013a), see Attachment A, we conceive sentence-level QE as a two-step problem. In a first step, a dimensionality reduction (DR) method strips out the noise present in the original features returning a reduced set of (potentially new) features. Then, the actual quality prediction is made from this reduced set. Typically, QE systems reduce the dimensionality by simply selecting a subset of the original features according to some relevance measure (Specia et al., 2009; Langlois et al., 2012; Soricut et al., 2012). However, DR methods based on a projection of the original features may be more effective. The intuition for this is clear, the new features extracted by a projection-based DR method summarize the “information” contained in the all the original features, in contrast, the information contained in the features discarded by a feature selection method is inevitably lost. We propose two novel DR methods based on on partial least squares regression (PLSR) (Wold, 1966): a DR method that selects a subset of the original features, and a method that projects the original data into a space of fewer dimensions. In the experiments, we compare their performance against different DR methods previously proposed in the literature. Moreover, we also study how the use of these DR methods influences the performance of different machine learning models.

In Attachment Attachment B we work on the foundations of González-Rubio et al. (2013a) and provide an exhaustive empirical study of the most successful QE approach described there. This approach involves a DR method based on a PLS projection of the data and an SVM as prediction model. We test this two-step QE approach in a wide variety of conditions where we compare the performance of PLS to the most widely-used projection-based DR approach, namely principal component analysis (PCA) (Pearson, 1901).

3.3 Conclusions

The key results of the experiments on DR methods González-Rubio et al. (2013a) are as follows:

- Methods based on a projection of the data are usually more effective than feature selection methods.
- Methods based on PLS are usually more effective than other methods.
- The performance-wise ranking of DR methods is to a great extent independent of the chosen learning model.
- A combination of PLS projection and SVM was the best performing setup in the experiments.

The empirical cross-validation results showed that the studied such best setup (PLS+SVM) was able to obtain very large feature reduction ratios, and at the same time, it usually outperformed systems built with all the original features and systems that use PCA instead of
PLS to reduce the dimensionality. Unfortunately, results in the held-out test partitions were disappointing. The results of different statistical tests seem to indicate that this was due to the small size of the training partitions. Hence, larger improvements in performance could be expected in test whenever a representative training partition is provided.

A complimentary advantage of the studied two-step QE approach is its time-efficiency which makes our approach well-suited to be deployed in scenarios with strict temporal restrictions. Moreover, we are working towards further minimizing response time with the goal of applying our approach even in real-time scenarios such as ITP. Alternatively, we could take advantage of this efficiency to predict translation quality from huge sets of features. Results in this direction show that our approach was able to efficiently manage more than a thousand features largely improving prediction accuracy.

4 Word-level Confidence Measures (Task 3.2)

4.1 Introduction

Word-level confidence measures (CM) are usually formalized as a conventional pattern classification problem in which a feature vector is obtained for each word in order to classify it as either correct or incorrect (Ueffing et al., 2003; Sanchis et al., 2007). This point of view provides a solid, well-known framework, within which accurate two-class classifiers can be derived. The challenges of this approach are to find an appropriate set of features, and to learn accurate classification models.

4.2 Development

In (González-Rubio et al., 2013c), see Attachment C, we extend previous word-level confidence approaches in several aspects, including the addition of new features, and the use of a novel classification model based on multidimensional statistical analysis. As Sanchis et al. (2007), we also compute prediction features based on posterior probabilities. However, we generalize this approach to take into account the context of each word. The key idea is that the reliability of a word is influenced by the context in which it appears, therefore by using context-aware features we expect to obtain a stronger estimation of each word reliability. Additionally, we propose a new classifier based on the partial least squares discriminant analysis (PLS-DA) (Wold, 1966). This classifier performs an intrinsic transformation of the features such that a maximum separation among classes is obtained. Thus it is an effective and efficient method that allows us to build robust classifiers even for ambiguous and redundant features such as those found in natural language processing.

4.3 Conclusions

Our results showed that features based on the Model 1 lexicon achieve the best performance, followed by those computed based on posterior probabilities from an N-best list of translations. One of the features based on the Model 1 lexicon is the one chosen to implement active interaction in the CASMACAT workbench, see task 3 in workpackage 2. Also, the use of context information did not improve the results of the previously used Ueffing et al. (2003) Sanchis et al. (2007) N-best-based features. Regarding the classification models, the proposed PLS-DA model consistently outperformed the smoothed naïve Bayes model proposed in Sanchis et al. (2007) in all test conditions. In fact, PLS-DA had shown to be an effective, scalable, and robust classification model quite adequate for the task.
5 Display Multiple Translation Options (Task 3.5)

Task 3.5 extends over the entire project period (month 1–36), and is concerned with the provision of translation options to the translators. Instead of the traditional approach of giving the translator a single machine translation to post-edit, statistical machine translation can provide multiple ranked examples in context.

In year 2, we developed some preliminary plans for the display of multiple translation options compactly to the user (Section 5.1) and implemented a method to show translation options in context (Section 5.2).

5.1 Translation Option Array

A common problem of translation is to find the right words to express the meaning of the original. The idea behind showing a number of possible translations to the translators is based on the big difference between active and passive vocabulary: speakers of a language can recognize a much larger number of words than they are able to recall when needed.

Below an example for the general idea:

| er | hat | seit | Monaten | geplant | , | im | März | einen | Vortrag | ...
|---|---|---|---|---|---|---|---|---|---|---
| he has | for months | the plan | in March | a lecture | ...
| it has | for months now | planned | in | March | a presentation | ...
| he was | for several months | planned to | in the March | a speech | ...
| he has made | since months | the pipeline | in March of | a statement | ...
| he did | for many months | scheduled | the March | a general | ...

Given the German input sentence "er hat seit Monaten geplant, im März einen Vortrag ..." up to five translation options are shown for each input word. These translation options are a subset of what the statistical machine translation model contains.

Given the source sentence, we plan to display an array of possible word and phrase translations. This idea was already explored in the Caitra translator tool. There are two conflicting problems that need to be addressed to make this work. First, we need to make efficient use of the limited screen space, and second, we need to display sufficient diversity.

At any given time, the translator will be concerned only about a specific sub-part of the source sentence. She will also be more concerned with content words, especially if they are unusual. Hence, we will focus on displaying translation options that are relevant to the current sub-part of the sentence. It is an option question if the selection of which sub-part needs to be supported is done automatically based on mouse or cursor position, or should be triggered on demand. We will implement multiple solutions and test them in user studies.

It is important to display sufficient variety in translation options. A weakness of the solution in Caitra was that often only morphological variants, alternatives with and without function words, and different phrase segmentations were displayed, leaving not much room for more valuable information. We developed solutions to the problem of increasing diversity in the work on automatic paraphrasing, using a clustering approach, which is described in more detail in Section 6. We continue to explore this problem deeper.

Finally, there is a question how the user interacts with this type of assistance. User studies will reveal, if glancing at the options and deriving inspirations for use in the translation is sufficient, or if the options could be used directly as clickable buttons that insert text automatically.
5.2 Translation Options in Context

Providing the translator with a list of several alternative translations for each word or phrase in the source sentence is a powerful tool, but in many cases, given a variety of possible translations a translator might be struggling to decide which of these options best fits the particular context that the source word of phrase appears in.

For instance, given the German word *Vortrag*, a translator might first think of *lecture, talk* or *speech* as English translations. These are all valid translations, but which specific one to pick depends on the context that the word is used in, and the user might want to stimulate their memory by having a quick look at existing translations from other translators, shown within their original sentences.

In a similar scenario, a translator given the French sentence *dépasser les attentes* in the context of a sales pitch might wonder which English translation among *surpass expectations, exceed expectations* or *go beyond expectations* is more idiomatic and more typical of sales writing.

The bilingual concordancer provides just this sort of functionality. The user can select any word or phrase in the source text, and at the click of a button they are presented with a list of example translations of the word. A sampling of all known translations is shown, with the more frequent translations shown first. For each example, an excerpt is shown from both the source and translated text surrounding the queried words. This allows the user to assert which of the translations has a specific meaning that is most suitable to the particular sentence they’re translating.

This sort of assistance is sought after by translators as a memory aid and as a tool for helping them pick the best possible translation.
It is also commonly used by translators in corporate environments who must adhere to a specific, rigid, imposed terminology, which ensures that all translators translating for the same company use a single consistent, coherent terminology. In this scenario, the biconcordancer can be trained exclusively on translations previously produced and approved by the same company, such that their content has already been vetted as conformant to the corporation’s sanctioned terminology. Thus both the translations of the queried word themselves, as well as all the contextual examples shown, will be guaranteed to comply with the company’s terminology.

**Technical Aspects**

The biconcordancer takes its example translations from databases stored in the same format of parallel data that is used to train the translation system, namely plain, sentence-aligned text, which is tokenized, truecased, and split into sentences.

One big advantage of reusing the same data format as is used by mainstream machine translation systems is that there are already large amounts of data readily available. Machine translation systems require great quantities of such parallel data, and hence a lot of effort has already been put into gathering these parallel corpora. The commonality of data formats allows us to harness those existing data sources, providing us with very large databases for the biconcordancer to read from at no extra cost.

Having large amounts of example translations to pick from is a highly important factor in achieving a high usability for the translation options tool. The very usability and usefulness of the biconcordancer lies in the underlying database comprising the widest possible array of examples and translations for each source word or phrase. As with many linguistic phenomena, words then have a small number of translations that appear often, and a large number of translations that each appear only rarely. It is by tapping into these numerous, rare translations that we can make the concordancer most useful, and access to large databases of parallel data is the best way to achieve high coverage.

As with other applications of plain text parallel corpora, however, the word alignments between the source and target texts have to be computed using stochastic methods, rather than being manually input by trusted human sources. This is inevitable due to the sheer volume of data that is being handled: manually aligning every sentence pair, i.e. indicating which word is a translation of which, and which phrase is a translation of which, would be beyond prohibitive in terms of cost.

A good, reliable alignment is of course the cornerstone of a good biconcordancer. Thankfully, word and phrase alignment is a challenge that has been studied for decades, and is fairly well understood and handled. Nevertheless it is important to keep in mind that the alignments that form the core of the biconcordancer technology are computer-generated, and do contain a small number of errors. These errors will manifest themselves as translations being suggested to the user where the translated word is not actually a translation of the user-selected source word. Fortunately, such errors tend to be obvious to the user, since rather than being slightly off in meaning, the translation will be patently absurd. The risk here is not that the translator will be misled by the software, but simply that they might get slightly annoyed.

Overall the biconcordancer tool has proved itself to be very useful in providing the kind of assistance to translators that is not readily given by bilingual dictionaries and other common translator tools. The biconcordancer was presented to a team of professional translators and elicited a very positive response.
6 Authoring Assistance (Task 3.6)

Work on authoring assistance in the reporting period focused on the development of a paraphrasing method that suggests reformulations of any part of the translation. Work was mainly carried out in the context of a Master of Science in Artificial Intelligence thesis project by Turan Rustamli at UEDIN.

6.1 Introduction

To assist translators in the editing and refinement of their translation, we developed a novel paraphrasing method. The method aims to help human translators to fix an automatic translation by suggesting paraphrases to any parts of the translation that they specify. See Figure 2 for screenshots of the integration of the paraphrasing method into the casmacat workbench. In the edit area, the user may mark up any sequence of words, and then request paraphrases. These are displayed in a pop-up window and can be selected to replace the marked content.

The paraphrasing method uses the search graph produced during the machine translation decoding process as the main data source. By exploring alternate paths in the search graph, it generates a set of possible paraphrases which are then re-ranked with additional features, including a feature that biases the list of paraphrases towards diversity.

To test the feasibility of our method, we carried out two major evaluation stages. During the first stage, we designed an original automatic evaluation tool and used it to assess ten different versions of our paraphrasing approach. In order to support our findings we carried out the second stage during which a group of four annotators judged paraphrasing results produced by our method. The results of both stages demonstrate that the information available from the search graph is sufficient to produce useful paraphrasing suggestions.

6.2 Requirements

An essential aspect of our paraphrasing approach is that it aims to provide real-time targeted paraphrase search. Unlike previous paraphrasing work, we consider an interactive environment where queries are sent by users and executed against our data model. The results of these queries after being dynamically filtered and ranked using various heuristics are returned back to user. In this section we list the main requirements that were considered during the design stage.

Integration with existing CAT software. Taking into account that most modern CAT implementations are providing a web based service, we also employed the client-server architecture to design our paraphrasing service. As a result it might be easily attached as a module to the services provided online. Moreover, our service could be hosted through an API. This way our paraphrasing assistance might be accessible from all major types of CAT applications.

Reusing results of machine translation. Another important feature of our paraphrasing system is being able to produce paraphrases using only data available as a result of machine translation carried out to provide other assistance types. We designed our implementation to support translation output generated by Moses.

Importance of ranking. Quality of the final ranking of the paraphrasing options is crucial requirement for our tool that aims to provide an interactive service. Indeed, basic concepts of Human-Computer Interaction suggest that paraphrasing options displayed to users will be efficient only in case if their count is not more than 7. This means that its important to rank top results in a way that they will contain at least one good paraphrase.
Figure 2: Integration of paraphrasing in the CASMACAT workbench
**Paraphrasing granularity.** Various levels of paraphrase granularity are studied. In case of entire sentences the task is known as sentential (or clausal) paraphrasing, for shorter items it is called lexical (or phrasal) paraphrasing. For our project we investigated only lexical paraphrases, motivating this decision by the fact that other assistance tools within a CAT system may already provide multiple translation options for entire sentence. For example, results of translation by using different bilingual parallel corpora could be provided. We also acknowledge that within the scope of lexical paraphrases there is a further separation into two classes that correspond to shorter and longer phrases. Both cases were considered by us during the evaluation.

We also took into account the study by Simard and Macklovitch (2005) which analyses usage patterns of a different assistance tool, a bilingual concordancer. The authors suggest that the average length of the input query is 2-3 words. Given the parallels between this type of assistance and ours, we expect average paraphrasing query length to be the same.

**Realtime experience.** Another requirement for our interactive tool is providing paraphrasing options as fast as possible. Our tool will not be useful if returning results takes too longer.

**Coexistence with manual editing.** While our tool aims to be used in order to fix an erroneous part in the translation by getting the corrected paraphrase automatically, we acknowledge that final high quality translation cannot be achieved only by using paraphrasing. An ability to manually post-edit translation is crucial. Designing our approach we considered that it will be used in an environment where translation could be altered by user at any time.

### 6.3 Generation of Paraphrases

The post-editor marks up a sequence of words in the machine translation and requests paraphrases, which we generate from the search graph of the machine translation process. We use all phrase translations that cover the same input words as the marked up sequence. For longer sequences, multiple adjacent phrase translations may be used. We consider all possible combinations, not only valid paths in the search graph, and we combine these phrase translations into paraphrasing options.

Given the combinatorial explosion of combinations of phrase translations, we generate only an n-best list of choices using heuristic search, which is very similar to the machine translation decoding, but by disallowing reordering, it can be handled with a finite state machine. To guide the search in these cases, we use the same scoring functions that we use for the ranking of paraphrases described in the next section.

### 6.4 Ranking of Paraphrases

To test out a number of different ways to rank paraphrases, we designed our paraphraser in a way that some parts of its logic are represented by dynamically attachable functions. These functions are grouped by their use cases. Each function within such a group accepts the same arguments and returns the result in the same format.

The groups are: filters, partial filters, score functions and sorters. In this section, we provide a list of the implemented functions. We assign a four letter code for each function. Later these codes will be used to describe functions that were used in different experiments.
6.4.1 Partial Filters

We described how paraphrasing options are built using a finite state machine. Transitions of this machine are phrase translations. We use partial filters to remove some phrase translations that are not likely to be part of a good paraphrasing option. There are two ways to instantiate a partial filter. The first way is to construct it using a list of phrase translations in a format as we get it from database. Items of this list contain all features that are initially available in search graph attributes. After processing this list, the partial filter will output a list in the same format with reduced number of items. The second way to instantiate a partial filter is to construct it using another partial filter. In this case before applying filtering logic, our partial filter will execute another partial filter and then will use items results as initial data for filtering. Using both constructors makes it possible to build complex filters.

The following partial filters were implemented during the project:

**PTPF: Punctuation Partial Filter.** We iterate the input list and remove all punctuation and special signs for each item. We also replace multiple spaces between words with one. We remove duplicates from the resulting list and return the final list.

**SDPF: String Distance Partial Filter.** For each item in the list we calculate the Levenshtein string distance between it and the following items. If the distance is below a given threshold \( t \) and a following item has the same coverage as the current item, we remove it from the list. For all our experiments we used \( t = 3 \), because we intuitively assumed that by filtering items with longer distance we might lose good partial paraphrasing candidates.

**FWPF: FunctionWord Partial Filter.** For each item we remove the following items if the only difference between two strings is in function words. This way in the resulting list there are still function words.

6.4.2 Score Functions

We use score function in order to assess a transition between two steps in the paraphrase generating finite state machine.

**BFSF: Best Forward Score Function.** This function reuses information about the best next step from the search graph. It checks if the target the hypothesis has the same id as the forward reference for the source hypothesis. In this case it outputs 1. Otherwise it retrieves the best next hypothesis and calculates a string distance between it and target. If distance is more than 9, function outputs 0. Otherwise it the result is calculated as \( 1 - \frac{d}{10} \), where \( d \) is the distance.

**SDSF: Score Difference Score Function.** Function calculates the difference between translation scores for two states. Output is a relative difference score between 0 and 1, calculated as: \( \min\left(\frac{\text{score}_t}{\text{score}_s}, 1\right) \), where \( \text{score}_t \) is the target score and \( \text{score}_s \) is the source score. As a result output will be high if as a result of transition search graph score increases, and low otherwise.

**LMSF: Language Model Score Function.** This function uses language model that was used for machine translation to score the result of merging outputs of two hypotheses.

6.4.3 Filters

In contrast with partial filters, regular filters are applied to the final list of paraphrasing options. This list is generated using FSM and we still may want to remove paraphrasing candidates that are similar. Two filters implemented by us are repeating logic of the String Distance and the
FunctionWord partial filters. They are correspondingly named String Distance Filter (SDRF) and FunctionWord Filter (FWFF).

The only fully novel filter is a score based filter:

**SBRF: Score Based Filter.** This simple filter removes items that have a final FSM transition score which is below a predefined threshold. The filter is used only in cases when the number of results is too large to be handled by sorters. Threshold depends on number of items in the list.

### 6.4.4 Sorters

Sorters are applied to the filtered list of paraphrasing candidates. These functions aim to produce the best possible final ranking for the paraphrases. They have a similar interface with filters, the difference is that instead of removing items, sorters reorder them in the resulting list. The initial list passed to sorters is already sorted by FSM transition scores.

**LMBS: Language Model Based Sorter.** In contrast to the Language Model Score Function, this sorter uses language model score for a whole sentence by replacing the marked input with the paraphrase. This way, the paraphrase is assessed in context of the surrounding words. Items in the list are rearranged by score in descending order.

**CBDS: Cluster Based Diversity Sorter.** This sorter is based on a heuristic which assumes that more diverse results are more useful for users. Indeed, a user searching for paraphrases is likely to be looking for a way to express an idea in a different way, rather than trying to fix a minor spelling mistake. To achieve more contrasting results we use a simple clustering approach. We group items into clusters based on word distance. The distance is calculated similarly to the string distance, considering words as atomic units instead of characters. For clustering we use \( k \)-Means. The number of clusters \( k \) is set to be equal to the number of top paraphrases displayed to user. The order within each cluster is the same as in the original list. The next step after clustering is to create an output list. To do so we iteratively populate an empty list, by adding the top items from each cluster. The order of clusters depends on the order of their first item in the original list. This way we achieve an output list where top results are significantly different from each other, but at the same time have high scores.

### 6.5 Evaluation

We are mainly interested in evaluating the usefulness of paraphrasing options provided by our tool in context of a translation task. We designed an automatic evaluation system that helped us to find out various issues and interesting features about our tool. We also carried out a small manual user study.

#### 6.5.1 Automatic evaluation

Our main goal was to test if our paraphraser can find a suitable paraphrase for a given part of machine translation output. To achieve this goal we created a large repository of test cases. Each test case contains a machine translation corrupted sentence and a set of suggested paraphrases for different parts of the sentence. In our evaluation, we treat the original phrase as gold standard.

In order to collect this kind of test data, we artificially corrupted 1000 English (natural language) sentences from news domain (newstest2012b dataset) by translating them into Russian.
and then back into English. This way, the resulting English sentences were expressing the same idea as the original ones. However most of them were stylistically or grammatically incorrect.

We detected differing parts in the original and the corrupted version and passed the corrupted parts to our paraphraser. We considered the test case successful if corresponding original part was a top \( n \) paraphrasing suggestion. In most experiments we considered \( n = 5 \), as this is the desired number of options displayed to the user. We begin this section by providing a short overview of the process of test case generation, then we will provide the evaluation results.

In order to generate the test cases, we used the Moses toolkit to translate 1000 English (natural language) sentences into Russian.

For instance, we translate

\[
\text{Unlike in Canada, the American states are responsible for the organisation of federal elections.}
\]

into

\[
\text{В отличие от Канады, американские штаты ответственны за организацию федеральных выборов в Соединенных штатах.}
\]

We then translate it back into English.

\[
\text{Unlike in Canada, the American states are responsible for the organisation of federal elections.}
\]

As we can see, the back translation is different from the original English input. In this case the difference is in phrases \textit{us states} and \textit{the American states}. Obviously, the original phrase is a better translation and if our paraphraser can suggest it in top 5 results we can consider it a successful test case.

We extracted all differences between original English and corrupted English sentences and stored them in our test cases repository. To detect corrupted parts that are different from original, we used coverage information to align parts of corrupted and original English sentences. This way we collected 2139 test cases, that we used for evaluation.

We tested ten different versions of our approach using the test case repository we discussed in the previous section. These versions differ in filters, sorters and score functions.

Table 1 shows the outcome of our automatic evaluation. The first columns identify types of functions we used. Description of these functions could be found in the previous section. The final column contains the score, which expresses the number of successfully solved test cases out of 2139 total tests.

As we can see from results shown in Table 1 we achieve the best performance with Approach 10. This approach uses all partial filters, all score functions and all sorters in same order as listed in the table. Before sorting final results the best approach also filters them through the Score Based Filter. In contrast, Approach 1 has the worst performance. For this approach we did not use any filters or sorters. However we used punctuation partial filter and score difference based score function. We consider Approach 1 as the baseline approach, because it uses only a minimum of required functions. Furthermore, we can see that we achieve the best
Table 1: Automatic evaluation results

<table>
<thead>
<tr>
<th>Method</th>
<th>Partial Filters</th>
<th>Score Function</th>
<th>Filters</th>
<th>Sorters</th>
<th>Match</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PTPF</td>
<td>SDSF</td>
<td>-</td>
<td>-</td>
<td>135/2139</td>
</tr>
<tr>
<td>2</td>
<td>PTPF, FWPF, SDPF</td>
<td>SDSF</td>
<td>-</td>
<td>-</td>
<td>161/2139</td>
</tr>
<tr>
<td>3</td>
<td>PTPF, FWPF, SDPF</td>
<td>BFSF</td>
<td>-</td>
<td>-</td>
<td>159/2139</td>
</tr>
<tr>
<td>4</td>
<td>PTPF, FWPF, SDPF</td>
<td>LMSF</td>
<td>-</td>
<td>-</td>
<td>211/2139</td>
</tr>
<tr>
<td>5</td>
<td>PTPF, FWPF, SDPF</td>
<td>SDSF, BFSF</td>
<td>-</td>
<td>-</td>
<td>161/2139</td>
</tr>
<tr>
<td>6</td>
<td>PTPF, FWPF, SDPF</td>
<td>SDSF, LMSF, BFSF</td>
<td>-</td>
<td>-</td>
<td>218/2139</td>
</tr>
<tr>
<td>7</td>
<td>PTPF, FWPF, SDPF</td>
<td>SDSF, LMSF, BFSF, PTRF, FNRF</td>
<td>LMBS</td>
<td>574/2139</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>PTPF, FWPF, SDPF</td>
<td>SDSF, LMSF, BFSF, PTRF, FNRF</td>
<td>CDBS</td>
<td>493/2139</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PTPF, FWPF, SDPF</td>
<td>SDSF, LMSF, BFSF, SBRF</td>
<td>LMBS</td>
<td>661/2139</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>PTPF, FWPF, SDPF</td>
<td>SDSF, LMSF, BFSF, SBRF</td>
<td>LMBS, CDBS</td>
<td>691/2139</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Automatic evaluation results

effects by adding language model based features. Also from the results we can see that using sorters significantly improves the performance of the paraphraser.

We applied the sign test to test significance of the results. Considering \( p \leq 0.05 \) as significance level, we found out that approaches 2 and 3 are not significantly different, as well as approaches 4 and 6. All other approaches were significantly different from each other.

Using a visualisation tool we detected multiple problems with this way of automatic paraphrases evaluation. Firstly, we noticed that our paraphraser performs significantly better in case of shorter phrases, while for longer phrases it almost always fails to find the desired paraphrase.

However, by analysing long phrases we noticed that some of them represent good paraphrasing candidates, that despite not matching the original natural language English phrases, should be considered as success cases for paraphraser. Indeed, if a paraphrase does not match the original text, it should not mean a failure, it still can be a good paraphrasing option.

Considering this fact our evaluation scores should be interpreted as at least N correct cases out of 2139. In order to verify results achieved by our automatic evaluation approach we decided to carry out a user study, which will be described in the next section.

6.5.2 Manual Evaluation

During this project we also developed a web based interactive evaluation tool. This tool presents the user with an interface for manual evaluation of paraphrasing results. It reuses automatic evaluation test cases, displaying machine translation of the sentence and highlighting an area that is considered to be corrupted as a result of alignment with original English sentences. Below the sentence we located top 5 selectable paraphrasing options that were returned by a given paraphraser version. The user can pick one of these options as a better paraphrase or leave the original in place. We also added an option for user to suggest a better paraphrase if neither any of the options nor original text are suitable translations.

Considering results of automatic evaluation, we picked three out of ten tested approaches for the user study. These approaches are 1, 7 and 10. For these, we randomly picked 50 out of our 2139 test cases, ensuring that both short and long phrases are present in the final selection. We had four volunteer participants who were provided with a background information about the context of paraphrasing, they were instructed to pick a better paraphrase for selected part of sentence and to press Accept, or alternatively to suggest a custom paraphrase and proceed by clicking Submit Custom.

The participants were encoded by their initials in the following way: HG, OM, NH and RM. For each of the three approaches each participant submitted his decisions on the same 50 test cases. We also considered the case when a user comes across a test case that he already
assessed for a previous approach and his desired paraphrase selection already is in top five current approaches. We automatically skipped these test cases, considering them successful.

Results of the user study are provided in Table 2. The first column is the approach number, the four next columns contain number of successful test cases for each user. And finally, the last column is a unified score, which considers case successful if at least two of four annotators decided that top 5 list contains a suitable paraphrase. We can see that these results correlate with the results of automatic evaluation. Indeed, by applying sign tests with $p \leq 0.05$ as significance level, we verified that all three approaches are significantly different and most importantly that Approach 10 is significantly better than the baseline approach.

<table>
<thead>
<tr>
<th>Method</th>
<th>HG</th>
<th>OM</th>
<th>NH</th>
<th>RM</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>6/50</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>17</td>
<td>12</td>
<td>10</td>
<td>13/50</td>
</tr>
<tr>
<td>10</td>
<td>24</td>
<td>20</td>
<td>26</td>
<td>29</td>
<td>26/50</td>
</tr>
</tbody>
</table>

Table 2: Manual evaluation
References


certainty of machine translation quality estimates. In *Proceedings of the Machine Translation
Summit*.


Attachment A

Task 3.1


Dimensionality reduction methods for machine translation quality estimation.

*Machne Translation.*
Dimensionality reduction methods for machine translation quality estimation

Jesús González-Rubio · J. Ramón Navarro-Cerdán · Francisco Casacuberta

Received: 4 October 2012 / Accepted: 24 April 2013
© Springer Science+Business Media Dordrecht 2013

Abstract Quality estimation (QE) for machine translation is usually addressed as a regression problem where a learning model is used to predict a quality score from a (usually highly-redundant) set of features that represent the translation. This redundancy hinders model learning, and thus penalizes the performance of quality estimation systems. We propose different dimensionality reduction methods based on partial least squares regression to overcome this problem, and compare them against several reduction methods previously used in the QE literature. Moreover, we study how the use of such methods influence the performance of different learning models. Experiments carried out on the English-Spanish WMT12 QE task showed that it is possible to improve prediction accuracy while significantly reducing the size of the feature sets.

Keywords Machine translation · Quality estimation · Dimensionality reduction · Partial least squares regression

1 Introduction

Despite an intensive research in the last 50 years, machine translation (MT) systems are still error-prone. Thus, a desirable feature to improve the broader and more effective
deployment of (nowadays) imperfect MT technology is the capability of predicting the reliability, namely the quality, of the generated translations. Historically, translation quality assessment has been done manually by human experts. These experts need to read the automatic translation and the source text to be able to judge whether the translation is good or not which, obviously, is a very time consuming task. Therefore, automatical translation quality assessment is a crucial problem, either to present the translations in such way as to make end-users aware of the quality (Specia et al. 2009b), or to filter out the translations according to the requirements of a given task and level of expertise of the professional translator, e.g. to avoid professional translators spending time reading/post-editing certain translations (Blatz et al. 2004; Quirk 2004; Specia et al. 2009a; González-Rubio et al. 2010). This task, referred to as confidence or quality estimation (QE), is concerned about predicting MT output quality without any information about the expected output. Quality information may be provided for each word (Gandrabur and Foster 2003; Ueffing and Ney 2007; Sanchis et al. 2007), sentence (Blatz et al. 2004; Quirk 2004; Gamon et al. 2005; Specia et al. 2009b) or document (Soricut and Echihabi 2010). This article focuses on sentence-level QE.

We distinguish the task of QE from that of MT evaluation by the need, in the latter, of reference translations. The goal of MT evaluation is to compare an automatic translation to reference translation(s) and provide a quality score which reflects how close the two translations are. In QE, the task consist in estimating the quality of the translation given only information about the input and output texts and the translation process.

Sentence-level QE is typically addressed as a regression problem (Quirk 2004; Blatz et al. 2004; Specia et al. 2009b). Given a translation generated by an MT system (and potentially other additional sources of information) a set of features is extracted. Then, a model trained using a particular machine learning algorithm is employed to compute a quality score from these features. Most QE works consider a fixed set of features and study the performance of different learning algorithms on those features. However, feature sets tend to be highly redundant, i.e. there is high multicollinearity between the features, and some of the features may even be irrelevant to predict the quality score. Moreover, a set of translations labeled with their “true” quality score is required to train the learning model. Since this labeling process is usually done manually, training sets rarely contain enough labeled samples to accurately train the model. By removing irrelevant and redundant features from the data, dimensionality reduction (DR) methods potentially improve the performance of learning models by alleviating the effect of the “curse” of dimensionality, enhancing generalization capability of the model, and speeding up the learning process. Additionally, DR may also help the researchers to acquire better understanding about their data by telling them which are the important features and how they are related with each other. Despite these potential improvements, works on QE usually put little attention on DR. For example, only six out of the eleven participants to the QE task of the 2012 workshop on statistical MT (Callison-Burch et al. 2012) applied DR, and even those participants that used DR only implemented simple feature selection methods.

In this article, we propose two novel DR methods based on partial least squares regression (PLSR) (Wold 1966). We consider both a DR method that selects a subset of the original features, namely a feature selection method, and a method that projects
Dimensionality reduction methods

Fig. 1 Dataflow of the proposed two-step quality estimation approach

the original data into a space of fewer dimensions, a feature extraction method. Despite being usually more complex, feature extraction methods have a potential advantage over feature selection: they can generate new features that summarize the “information” contained in all original features. In contrast, the information contained in the features discarded by a feature selection method is inevitably lost. The proposed methods are compared to other DR methods previously used in the literature: methods based on statistical multivariate analysis such as PCA (Pearson 1901) and PLSR regressors selection (Specia et al. 2009b), and heuristic wrapper selection methods (Kohavi and John 1997). Moreover, we study how these DR methods affect the performance of different learning models.

The performance of each DR method was evaluated by the prediction accuracy of the models trained in the corresponding reduced feature sets. Figure 1 shows a scheme of the process followed to obtain a quality score from a given translation. First, from the translation, and additional information sources, we compute a (possibly high-dimensional and highly-redundant) set of features that represent the translation. Then, we apply a DR method to obtain a reduced feature set that still contains the relevant information present in the original feature set. Finally, we use a trained learning model to predict the quality score of the translation from this reduced feature set. To assure an accurate comparison between the different DR methods, identical pipelines were used to train the models. By providing a detailed description and a systematic evaluation of these DR methods, we give the reader various criteria for deciding which method to use for a given task.

It should be noted that despite being tested in a QE task, the proposed two-step training and DR methods do not make particular assumptions about the features or the learning model. Thus, they constitute a general methodology that can be applied to a great variety of supervised learning tasks.

The rest of the article is organized as follows. In Sect. 2, we formalize the regression approach to QE. In Sect. 3, we state the DR problem and present the different DR methods under study. Section 4 is devoted to describe our experimental setting which include a description of the features extracted for each translation (Sect. 4.2), and the different learning models used in the experimentation (Sect. 4.3). In Sect. 5, we present and discuss the empirical results obtained in the experimentation, and, finally, we conclude with a summary in Sect. 6.

2 Quality estimation

We formalize QE as a regression problem where we model the relationship between a dependent variable \( y \) (the quality score), and a vector of \( m \) explanatory variables
\( x^T = (x_1, \ldots, x_m) \) (the features that represent the translation). Given a data set with \( n \) samples \( \{y_i, x_i\}_{i=1}^n \), our goal is to build a predictive model \( M_{\theta} : \mathbb{R}^m \rightarrow \mathbb{R} \) with free parameters \( \theta \). The data set is usually represented in matrix form where \( y \) is a vector that contains the quality scores, and \( X \) is a matrix where each row is the feature vector of one training sample:

\[
\begin{pmatrix}
  y_1 \\
  y_2 \\
  \vdots \\
  y_n
\end{pmatrix}
= 
\begin{pmatrix}
  x_1^T \\
  x_2^T \\
  \vdots \\
  x_n^T
\end{pmatrix}
= 
\begin{pmatrix}
  x_{11} & x_{12} & \cdots & x_{1m} \\
  x_{21} & x_{22} & \cdots & x_{2m} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_{n1} & x_{n2} & \cdots & x_{nm}
\end{pmatrix}
\]

To carry out the regression, the form of the model \( M_{\theta} \) must be specified. Since we do not know how \( y \) and \( X \) actually relate, we use different flexible models (see Sect. 4.3) whose free parameters \( \theta \) can be estimated to fit the data. Typically, these models include a regularization term (Tibshirani 1996) that facilitates the learning process in the presence of noisy and collinear data. One of the goals of the experimentation will be to study if regularized models can also benefit from an explicit DR of the feature space.

3 Dimensionality reduction

3.1 Motivation

The proposed QE formalization assumes that translation quality can be described by a number of independent variables. Since these underlying variables are unknown, in practice, we instead extract a (possibly larger) set of features that aim at describing the prediction information contained in the underlying variables. This approach implies to consider translation quality as governed by more variables than it really is, which results in several learning problems due to the addition of irrelevant features, or the multicollinearity between them. However, provided the influence of this “extra” features is not too strong as to completely mask the original structure, we should be able to “filter” them out and recover the original variables or an equivalent set of them. DR methods aim at somehow strip off this redundant information, producing a more economic representation of the data.

DR can also be seen as a method to overcome the so-called “curse” of dimensionality. This term, coined in Bellman (1961), refers to the fact that, in the absence of simplifying assumptions, the sample size needed to estimate a function of several variables to a given degree of accuracy grows exponentially with the number of variables. Responsible for the “curse” of dimensionality is the fact that high-dimensional spaces are inherently sparse which is known as the empty space phenomenon (Scott and Thompson 1983). This is a difficult problem in model estimation, as regions of relatively very low density can contain a considerable part of the distribution, whereas regions of apparently high density can be completely devoid of observations in a sample of moderate size. DR technology address these problems, by reducing the input dimension of the function to be estimated.
3.2 Problem statement and approaches

The DR problem can be stated as follows: given a regression problem \( P_1 : \mathbb{R}^m \rightarrow \mathbb{R} \), we want to obtain an equivalent problem \( P_2 : \mathbb{R}^r \rightarrow \mathbb{R} \) where \( r \ll m \). In other words, we want to obtain a low-dimensional, compact representation of the input data that still retains the information required to perform an accurate prediction. Formally, DR is defined by a function \( \Delta \) that transforms an \( m \)-dimensional space into an \( r \)-dimensional space:

\[
\Delta : \mathbb{R}^m \rightarrow \mathbb{R}^r
\]  

(1)

The determination of the dimension \( r \) of this compact representation is central to the DR problem, because knowing it would eliminate the possibility of over- or under-fitting. All the methods studied in this article take this intrinsic dimension as a parameter to be given by the user; a trial-and-error process is thus necessary to obtain a satisfactory value for it.

Next, we describe the different DR methods tested in the experimentation. For a more clear presentation, we distinguish between heuristic methods and methods derived from statistical multivariate analysis.

3.3 Heuristic feature selection methods

We consider heuristic wrapper (Kohavi and John 1997) methods to address the problem of feature selection. In the wrapper methodology, the learning model is considered a perfect black box. In its most general formulation, this methodology consists in using the prediction accuracy of a given learning model to assess the relative usefulness of subsets of features. In practice, the different wrapper methods are defined by the search strategy implemented to explore the space of possible subsets. An exhaustive search can conceivably be performed if the number of features is not too large. For example, all the subsets for 24 features \( 2^{24} \) were explored in Soricut et al. (2012). However, the problem is known to be NP-hard (Amaldi and Kann 1998) and the search quickly becomes computationally intractable.

In our experimentation, we tested two search strategies that define two different heuristic feature selection methods: ranking of feature selection, and greedy forward selection (GFS). Since the computational complexity of these simple methods depends on the complexity of the chosen learning model, we use symbol \( \zeta(n, m) \) to denote the time complexity to train the actual learning model with \( n \) samples of \( m \)-dimensional feature vectors.

3.3.1 Rank of feature

Rank of feature selection (RFS) generates subsets of features by selecting the top-scoring features according to the prediction accuracy of a QE system trained solely with that feature (González-Rubio et al. 2012). RFS is typically used as a baseline selection mechanism because of its simplicity, scalability and (somewhat) good empirical
success (Guyon and Elisseeff 2003). The computational complexity of RFS to generate the first reduced feature set is given by \( O(m \cdot \zeta(n, 1)) \); once the scores for the features are computed, we can generate reduced groups of different sizes with no further calculations. For example, the complexity of RFS if we use a linear model\(^1\) is in \( O(m \cdot n) \) given that \( \zeta(n, 1) \) is proportional to \( n \).

Since RFS selects the features according to their individual prediction accuracy, we expect to obtain subsets of features that also provide good prediction accuracy. However, RFS does not take into account the correlations that may exist between the different features, thus, these subsets will probably contain a large number of redundant features.

### 3.3.2 Greedy forward

Greedy forward selection (Kohavi and John 1997; Avramidis 2012) incrementally creates subsets of features by selecting at each iteration the feature that, when added to the current set, yields the learned model that performs best. In contrast to RFS, GFS recomputes the importance of each feature at each step having into account the current subset of features. Thus, the computational complexity of GFS to compute a reduced set of size \( r \) is \( O(\sum_{i=1}^{r} \sum_{j=1}^{m-i+1} \zeta(n, i)) \) that is upper bounded by \( O(r \cdot m \cdot \zeta(n, r)) \). For example, if we use a linear model the temporal complexity of GFS is in \( O(r^2 \cdot m \cdot n) \) given that \( \zeta(n, r) \propto n \cdot r \).

Since GFS selects at each step the feature that improves most the QE model performance, we expect to obtain subsets with lower redundancy in comparison to RFS. However, it requires to re-compute the contribution of each feature to the QE model at each step, \( O(\zeta(n, r)) \), which penalizes GFS complexity.

### 3.4 DR methods based on statistical multivariate analysis

Statistical multivariate analysis is a generic term for any statistical technique concerned with analyzing data in high dimensions (Anderson 1958). In particular, we focus on statistical techniques to partition the variability of the data into components attributable to different sources of variation. In this work, we consider two of these techniques: principal component analysis (PCA), and PLSR. Given a number of dimensions \( r \), both PCA and PLSR compute a transformation of the original data space into an orthogonal \( r \)-dimensional space. However, they differ in the criteria followed to compute this transformation.

The main advantage of these methods stems in the orthogonality of the output space; which means that the transformed features will be linearly independent by construction. Therefore, using these transformations we obtain reduced feature sets with almost no redundant information. Moreover, statistical multivariate methods are mathematically well-founded and independent of the chosen learning model. However, these methods also have an obvious drawback, i.e. new features are computed as

---

\(^1\) This particular setup can be considered as a lower bound of the complexity of RFS.
Dimensionality reduction methods

Fig. 2  PCA example for a 2-dimensional gaussian distribution. The vectors represent the two principal components of the data

a linear combination of all original features which makes it often difficult to interpret them.

3.4.1 Principal component analysis

Principal component analysis (Pearson 1901) defines a transformation of the original data into a new space of features, known as principal components. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint of being uncorrelated with the preceding components. Therefore, each of these principal components represent one of the individual latent factors that actually govern the variability of the data, as exemplified in Fig. 2.

Given a matrix \( X \) whose rows represent the \( n \) samples and each column represents one of the \( m \) features, PCA is formalized by the following decomposition:

\[
X = TP^T
\]  
(2)

where \( P \) is the space transformation matrix that contains the eigenvectors of the covariance matrix \( X^T X \), and the rows of \( T \) represent the principal components of each training sample. The nonlinear iterative partial least squares (NIPALS) algorithm (Wold 1966) is commonly used to obtain the eigenvectors.

Given that the eigenvectors in \( P \) are unitary and orthogonal \((P^T P = I)\), we can multiply both sides of Eq. (2) by \( P \) to obtain the principal components \( T \) of the data:

\[
XP = T
\]  
(3)

Figure 3 shows a graphical example of the computation of two principal components \( t = (t_1, t_2) \) for a single data point \( x \). Each principal component \( t_k \) is computed by projecting \( x \) over the corresponding unitary eigenvector \( p_k \). Specifically, \( t_k = x \cdot p_k = ||x|| \cdot ||p_k|| \cdot \cos(\alpha_k) = ||x|| \cdot \cos(\alpha_k) \), where \( \alpha_k \) is the angle between \( x \) and \( p_k \).

3.4.1.1 PCA projection  The principal components are linearly independent, and each of them accounts for the maximum variability in \( X \) not explained by previous components, thus we follow González-Rubio et al. (2012) and select the first \( r \) components to
create the reduced feature sets. Since each of these components is a linear combination of the original features, this is a feature extraction method. In the experiments, we use PCA-P to denote this DR approach.

The complexity of PCA-P to compute a reduced set of size $r$ is given by the complexity of the NIPALS algorithm: $O(r \cdot m \cdot n)$. Note that in contrast to the previously presented heuristic methods, the cost of PCA-P does not depend on the complexity of the chosen learning model.

### 3.4.2 Partial least squares regression

PCA generates sets of orthogonal features where each feature explains the variability of the data $X$ in one principal direction. However, this transformation ignores the scores $y$ to be predicted. Thus, although the features generated by PCA-P contain almost no redundancy, they do not necessarily have to be the best set of features to perform the prediction. Partial least squares regression (PLSR) (Wold 1966) is an alternative to PCA that takes into account $y$ when computing the transformation of $X$. Specifically, PLSR computes an ordered set of latent variables such that each of them accounts for the maximum co-variability between $X$ and $y$ under the constraint of being uncorrelated with previous latent variables. Formally, PLSR builds the following model where $b$ is a vector of regressor coefficients, and $f$ is a vector of zero-centered Gaussian errors:

$$y = Xb + f \quad (4)$$

Even though this is a linear regression model the estimation of the regression coefficients $b$ for PLSR is different from the conventional least squares regression, see Sect. 4.3.1. The intuitive idea of PLSR is to describe $y$ as well as possible, hence to make $||f||$ as small as possible, and, at the same time, take advantage of the relation between $X$ and $y$. To do that, PLSR defines two independent PCA-like transformations $P$ and $q$ (for $X$ and $y$ respectively) with $E$ and $f$ being the corresponding residual errors, and a linear relation $R$ linking both blocks:

$$X = TP^T + E \quad y = Uq^T + f \quad (5)$$

$$U = TR \quad (6)$$
where matrices $\mathbf{T}$ and $\mathbf{U}$ are the projections from $\mathbf{X}$ and $\mathbf{y}$ respectively. Specifically, each of the columns of the $\mathbf{T}$ matrix represents one of the latent variables of $\mathbf{X}$.

The NIPALS algorithm (Wold 1966) is also used to solve this optimization problem. In this case, $\mathbf{b}$ is estimated as:

$$\mathbf{b} = \mathbf{Rq}^T \quad \text{where} \quad \mathbf{R} = \mathbf{W}(\mathbf{P}^T\mathbf{W})^{-1}$$

(7)

where $\mathbf{W}$ is an internal weight matrix used by the algorithm that accounts for the correlation between $\mathbf{X}$ and $\mathbf{U}$. An exhaustive description of the NIPALS algorithm for PLSR can be found in Geladi and Kowalski (1986).

Since PLSR is a much more sophisticated model than PCA, different elements of the PLSR model can be used to obtain reduced feature sets. In addition to the regressors-based selection method previously described Specia et al. (2009b), we propose one new feature selection method, variance importance in projection (VIP), and one new feature extraction method, PLSR projection. Similarly to PCA-P, the computational complexity of these three PLSR-based DR methods is also given by the complexity of the NIPALS algorithm, $O(r \cdot m \cdot n)$.

### 3.4.2.1 Feature importance in regression

Let us consider a linear model such as the one used by PLSR:

$$\hat{y} = b_0 + b_1x_1 + \cdots + b_ix_i + \cdots + b_mx_m$$

(8)

Regressor scores $b_i$ denote the expected value increment of the predicted quality score $\hat{y}$ by unitary increment of feature $x_i$, i.e., they denote the importance of each feature in the regression. However, due to the usually different scale of the features, these values cannot be directly compared; first data need to be standardized by subtracting the feature mean from the raw data values and dividing the difference by the standard deviation. Standardized features become dimensionless, and then regressors are directly comparable. We thus can create a reduced set of features by selecting them in descending regressor absolute value ($b$ in Eq. (4)). This method, first proposed by Specia et al. (2009b), is labeled FIR in the experiments.

### 3.4.2.2 Variance importance in projection

Given the weight matrix $\mathbf{W}$, we can compute the VIP (Chong and Jun 2005) of the features. VIP is a score that evaluates the importance of each feature to find the $r$ latent variables. Therefore, similarly as done for RFS in Sect. 3.3.1, we propose to select subsets of top-scoring features according to their VIP. The VIP score for the $k$th feature is given by:

$$\text{VIP}_k = \sqrt{\frac{m \sum_{j=1}^{r} \left( \frac{w_{kj}}{||w_j||} \right)^2 \text{ESS}_j}{\sum_{j=1}^{r} \text{ESS}_j}}$$

(9)

where $m$ is the number of original features, $\text{ESS}_j = b_j^2 t_j^T t_j$ is the square of the contribution of the $j$th latent variable to the score predicted by the PLSR model, $t_j$ is
the $j$th column of matrix $T$, $b_j$ is the $j$th regressor coefficient in $b$, and $\frac{w_{kj}}{|w_j|}$ is the normalized value of weight $w_{kj}$.

3.4.2.3 PLSR projection The latent variables are linearly independent, and each of them accounts for the maximum co-variability between $X$ and $y$ not explained by previous latent variables. Thus, we propose to obtain a reduced feature set by extracting the first $r$ latent variables, i.e., the first $r$ columns in matrix $T$. In contrast to PCA, the latent variables computed by PLSR take into account the relation between the features $X$ and the quality scores $y$. Therefore, in addition of being linearly independent, we expect the latent variables to attain more predictive potential than the equivalent number of principal components. This feature extraction method is labeled PLS-P in the experiments.

4 Experimental setting

4.1 Data

We computed quality scores for translations of the English-Spanish news evaluation data used in the shared QE task\(^2\) featured at the 2012 workshop on statistical MT (Callison-Burch et al. 2012). Those translations were generated by a phrase-based MT system (Koehn et al. 2007) trained on the Europarl and News Commentaries corpora as provided for the shared translation task.\(^3\) Evaluation data contains 1,832 translations for training and 422 translations for test. Each translation was manually scored by several professional translators in terms of post-editing effort according to the following scheme:

1. The translation is incomprehensible. It must be translated from scratch.
2. About 50–70% of the translation needs to be edited to be publishable.
3. About 25–50% of the translation needs to be edited.
4. About 10–25% of the translation needs to be edited.
5. The translation is clear and intelligible. It requires little to no editing.

The final quality score of each translation (a real number in the range [1,5]) is the average of the scores given by the different experts. Additionally, for each translation the corresponding source sentence, and decoding information (decoding graph and 1000-best translations) are available. We used these and the training data of the shared translation task to compute the features of each translation.

4.2 Features

We extract a total of 480 features described in previous works for translation QE (Blatz et al. 2004; Ueffing and Ney 2007; Sanchis et al. 2007). Some of these features are highly-correlated, for example, we consider both the translation probability and the

\(^3\) http://statmt.org/wmt12/translation-task.html.
perplexity given by a language model. As described in Sect. 3.1, working with such redundant features involves several learning issues. However, these inherent learning issues make translation QE a task where DR techniques may lead to important improvements in prediction accuracy.

Following Specia et al. (2009b), we consider both black-box and glass-box features. On the one hand, black-box features (B) can be extracted given only the input sentence and the translation produced by the MT system, i.e. the source and target sentences, and possibly additional monolingual or parallel data. On the other hand, glass-box (G) features may also depend on some aspect of the translation process.

We distinguish between sentence- and subsequence-based features. Sentence-based features consider the translated sentence as an atomic unit and represent the translation as a whole. In contrast, subsequence-based features consider the translation as a sequence, and are computed by combining the feature scores of the subsequences (words or sequences thereof) contained in each translation.

4.2.1 *Sentence-based features*

- Source and translation lengths, and their ratio (B, 3 features).
- Source and translation probabilities, probabilities divided by length, and perplexities computed by language models of order 1–5 (B, 30 features).
- Translation probability, probability divided by translation length, and perplexity computed by language models of order 1–5 trained on the complete 1000-best file, and in the particular 1000-best translations of each source sentence (G, 3 indicators \(\times\) 5 orders \(\times\) 2 training corpora = 30 features).
- Average length of the 1000-best translations, vocabulary size of the 1000-best translations divided by average length, and 1000-best vocabulary size divided by source length (G, 3 features).
- Proportion of death nodes in the decoding search graph. (G, 1 feature)
- Number of source phrases of sizes one to six used in decoding (G, 6 features).
- Number and average size of the alternative translations considered in decoding for source phrases of sizes one to six (G, 12 features).

4.2.2 *Subsequence-based features*

We represent each subsequence feature by five sentence-level indicators: the average value of the subsequence scores in the translation, and the percentage of scores belonging to each frequency quartile. Each method represents a different approach to summarize the subsequence scores. The average value is a rough indicator that measures the “middle” value of them, while the quartile percentages are more fine-grained indicators that denote how spread out the scores are. We compute the following features for subsequences of sizes 1–4:

- Number of translation options for each source word in a Model-1 lexicon trained on the translation task data (B, 1 \(\times\) 5 = 5 features).

---

4 Frequency quartiles were computed on the training data of the shared translation task.
– Frequencies of source sentence subsequences in the training data of the translation task (B, 4 sizes × 5 = 20 features).
– Confidence score of each translation word computed by a Model-1 lexicon as in Ueffing and Ney (2007) (B, 1 × 5 = 5 features).
– Posterior probabilities of translation subsequences computed on the 1000-best translations (Ueffing et al. 2003). We follow Sanchis et al. (2007) and use four different criteria to align the subsequences of the translation to the subsequences of the alternative 1000-best translations, and three different weighting schemes to score each alignment. The accumulated score of the alignments of each subsequence is normalized to obtain the posterior probability of the subsequence (B, 4 sizes × 4 criteria × 3 weightings × 5 = 240 features).
– Confidence scores of the translation subsequences computed from the corresponding posterior probabilities by a smoothed naïve Bayes classifier as in Sanchis et al. (2007). We used three position correctness criteria to automatically generate the reference correctness labels required to train the classification model (B, 4 sizes × 3 criteria × 5 = 60 features).

We also compute the number of words in the translation with zero (<10⁻⁷) confidence according to the Model-1 lexicon (B, 1 feature), the number of source subsequences that do not appear in the training data of the translation task (B, 4 sizes = 4 features), the number of translation subsequences with zero (<10⁻⁷) posterior probability (B, 4 sizes × 4 criteria × 3 weightings = 48 features), and the number of translation subsequences classified as correct by the naïve Bayes classifier (B, 4 sizes × 3 criteria = 12 features).

4.3 Machine learning models

Now, we describe the particular learning models (Mθ in Sect. 2) tested in the experiments. We use the WEKA (Hall et al. 2009) package to estimate the values of the free parameters θ that best fit training data.

4.3.1 Linear regression

Linear regression assumes a linear relationship between the prediction value yᵢ and the vector of features xᵢ which is modeled by a vector of weights θᵀ = (θ₁, ..., θₘ). Formally, linear regression models take the form of a set of equations:

\[ yᵢ = θ₁xᵢ₁ + \cdots + θₘxᵢₘ + εᵢ, \quad i = 1, \ldots, n \] (10)

where \( n \) is the number of training samples, \( m \) is the number of features, and \( εᵢ \) are zero-centered Gaussian error variables. Often all equations are stacked together and written in matrix form:

\[ y = Xθ + ε \] (11)
The most common technique to estimate the free parameters $\theta$ of linear models is known as least squares estimation. This method minimizes the sum of squared errors, and leads to a closed-form expression for the optimum values of $\theta$:

$$\hat{\theta} = (X^T X)^{-1} X^T y$$

(12)

Additionally, different regularization techniques are usually implemented to prevent ill-posed learning problems when multicollinearity is present. Regularization techniques deliberately introduce bias into the estimation of $\hat{\theta}$ to penalize complex models. In the experiments, we used ridge and LASSO regression (Tibshirani 1996). Both methods constrain the norm of the parameter vector ($L^2$-norm ridge and $L^1$-norm LASSO) to be lower than a given value $\gamma$.

4.3.2 Support vector machines

In practice, few natural phenomena exhibit a linear relationship between their explanatory variables $x$ and the corresponding dependent variable $y$. Thus, linear regression cannot adequately describe such nonlinear phenomena.

Support vector machines (SVMs) (Cortes and Vapnik 1995) are a class of machine learning models that, as linear regression, assume a linear relationship between $X$ and $y$. However, prior to any calculation, SVMs project the data into an alternative space. This projection, defined by a kernel function $\phi(x)$, may be nonlinear; thus, though a linear relationship is learned in the projected feature space, this relationship may be nonlinear in the original input space. Choice of the kernel determines whether the resulting SVM is a polynomial regressor, a two-layer neural network, a radial basis function machine, or some other learning machine.

The linear relationship is estimated as a regularized ($L^2$-norm) optimization problem. In contrast to linear regression, the SVM model depends only on a subset of the training data, because the cost function for building the model does not care about those training samples that already lie within a given margin. There exist several specialized algorithms for solving the quadratic programming problem that arises. For example, sequential minimal optimization (Platt 1999) breaks the problem down into 2-dimensional sub-problems that can be solved analytically.

Preliminary experiments studying different kernels showed that radial basis kernel obtained among the best results and additionally was easier to train than other kernels such as polynomial kernels. Therefore, in the experimentation we used SVMs with a radial basis kernel.

4.3.3 Regression trees

Typical regression models, such as linear regression or SVMs, are global. In other words, there is a single predictive formula holding over the entire data-space. When the data has lots of features which interact in complicated, nonlinear ways, assembling a single global model can become a very difficult problem. An alternative regression
approach is to recursively partition the data-space into smaller regions, until they are simple enough to fit elemental models to them.

Regression trees use a tree structure to represent such a recursive partition. Each of the terminal nodes of the tree represents a region of the partition, and has attached to it a simple model which applies in that region only. We start at the root node of the tree, and ask a sequence of questions about the features. The interior nodes are labeled with questions, and the edges between them are labeled with the answers. Typically, each question refers to only a single feature, and has a yes or no answer, e.g., “Is Horsepower > 50?” or “Is GraduateStudent == FALSE?”. Features can be of different types (continuous, discrete, categorical, etc), and more-than-binary questions can be done, but these can always be accommodated as a larger binary tree. Figure 4 shows an example of a regression tree using gaussian normal distributions to model the data on each partition.

Once we fix the tree structure, local models are completely determined, and easy to find, so all the effort should go into finding a good tree structure, which is to say into finding a good partitioning of the data-space. In our experiments, we specifically use M5 regression tree (Quinlan 1992) because one of the best submissions to the 2012 QE task (Callison-Burch et al. 2012) used such tree model.

5 Experiments

5.1 Methodology

We extracted the 480 features described in Sect. 4.2 for each of the automatic translations in the evaluation data of the QE task. As a result, we obtained a training and a test set of 480-dimensional real vectors with 1,832 and 422 samples respectively. All features were standardized by subtracting the feature mean from the raw values, and dividing the difference by the standard deviation.

Then, we carried out an exhaustive experimentation to test the different DR methods described in Sect. 3, and to study how their use affect the prediction performance of the different learning models presented in Sect. 4.3. We tested all 18 combinations of
a DR method and a learning model in a series of two-step experiments as depicted in Fig. 1. Since we did not know the optimum size $r$ of the reduced feature set (see Sect. 3.2), each experiment involved several trains of the model with reduced feature sets of different sizes. For each size, we performed a cross-validation training with ten randomly-chosen data splits to learn the meta-parameters of the models, e.g. the $\gamma$ parameter of ridge regression.

5.2 Evaluation criteria

Since we view DR as a way to build robust prediction models, we evaluated each DR method by the prediction accuracy of the regression models trained on the corresponding reduced feature sets. The performance of a regression model is usually measured by the average error of the predictions $\hat{y} = \{\hat{y}_1, \ldots, \hat{y}_n\}$ with respect to the actual scores $y = \{y_1, \ldots, y_n\}$. Specifically, we compute the root mean squared prediction error (RMSPE) as in Specia et al. (2009b):

$$\text{RMSPE}(y, \hat{y}) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

where $n$ is the number of test samples. RMSPE quantifies the average deviation of the estimation with respect to the expected score. I.e. the lower the value, the better the performance of the learning model.

5.3 Cross-validation training results

We now present the results for cross-validation training experiments. The conclusions were similar for all learning models. Thus, to keep the presentation clear, we only show RMSPE results using SVMs as learning model. Figure 5 shows SVMs cross-validation RMSPE for the different DR methods presented in Sect. 3.

The results of the four feature selection methods were very close, and all of them slightly outperformed the baseline SVM model trained with the whole 480-dimensional feature set (0.71 RMSPE). RFS, VIP, and feature importance in regression (FIR) obtained virtually the same results. Their performance improved as more features were selected, and they required to select above 100 features to reach their top performance. Then, as more features were selected their results slowly converged to the performance of the baseline model. Since these methods do not take into account the correlations that may exist between the features, their reduced feature sets were highly-redundant; which explains the large number of features they needed to stabilize. In contrast, GFSobtained great improvements with few features. However, its higher computational complexity complicates its practical deployment; reason why we carried out experiments only up to 30 features. Nevertheless, with only these 30 features it was able to equal the performance of the baseline model trained on the original 480 features.
Regarding the two feature extraction methods, they exhibited important differences in performance. PCA projection (PCA-P) obtained worse results than the four feature selection methods, moreover it did not even improve the results of the baseline model. PCA-P reached its top performance when \( \sim 120 \) principal components were generated, and it slightly deteriorated as the number of features increased. In contrast, PLSR projection (PLS-P) obtained much better results consistently outperforming PCA-P and all feature selection methods. Moreover, with only five latent variables PLS-P was able to outperform the baseline SVM model trained with 480 features, and it only required 44 features to reach its top performance. Additionally, the performance difference observed between the best result of PLS-P and the rest of the DR methods was significant with a probability of improvement of 95 % according to a pair-wise bootstrap analysis (Bisani and Ney 2004). These results indicate that PLS-P generates more “information-dense” features that constitute a better summary the original high-dimensional feature set.

Although results in Fig. 5 are representative for all learning models, we observed important differences in the stability of the learning curves of the different models. Figure 6 displays training cross-validation results for linear ridge regression using PCA-P and PLS-P as DR methods. We present results only for these two DR methods for simplicity. Since the baseline ridge model (trained with the original 480 features) obtained a dreadful RMSPE of 16.73, we present results for two alternative linear regression baselines: a LASSO regression model also trained with all the original 480 features, and for the predictions directly generated by the PLSR model according to Eq. (4). In contrast to the results for SVMs, we now obtained rougher learning curves with large performance variations, particularly as we increased the number of features. However, the proposed two-step training procedure (see Fig. 1) partially addresses this problem. This is exemplified in the comparison between PLSR and PLS-P. Both methods use a linear model to predict the quality scores from the projected data, however PLS-P obtains a much smoother learning curve than PLSR. Finally, we could extract the same conclusion as for SVMs: among all the tested DR methods, PLS-P is
the best performing one allowing us to improve the performance of even sophisticated regularized models such as SVMs or linear LASSO regression.

These results show that the proposed two-step training is an efficient procedure to deal with noisy and correlated input features, and it can outperform models such as LASSO regression and PLSR that integrate DR in their formulation.

5.4 Blind test results

Next, for each combination of a DR method and a learning model, we built a new model using the full training set and the best configuration (size of the reduced feature set, and values of the meta-parameters of the learning model) observed in the corresponding cross-validation experiments. Then, we reduced the test set to the optimal dimension estimated by cross-validation training, and tested the performance of the new trained model for the reduced test set. Table 1 displays these results. In contrast to the previous cross-validation experiments, results on the test set were quite different for the three learning models. While for SVMs, the use of DR improved the performance of the baseline model trained on the 480 original features, no improvement was obtained at all for linear ridge regression, or for regression trees. This was quite a surprising result. Given the large improvements over the baseline obtained in the cross-validation experiments, we expected to obtain similar improvements over baseline in test.

To better understand these results, we carried out a multivariate Hotelling’s two-sample T-squared test (Hotelling 1931; Anderson 1958) to study the possible differences that may exist between the training and test feature sets. The objective of such tests is to determine whether two samples, in our case the training and test sets, have been sampled from the same population or not. The result of the test indicated that there were a statistically significant difference between the two feature sets ($p < 0.01$), and thus they seemed to come from different populations. Since the training and test
translations come from a similar news domain (Callison-Burch et al. 2012), we hypothesize that the difference between the feature sets was mainly due to the specific chosen features. In fact, results of individual Student’s two-samples t tests for each feature showed that 260 of the 480 extracted features were significantly different ($p < 0.01$) between training and test. For example, the number of words with zero posterior probability is significantly different between the samples in training ($\mu = 1.7$, $\sigma = 1.39$) and test ($\mu = 0.90$, $\sigma = 0.80$).

In addition to the relatively small number of training samples (1,832), this mismatch between the distribution of the features values in the training and test sets may be the explanation for the unintuitive results displayed in Table 1, compared to the cross-validation results in Figs. 5 and 6 where PLS-P largely improved Baseline. DR methods obtain a reduced feature set based on the training set, thus, if the training set is not representative of the test set, as proved by the Hotteling’s test, the computed reductions cannot be adequate for test. Also, the fact that SVMs actually improved baseline test results when DR methods were used can be explained by the fact that SVMs are more complex models than ridge regression and regression trees. SVMs performance is more heavily penalized due to the lack of data. Thus, we hypothesize that the use of reduced feature sets, even if they are inadequate, allows to improve SVMs performance.\(^5\)

Despite these problems, Table 1 shows that PLS-P was the top-performing DR method for linear regression and SVMs. However, for regression trees, all methods obtained similar results. This fact indicates that regression trees were not able to fully exploit the more “information-dense” features generated by PLS-P. Since these “information-dense” features are the combination of several of the original features, we hypothesize that they are also more difficult to be partitioned into regions to create the tree structure of the model. Nevertheless, even in this pessimistic setting PLS-P generated reduced sets of features that performed similarly as the original 480 features. We consider that, given the cross-validation results in Sect. 5.3, larger performance improvements could be expected whenever an adequate set of features, and/or a large enough training set are provided.

Additionally, since the time required to train the model and to perform the prediction are directly related to the number of features, an additional advantage of DR methods is that they can improve the practical deployment of QE technology by reducing training/test time. For example, training an SVM model (including meta-parameter optimization) using the original 480 features typically required $\sim 30$ h in our test machine, while the training time using the optimal 44 latent variables extracted by PLS-P was below 3 h.

5.5 Feature analysis

We perform a final analysis on the features that contribute more to create the reduced feature sets. For feature selection methods, we simply looked for the most frequently selected features. For PCA-P and PLS-P, that combine the original features into new features (the principal components and the latent variables respectively) by a matrix

\(^5\) Few features imply few parameters to be estimated with the same amount of data.
Table 1 Prediction results (RMSPE) on the test set for the different DR methods and learning models under study

<table>
<thead>
<tr>
<th>Method</th>
<th>Original features</th>
<th>RFS</th>
<th>GFS</th>
<th>VIP</th>
<th>FIR</th>
<th>PCA-P</th>
<th>PLS-P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ridge regression</td>
<td>480 0.79</td>
<td>69 0.83</td>
<td>22 0.82</td>
<td>67 0.83</td>
<td>82 0.83</td>
<td>57 0.83</td>
<td></td>
</tr>
<tr>
<td>Support vector machines</td>
<td>480 0.97</td>
<td>162 0.84</td>
<td>22 0.83</td>
<td>126 0.83</td>
<td>136 0.82</td>
<td>122 0.81</td>
<td></td>
</tr>
<tr>
<td>Regression trees</td>
<td>480 0.87</td>
<td>72 0.91</td>
<td>16 0.89</td>
<td>57 0.88</td>
<td>71 0.86</td>
<td>31 0.90</td>
<td></td>
</tr>
<tr>
<td>NF</td>
<td>480 0.78</td>
<td>44 0.78</td>
<td>9 0.88</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NF denotes the number of features of the reduced test sets. Best results for each learning model are displayed in bold. As a comparison, the result for a linear LASSO regression model was 0.82 RMSPE.

transformation ($P$ in Eqs. (2) and (6)), we computed the contribution of each feature by summing up the absolute value of the scores in the corresponding column of $P$. We then can highlight the following features:

- Source and translation lengths and language model probabilities.
- Vocabulary of the 1000-best translations divided by their average length.
- Number of source phrases of size one used in decoding.
- Number of source phrases used in decoding.
- Frequencies of source subsequences (sizes 1–4).†
- Posterior probabilities of translation subsequences (sizes one and two).†
- Probability of the translation subsequences (sizes one and two) by a naïve Bayes’ classifier.†

Additionally, for the subsequence-based features (marked with †) the most important sentence-level indicators were specifically the average value of the feature, and the number of subsequences in the first and fourth quartile.

Despite this general result, we observed slight differences in the importance of each feature according to the different methods. For example, the simple RFS method tended to add lots of similar features, such as the posterior probabilities of the target subsequences, which independently are quite informative but together are highly redundant. In contrast, the more computationally complex GFS method selected only one or two features that represent all features of the same type.

6 Summary and future work

We have proposed two novel DR methods based on PLSR and compared them against several DR methods previously used in the QE literature. The DR methods under consideration can be classified by their theoretical background: statistical multivariate analysis or heuristic methods, or by how they perform the reduction: feature selection or feature extraction methods. Moreover, we have studied how DR affect the prediction performance of different learning models.

We have evaluated each DR method by the prediction performance of the learning models trained on the corresponding reduced feature set. This quality measure has the
advantage of automatic evaluation, and, using identical pipelines to train the models, it allows us to accurately compare the different DR methods. The key results of the experiments are as follows:

- Feature extraction methods can outperform feature selection methods.
- Methods based on multivariate analysis can outperform heuristic methods.
- To obtain a good prediction performance, DR methods have to take into account the scores to be predicted.
- The performance-wise ranking of the DR methods is to a great extent independent of the chosen learning model.
- However, for simple models such as linear regression the use of some DR methods may result in erratic learning curves.

One of the proposed DR methods, PLS-P, can be seen as a summary of the conclusions: a feature extraction method based on multivariate analysis that takes into account the values to be predicted to perform the reduction. Thus, it consistently obtained the best results in the cross-validation training experiments. Additionally, the unintuitive results observed in test (where PLS-P did not improve baseline) can be explained by a difference between the distribution of the features in training and test. The use of statistical tests to detect this problem is then a necessary tool to build robust QE systems.

As future work, we plan to explore additional feature selection methods based on redundancy minimization and relevancy maximization, and new feature extraction methods based on nonlinear projections, and also to integrate statistical tests over the features as a preliminary step to filter out problematic features. Additionally, we also plan to investigate automatic techniques to estimate the internal dimension $r$ of the problem, interactions between the features, and outliers detection methods to efficiently use of the (usually) scarce training data.

**Acknowledgments** This work supported by the European Union Seventh Framework Program (FP7/2007-2013) under the CasMaCat project (grants agreement no. 287576), by Spanish MICINN under TIASA (TIN2009-14205-C04-02) project, and by the Generalitat Valenciana under grant ALMPR (Prometeo/2009/014).

**References**


Dimensionality reduction methods


In: Proceedings of the seventh workshop on statistical machine translation, pp 104–108


Pearson K (1901) On lines and planes of closest fit to systems of points in space. Philos Mag 2:559–572


Attachment B

Task 3.1


Empirical study of a two-step approach to estimate translation quality.
Empirical Study of a Two-Step Approach to Estimate Translation Quality

Jesús González-Rubio†, J. Ramón Navarro-Cerdán‡, Francisco Casacuberta†

†D. de sistemas informáticos y computación ‡Inst. Tecnológico de Informática
Universitat Politècnica de València, camino de Vera s/n, 46022 Valencia, Spain
jegonzalez@dsic.upv.es, jonacer@iti.upv.es, fcn@dsic.upv.es

Abstract
We present a method to estimate the quality of automatic translations when reference translations are not available. Quality estimation is addressed as a two-step regression problem where multiple features are combined to predict a quality score. Given a set of features, we aim at automatically extracting the variables that better explain translation quality, and use them to predict the quality score. The soundness of our approach is assessed by the encouraging results obtained in an exhaustive experimentation with several feature sets. Moreover, the studied approach is highly-scalable allowing us to employ hundreds of features to predict translation quality.

1. Introduction
Despite an intensive research in the last fifty years, machine translation (MT) systems are still far from perfect [1]. Hence, a desirable feature to improve their practical deployment is the capability of predicting at run-time\(^1\) the reliability of the generated translations. This task, referred to as quality estimation [2] (QE), is becoming a crucial component in practical MT systems [3, 1]. For instance, to decide if an automatic translation is worth being supervised by a translator or it should be translated from scratch. Quality can be estimated at the word, sentence, or document level. Here, we focus on the estimation of sentence-level quality.

Sentence-level QE is typically addressed as a regression problem [4, 2]. Given a translation (and other sources of information), a set of features is extracted and used to build a model that predicts a quality score. This point of view provides a solid framework within which accurate predictors can be derived. However, several problems arise when applying this approach to predict the quality of natural language sentences. For example, while the concept of translation quality is quite intuitive, the definition of features that reliably account for it has proven to be elusive [4, 1]. Thus, in practice, feature sets contain a large number of noisy, collinear and ambiguous features that hinder the learning process of the regression models, e.g., due to the “curse of dimensionality” [5].

An interesting approach to overcome these problems is to conceive QE as a two-step problem. In a first step, a dimensionality reduction (DR) process strips out the noise present in the original features returning a reduced set of (potentially new) features. Then, the actual quality prediction is made from this reduced set. Typically, QE systems reduce the dimensionality by simply selecting a subset of the original features according to some relevance measure [2, 6, 7]. However, a recent study [8] have shown that DR methods based on a projection of the original features may be more effective. The intuition for this is clear, the new features extracted by a projection-based DR method summarize the “information” contained in the all the original features, in contrast, the information contained in the features discarded by a feature selection method is inevitably lost.

We work on the foundations of [8] and provide an exhaustive empirical study of the most successful QE approach described there. This approach (§2) involves a DR method based on a partial least squares [9] (PLS) projection of the data and a support vector machine [10] (SVM) as prediction model. We test this two-step QE approach in a wide variety of conditions (§3) where we compare the performance of PLS to the most widely-used projection-based DR approach, namely principal component analysis [11] (PCA). Empirical results (§4) show that PLS consistently outperformed PCA in prediction accuracy and feature reduction ratio. This latter result is particularly interesting because it allows us to apply QE in scenarios with strict temporal restrictions, for instance interactive machine translation tasks.
2. A Two-Step QE Approach

The method proposed in [8] divide QE ($\mathbb{R}^m \rightarrow \mathbb{R}$) into two sub-problems. First, the original $m$-dimensional set of features is projected into a new $r$-dimensional set of features ($\mathbb{R}^m \rightarrow \mathbb{R}^r$, $r < m$). Then, this reduced feature set is used to build a regression model that predicts the actual quality scores ($\mathbb{R}^r \rightarrow \mathbb{R}$). Figure 1 shows a diagram of this two-step training methodology. Next sections describe how to solve these two sub-problems.

2.1. Dimensionality Reduction

Typical approaches to reduce a set of noisy features involve the use of principal components analysis [11] (PCA). PCA projects the set of features into a set of principal components (PCs) where each PC explains the variability of the features in one principal direction. As a result, these PCs contain almost no redundancy but, since the PCA transformation ignores the quality scores to be predicted, they do not necessarily have to be the best features to perform the prediction.

Instead, we implement a feature reduction technique based on partial least squares [9] (PLS). PLS extracts a ordered set of latent variables (LVs) such that each of them accounts for the maximum possible co-variability between the features and the scores to be predicted under the constraint of being uncorrelated with previous LVs. That is, LVs are uncorrelated as PCs do, and additionally, they explain as much as the variability in the quality scores as possible. As a result, usually few LVs than PCs are required to reach a certain accuracy.

Let $\{x_i,y_i\}_{i=0}^n$ be a corpus with $n$ samples where $x_i$ are $m$-dimensional feature vectors, and $y_i$ are quality scores. This corpus can be written in matrix form where symbol $\top$ indicates the transpose of a matrix or vector:

$$X = \begin{pmatrix} x_1 \top \\ \vdots \\ x_n \top \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{pmatrix}, \quad y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \quad (1)$$

Then, PLS constructs the following linear model where $b$ is a vector of regressor coefficients, and $f$ is a vector of zero-centered Gaussian errors:

$$y = Xb + f \quad (2)$$

PLS also defines two PCA-like transformations ($P$ for $X$, and $q$ for $y$) with $E$ and $f$ being the corresponding errors, and a linear relation $R$ linking both blocks:

$$X = TP^\top + E \quad y = Uq^\top + f \quad U = TR \quad (3)$$

where matrices $T$ and $U$ are the projections of $X$ and $y$ respectively. The value of the regression coefficients $b$ are finally computed as [9]:

$$b = Rq^\top \quad \text{where} \quad R = W(P^\top W)^{-1} \quad (4)$$

The columns in matrix $T$ are the LVs of $X$. Each of these LVs accounts for the maximum co-variability between $X$ and $y$ not explained by previous LVs. Therefore, similarly as it is usually done with PCA, we can collect the first $r$ LVs and use them to represent the original $m$-dimensional feature set. Given that $r < m$, and that the LVs are orthogonal by definition, we are simultaneously addressing the “curse of dimensionality” and reducing the noise present in the original features. Moreover, the reduced set also explains most of the variability in the quality scores to be predicted.

In the experiments, we used the pls library [12] of the R toolkit. The dimension of the reduced set $r$ is one of the meta-parameters of the studied QE approach.

PLS can be directly used as a predictor model (see Equation (2)). However, its simple linear model is not adequate to model the nonlinear relation that may exist between the features and the quality scores. Preliminary experiments confirmed this intuition.

2.2. Prediction Model

Once the reduced feature set is extracted, a support vector machine (SVM) is used predict the quality scores ($\mathbb{R}^r \rightarrow \mathbb{R}$). We choose SVMs because they have shown good prediction accuracy and robustness when dealing with noisy data in a number of tasks.
SVMs, first proposed for classification problems by Cortes and Vapnik [10], are a class of machine learning models that are able to model nonlinear relations between the features and the values to be predicted. Prior to any calculation, SVMs project the data into an alternative space. This projection, defined by a kernel function, may be nonlinear; thus, though a linear relationship is learned in the projected feature space, this relationship may be nonlinear in the original space. Following previous works on QE [2], we use SVMs with a radial basis kernel as implemented in the LibSVM package [13]. Values $\gamma$, $\epsilon$, and $C$ are additional metaparameters to be optimized.

3. Experimental Setup

3.1. Corpus

We computed quality scores for the English-Spanish news evaluation data used in the QE task of the 2012 workshop on statistical MT [1] (WMT12). The Spanish translations were generated by a phrase-based MT system trained on the Europarl and News Commentaries corpora as provided for the WMT12 translation task. Evaluation data contains 1832 translations for training, and 422 translations for test. The quality score of each translation $y \in \mathbb{R} | 1 \leq y \leq 5$ is computed as the average of the scores given manually by three different experts in terms of post-editing effort:

5: The translation requires little editing to be publishable
4: 10%–25% of the translation needs to be edited
3: 25%–50% of the translation needs to be edited
2: 50%–70% of the translation needs to be edited
1: The translation must be translated from scratch

3.2. Feature Sets

We conducted QE experiments with several feature sets submitted to the WMT12 QE task\(^2\). These sets allow us to test our approach under a wide variety of conditions. Table 1 displays, for each set, the number of features, whether or not the features are result of a feature selection process, the percentage of features in the training partition that are collinear with the rest of features (redundancy), and the percentage of features in the training partition that are constant, and hence, irrelevant to perform the prediction. We estimated the degree of collinearity of each feature by its condition number considering a value above 100 to denote collinearity [14].

<table>
<thead>
<tr>
<th>Name</th>
<th>#features</th>
<th>feature selection?</th>
<th>collinear</th>
<th>constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCU-SYMC</td>
<td>308</td>
<td>no</td>
<td>34.6%</td>
<td>0.7%</td>
</tr>
<tr>
<td>LORIA</td>
<td>49</td>
<td>yes</td>
<td>12.2%</td>
<td>0.0%</td>
</tr>
<tr>
<td>SDLNW</td>
<td>15</td>
<td>yes</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>TCD</td>
<td>43</td>
<td>no</td>
<td>18.6%</td>
<td>0.0%</td>
</tr>
<tr>
<td>UEDIN</td>
<td>56</td>
<td>no</td>
<td>5.5%</td>
<td>1.8%</td>
</tr>
<tr>
<td>UPV</td>
<td>497</td>
<td>no</td>
<td>54.3%</td>
<td>6.8%</td>
</tr>
<tr>
<td>UU</td>
<td>82</td>
<td>no</td>
<td>7.5%</td>
<td>2.5%</td>
</tr>
<tr>
<td>WLV-SHEF</td>
<td>147</td>
<td>no</td>
<td>21.0%</td>
<td>2.7%</td>
</tr>
</tbody>
</table>

Table 1: Main properties of the feature sets. We estimated the collinearity with the condition number [14].

We consider the feature sets as independent corpora provided by an external agent. Hence, and due to space limitations, we only provide a brief description of each set; an exhaustive description can be found in the corresponding citation. Many of the sets include the 17 baseline features provided by the organizers [1].

DCU-SYMC: [15] 308 features including features based on latent Dirichlet allocation; source grammatical features from the TreeTagger part-of-speech tagger, an English grammar, the XLE parser, and the Brown re-ranking parser; and target TreeTagger features.

LORIA: [6] 66 features including the baseline features, and features based on cross-lingual triggers.

SDLLW: [7] 15 features exhaustively selected from an original set of 45 features: the 17 baseline features, 8 features based on decoder information, and 20 features based on n-gram precisions and word alignments.

UDS: [16] 43 features including the baseline features, and features based on similarity measures with respect to the Google n-grams data set.

UEDIN: [17] 56 features including the baseline features and features based on named entities, morphological information, lexicon probabilities, word-alignments, and sentence and n-grams similarities.

UPV: [18] 497 features including the baseline features and features based on word-level quality scores.

UU: [19] 82 features computed from syntactic, constituency, and dependency trees.

WLV-SHEF: [20] 147 features based on part-of-speech information, subject-verb agreement, phrase constituency and target lexicon analysis.

3.3. Experimental Methodology

For each feature set, a QE system was built following the two-step methodology described in §2 and depicted

\(^2\)These are available in https://github.com/lspecia/QualityEstimation.
Table 2: RMSE and number of LVs obtained by cross-validation for the different feature sets. In parenthesis, we show the number of LVs as a percentage of the original features. Baseline denotes a system trained with the whole feature set. PCA denotes a system built using PCA instead of PLS. Best mean RMSE values and lowest number of features are displayed boldface. Asterisks denote a statistically better result than both the other two systems (95% confidence).

<table>
<thead>
<tr>
<th>Feature set</th>
<th>Baseline</th>
<th>PCA</th>
<th>Our approach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>#features</td>
<td>RMSE</td>
</tr>
<tr>
<td>DCU-SYMC</td>
<td>0.71 ±0.02</td>
<td>308</td>
<td>0.70 ±0.02</td>
</tr>
<tr>
<td>LORIA</td>
<td>0.72 ±0.03</td>
<td>49</td>
<td>0.75 ±0.01</td>
</tr>
<tr>
<td>SDLW</td>
<td>0.67 ±0.02</td>
<td>15</td>
<td>0.67 ±0.02</td>
</tr>
<tr>
<td>TCD</td>
<td>0.76 ±0.01</td>
<td>43</td>
<td>0.74 ±0.02</td>
</tr>
<tr>
<td>UEDIN</td>
<td>0.72 ±0.03</td>
<td>56</td>
<td>0.71 ±0.02</td>
</tr>
<tr>
<td>UPV</td>
<td>0.74 ±0.02</td>
<td>497</td>
<td>0.69 ±0.02</td>
</tr>
<tr>
<td>UU</td>
<td>0.72 ±0.02</td>
<td>82</td>
<td>0.68 ±0.02</td>
</tr>
<tr>
<td>WLV-SHEF</td>
<td>0.71 ±0.02</td>
<td>147</td>
<td>0.71 ±0.02</td>
</tr>
</tbody>
</table>

where \( n \) is the number of samples. RMSE quantifies the average error of the estimation with respect to the actual quality score. I.e., the lower the value, the better the performance of the QE system.

Additionally, we perform different significance tests for the reported RMSE results. On the one hand, we obtain confidence intervals for the averaged cross-validation test results with Student’s t-tests [21]. On the other hand, we use paired bootstrap re-sampling [22] to measure the significance of the RMSE differences observed between the different methods in the test sets.

### 4. Results

We now present the results of the empirical evaluation of the studied QE approach. First, we predicted quality scores for each of the feature sets described in §3.2. Then, we took advantage of the scalability of the studied QE approach using jointly all the features in those sets to perform the prediction.

#### 4.1. Results for the Individual Feature Sets

Table 2 shows the cross-validation results (RMSE and number of LVs) obtained for the different feature sets. As a comparison, we present results for SVMs trained with all the features in each set (Baseline), and for systems built using the widespread PCA instead of PLS in the studied two-step training methodology.

We can observe that the studied approach consistently obtained equal or better prediction accuracy (RMSE) than the baseline systems. Additionally, the number of LVs used to build the final SVMs was much lower than the number of original features. The size of the reduced sets varied between two thirds and one tenth in Figure 1. All features were standardized by subtracting the feature mean from the raw values, and dividing the difference by the corresponding standard deviation.

The number of LVs (\( r \)) was optimized by ten-fold cross-validation using the training partitions (1832 samples). Each cross-validation experiment took eight folds for training (dev-train), one held-out fold for development and the other held-out fold for test (dev-test). We used the dev-train folds to estimate a PLS model. Then, this model was used to extract the \( r \) LVs of dev-train, and of the separated development fold and the dev-test fold. Next, we used the reduced dev-train folds to estimate an SVM model, the reduced development fold to optimize the SVM meta-parameters (\( \gamma \), \( \epsilon \), and \( C \)), and the reduced dev-test fold to test the optimized SVM model. The result of each complete cross-validation experiment was the averaged prediction accuracy on the ten held-out dev-test folds. The number of LVs was selected to optimize this average accuracy.

Once the number of LVs was fixed, we built a new prediction model with the whole training partition optimizing the SVM meta-parameters (\( \gamma \), \( \epsilon \), and \( C \)), and cross-validation. Finally, we used this optimized SVM model to predict the quality scores of the test partitions (422 samples).

#### 3.4. Assessment Criteria

We measure the accuracy of a QE system by the deviation of its predictions \( \hat{y} = \{y_1, \ldots, y_n\} \) respect to the reference quality scores \( y = \{y_1, \ldots, y_n\} \). Following previous QE works [2, 1], we calculate the root-mean-squared error (RMSE) between them:

\[
\text{RMSE}(\hat{y}, y) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}
\]

(5)

where \( n \) is the number of samples. RMSE quantifies the average error of the estimation with respect to the actual quality score. I.e., the lower the value, the better the performance of the QE system.
of the original features. This reductions are roughly related with the percentage of collinear and constant features in Table 1. In comparison to PCA, the studied DR technique, PLS, was able to obtain better prediction accuracy with less features. Usually, the number of LVs is less than half the number of PCs.

These result indicate that the studied QE approach was indeed able to strip out the noise present in the original features. Additionally, the DR technique based on PLS projections showed a better performance (both in prediction accuracy and reduction ratio) that the commonly-used PCA. As a result, even for highly-engineered features sets such as SDLLW [7] that contain no collinear or redundant features, our approach was able to obtain a more compact feature set (10 LVs) that still retained the prediction potential of the whole original set (15 features).

Next, to better understand the influence of the number of LVs in the results, Figure 2 displays the prediction accuracy as a function of the number of features for two prototypical feature sets: the highly noisy and collinear UPV set, and the low redundant SDLLW set.

The prediction accuracy of our method for the UPV feature set (left panel in Figure 2) rapidly improved as more LVs were considered. With only 5 LVs, prediction accuracy already statistically outperformed the baseline (497 features), and it reached its top performance for 58 LVs. As we considered more LVs (for simplicity the graph only shows up to 100 LVs), prediction error steadily increased which was indicative of over-training. Thus, we chose 58 as the optimum number of variables for the UPV set. The quite large RMSE reduction respect to the baseline can be explained by the ability of our approach to strip out the great amount of noise present in the original UPV set, see Table 1. Regarding PCA, it was consistently outperformed by our approach and only slightly improved the RMSE score of the baseline system.

For the concise SDLLW feature set (right panel in Figure 2), our system showed approximately the same behavior: prediction accuracy rapidly improved up to a point from where the performance remains approximately stable. In this case, 10 was the optimal number of LVs. In contrast to the UPV set, our approach could not improve Baseline performance which is reasonable since SDLLW is a very clean set with no redundant or irrelevant features (see Table 1) that could hinder the learning process. Nevertheless, our method was able to obtain the same prediction accuracy as Baseline with only two thirds of the original features.

In a following experiment, we built QE systems with the whole training partitions and the optimal number of LVs estimated in the previous cross-validation experiments. The SVM meta-parameters ($\gamma$, $\epsilon$, and $C$) were optimized by standard cross-validation and the optimized models were used to predict the quality scores of the test partitions. Note that due to variations in the learning procedures, Baseline results may differ from those reported in the WMT12 QE task [1].

Table 3 displays, for each feature set, the RMSE obtained by our approach in the test partition. We also show baseline results for SVMs built with all the features in each set, and for systems that used PCA instead of PLS to reduce the dimensionality. RMSE confidence intervals for Baseline, PCA and our approach always overlapped but the observed differences were still statistically significant for a number of sets: for DCU-SYMC, Baseline obtained a statistically bet-
Table 3: RMSE and 95% confidence intervals of the predictions for the test partitions. Best mean results are displayed boldface. Asterisks denote a significant difference in performance (paired re-sampling, 95% confidence) respect to both the other two methods.

<table>
<thead>
<tr>
<th>Feature set</th>
<th>Baseline</th>
<th>PCA</th>
<th>Our approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCU-SYMC</td>
<td>0.87±0.07*</td>
<td>1.01±0.07</td>
<td>0.96±0.08</td>
</tr>
<tr>
<td>LORIA</td>
<td>0.84±0.06</td>
<td>0.87±0.06</td>
<td>0.85±0.06</td>
</tr>
<tr>
<td>SDLLW</td>
<td>0.76±0.05</td>
<td>0.77±0.05</td>
<td>0.76±0.05</td>
</tr>
<tr>
<td>TCD</td>
<td>0.82±0.06</td>
<td>1.00±0.05</td>
<td>0.83±0.06</td>
</tr>
<tr>
<td>UEDIN</td>
<td>0.86±0.06</td>
<td>0.85±0.05</td>
<td>0.86±0.05</td>
</tr>
<tr>
<td>UPV</td>
<td>0.82±0.06</td>
<td>0.83±0.05</td>
<td>0.78±0.05*</td>
</tr>
<tr>
<td>UU</td>
<td>0.81±0.05</td>
<td>0.81±0.05</td>
<td>0.82±0.06</td>
</tr>
<tr>
<td>WLV-SHEF</td>
<td>0.84±0.05</td>
<td>0.84±0.05</td>
<td>0.82±0.05*</td>
</tr>
</tbody>
</table>

Table 4: Percentage of the features in each feature set that have significantly different values in the training and test partitions. Significance computed by Student’s two-sample t-test (99% confidence).

<table>
<thead>
<tr>
<th>Feature set</th>
<th>Baseline</th>
<th>PCA</th>
<th>Our approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCU-SYMC</td>
<td>45.1%</td>
<td>UEDIN 48.1%</td>
<td></td>
</tr>
<tr>
<td>LORIA</td>
<td>24.5%</td>
<td>UPV 67.4%</td>
<td></td>
</tr>
<tr>
<td>SDLLW</td>
<td>73.3%</td>
<td>UU 38.8%</td>
<td></td>
</tr>
<tr>
<td>TCD</td>
<td>30.2%</td>
<td>WLV-SHEF 28.6%</td>
<td></td>
</tr>
</tbody>
</table>

This mismatch can be partially explained by the fact that the training and test partitions contain news texts of different years [1], but we still consider that the main issue is the size (only 1382 samples) of training partitions that did not adequately represent test partitions. However, both our approach and the baseline systems had to deal with this mismatch, so, why our method and PCA seemed to be more heavily penalized than Baseline?

The projection of the features is computed based on the training data. Thus, if the training partition is not representative of the test partition, the reduced feature sets will be projected in a “direction” that may penalize the prediction accuracy for the test set. That is, crucial information to predict the quality scores of the test partition may be stripped out. This drawback is common to any dimensionality reduction technique as exemplified by the also poor test results (Table 3) obtained by PCA.

The conclusion that can be extracted from these results is that the use of feature reduction implies a greater risk of over-training the prediction system. This effect particularly important if training data is scarce but it is mitigated as more training data is available. Thus, given the encouraging cross-validation results in Table 2, better prediction accuracy could be expected in test whenever an adequate training partition is provided.

Under the assumption that the original features can be computed in advance, a complimentary advantage of the studied two-step QE approach is that it allows us to build more time-efficient QE systems. Figure 3 displays the time required to build an SVM model (including meta-parameter optimization) and obtain the test predictions as a function of the number of features used to train the model. Specifically, we built QE systems with an increasing number of LVs extracted from the WLV-SHEF feature set. Each point in the figure is the average time of ten experiments. Results show how operating times increased with the number of LVs. For instance, the operating time of the baseline model trained with the original 147 features (0.84 RMSE) was ~200 seconds, while the operating time of the system
built with the 14 LVs extracted by PLS (0.82 RMSE) was only \( \sim 15 \) seconds which represents one order of magnitude less operating time. Hence, our approach is well-suited to be applied to scenarios, such as interactive MT [3], with strict temporal restrictions.

4.2. Exploiting the scalability of our approach

Results in the previous section have shown that the studied QE approach was able to extract the relevant prediction information from different sets of noisy features. We now take a further step in this direction and present results where all the features used in the previous experiments are joined together to create an extremely high-dimensional feature set from which to predict quality scores. This aggregated set, denoted by ALL, contains 1197 features for each translation; approximately 55\% of them being collinear with the rest.

Figure 4 shows cross-validation prediction accuracy (RMSE and 95\% confidence interval) of the studied QE approach as a function of the number of LVs. Again, we also display results for a baseline SVM model built using all the features, and for a system built using PCA instead of PLS. Our approach obtained a score of \( 0.45 \pm 0.01 \) RMSE with only 86 LVs. This result represents a 30\% reduction relative to the baseline RMSE calculated with 1197 features. Regarding PCA, it barely reached Baseline performance. These results indicate that our approach was able to exploit the information contained in the ALL set to improve prediction accuracy. In contrast, both Baseline and PCA were unable to adequately manage the huge number of noisy and collinear features. Additionally, the operating time of the Baseline systems was \( \sim 23 \) minutes, while it reduced to \( \sim 2 \) minutes when we used the optimal 86 LVs.

Test results were again quite disappointing: 1.4±0.1 RMSE of our approach versus 0.78 ± 0.06 RMSE of Baseline and 0.81 ± 0.07 of PCA. We hypothesize that the clearly worse result of our approach in this case was due to the larger number features. As more features are available, our system can generate more “specialized” LVs. Given that the training data does not adequately represents the test data (see discussion in §4.1), this better projection (as shown in Figure 4) actually hinders prediction accuracy in the test set.

5. Summary

We have described an empirical study of a two-step QE approach specifically designed to manage the noisy features usually derived from natural language sentences. This approach, first described in [8] implements a method based on PLS to extract, from the original features, the LVs that actually govern translation quality, and an SVM model to actually predict the quality scores from these LVs.

Empirical cross-validation results showed that the studied QE approach was able to obtain very large feature reduction ratios, and at the same time, it usually outperformed systems built with all the original features and systems that use PCA instead of PLS to reduce the dimensionality. Unfortunately, results in the held-out test partitions were disappointing. The results of different statistical tests seem to indicate that this was due to the small size of the training partitions. Hence, larger RMSE improvements could be expected in test whenever a representative training partition is provided.

A complimentary advantage of the studied QE approach is its time-efficiency. This fact makes our approach well-suited to be deployed in scenarios with strict temporal restrictions, such as interactive MT sys-
tems. Alternatively, we could take advantage of this efficiency to predict translation quality from huge sets of features. Results in this direction show that our approach was able to efficiently manage more than a thousand features largely improving prediction accuracy.

6. Acknowledgments

Work supported by the European Union 7th Framework Program (FP7/2007-2013) under the CasMaCat project (grants agreement nº 287576), by Spanish MICINN under grant TIN2012-31723, and by the Generalitat Valenciana under grant ALMPR (Prometeo/2009/014).

7. References


Attachment C

Task 3.2


Partial least squares for word confidence estimation in machine translation.

In 6th Iberian Conference on Pattern Recognition and Image Analysis, (IbPRIA) LNCS 7887, pages 500–508. Springer.
Partial Least Squares for Word Confidence Estimation in Machine Translation

Jesús González-Rubio1, J.Ramón Navarro-Cerdán2, and Francisco Casacuberta1

1 Universitat Politècnica de València, Camino de Vera s/n, 46022, Valencia, Spain, {jgonzalez, fcn}@dsic.upv.es
2 Instituto Tecnológico de Informática, Camino de Vera s/n, 46022, Valencia, Spain, jonacer@iti.upv.es

Abstract We present a new technique to estimate the reliability of the words in automatically generated translations. Our approach addresses confidence estimation as a classification problem where a confidence score is to be predicted from a feature vector that represents each translated word. We describe a new set of prediction features designed to capture context information, and propose a model based on partial least squares to perform the classification. Good empirical results are reported in a large-domain news translation task.

Keywords: Machine translation, Word confidence estimation, Statistical multivariate analysis, Partial least squares discriminant analysis

1 Introduction

Despite an intensive research in the last twenty years, the pattern recognition approach to translation, known as statistical machine translation (SMT), is still far from perfect [1]. Thus, a desirable feature to improve its broader and more effective deployment is the capability of predicting the reliability of the generated translations. This task is referred to as confidence estimation (CE).

Following previous works in the literature [2,3], we address CE for translated words as a conventional pattern classification problem in which a feature vector is obtained for each word in order to classify it as either correct or incorrect. This point of view provides a solid, well-know framework, within which accurate two-class classifiers can be derived. The challenges of this approach are to find an appropriate set of features, and to learn accurate classification models.

Ueffing et al. [2] were the first to apply word posterior probabilities, very effective in speech recognition, to estimate MT confidences. They compute such probabilities from N-best lists of translations, and use them as direct estimations of the reliability of the translated words. Sanchís et al. [3] proposed new approaches to compute features from N-best lists, similarly as posterior probabilities are computed in [2]. Moreover, they proposed a smoothed naïve Bayes classifier to combine these features and improve prediction accuracy. Since naïve
Bayes models work on discrete domains, continuous features such as these must be mapped to a discrete domain which involves an additional tuning step.

Our work extends previous approaches in several aspects, including the addition of new features, and the use of a novel classification model based on multidimensional statistical analysis. As [3], we also compute prediction features based on posterior probabilities. However, we generalize this approach to take into account the context of each word. The key idea is that the reliability of a word is influenced by the context in which it appears, therefore by using context-aware features we expect to obtain a stronger estimation of each word reliability. Additionally, we propose a new classifier based on the partial least squares [4]. This classifier performs an intrinsic transformation of the features such that a maximum separation among classes is obtained. Thus it is an effective and efficient method that allows us to build robust classifiers even for ambiguous and redundant features such the ones in natural language processing.

The rest of the article is organized as follows. A brief review of SMT is given in Section 2; Section 3 describes the predictor features used in the experimentation; Section 4 describes the classifier which is based on partial least squares discriminant analysis; Section 5 presents the experimental setup, the assessment measures, and the results of the experiments; and, finally, Section 6 provides a summary and presents the final conclusions.

## 2 Statistical Machine Translation

SMT formalizes the translation problem as follows. Given a source language sentence $s \in S$, the goal is to obtain its equivalent target language translation $t \in T$. From the set of all possible target language sentences, we are interested in that $\hat{t}$ with the highest probability\(^3\):

$$\hat{t} = \arg\max_{t \in T} Pr(t|s)$$  \hspace{1cm} (1)

The posterior probability $Pr(t|s)$ is usually modeled by a log-linear, namely a maximum-entropy [5], model. The posterior probability is computed from a set of feature functions $f_m(t, s)$ and a corresponding set of weights $\lambda_m$:

$$Pr(t|s) \approx P_\lambda^*(t|s) = \frac{\exp \left( \sum_m \lambda_m f_m(t, s) \right)}{\sum_{t' \in T} \exp \left( \sum_m \lambda_m f_m(t', s) \right)}$$  \hspace{1cm} (2)

Since the normalization term in the denominator does not depend on the target sentence $t$, it can be omitted during the search process. Thus, the optimum target language sentence $\hat{t}$ can be finally computed as:

$$\hat{t} = \arg\max_{t \in T} \sum_m \lambda_m f_m(t|s)$$  \hspace{1cm} (3)

\(^3\) We use $Pr(\cdot)$ to denote general probability distributions, and $P(\cdot)$ to denote model-based distributions.
3 Prediction Features

Now, we describe the set of prediction features used in the experimentation. As in [3], our features are based on posterior probabilities computed from $N$-best lists [2]. However, we generalize this approach to take into account the context of each target word. As done in language modeling, we consider the preceding words as the context, namely the history, of each target word.

Given a source sentence $s$, let $t = t_1 \ldots t_i \ldots t_{|t|}$ be the translation hypothesized by an SMT system, and let $\mathcal{L} = \{t_1, \ldots, t_n, \ldots, t_N\}$ be the corresponding $N$-best list of translations. $\mathcal{L}$ is ordered by a given score $W(t_i)$ assigned to each translation. For each word $t_i \in t$ with history $\hat{h}_c(t_i) = t_{i-c} \ldots t_{i-1}$ of size $c$, we compute features $F(t_i, c)$ by summing up the scores of those translations in $\mathcal{L}$ that contain word $t_i$ with history $\hat{h}_c(t_i)$ in a position aligned to position $i$ of $t$:

$$F(t_i, c) = \frac{1}{Z} \sum_{t' \in \mathcal{L}} W(t')$$

where $A(t', t_i)$ is an alignment function that returns the position in $t'$ aligned to word $t_i$. Therefore, $t'_A(t', t_i)$ is the actual word in $t'$ aligned to $t_i$, and $h_c(t'_A(t', t_i))$ is its history. We sum the scores of those sentences for which the aligned word $t'_A(t', t_i)$ is equal to $t_i$ and the history of the aligned word $h_c(t'_A(t', t_i))$ is equal to the history of $t_i$. Finally, $Z = \sum_{t' \in \mathcal{L}} W(t')$ is a normalization term introduced to obtain probability-like features.

The actual value of the feature $F(t_i, c)$ in Equation (4) depends on the definition of the alignment $A(t', t_i)$ and score $W(t')$ functions. Following [3], we consider three different alignment methods:

- **Lev**: Levenstein alignment [6]
- **Target**: Word $t_i$ is aligned to position $i$ in $t'$
- **Any**: Word $t_i$ is aligned to any position $i'$ in $t'$ so that $t'_{i'} = t_i$

and three different scoring schemes:

- **Prob**: The score of $t'$ is the probability $P(t'|s)$ given by the SMT model
- **Rank**: The score of $t'$ is one divided by the position of $t'$ in $\mathcal{L}$
- **Freq**: All translations are assigned equal score: $\forall t' \in \mathcal{L}, W(t') = 1.0$

Thus, given a target word $t_i$, we compute nine different features for each history size $c$. We name each feature with the size of the context, the scoring scheme, and the alignment method. E.g., 0-ProvLev stands for history size equal to zero, probability scoring, and Levenstein alignment. Note that the nine features for a history size $c = 0$ are equal to those described in [3].

We compute two additional features based on the simple SMT model 1 [7] by IBM. Again, let $t$ be the translation of source sentence $s = s_1 \ldots s_j \ldots s_{|s|}$. The features are given by the average lexicon probability of word $t_i$ over all source words (Ibm1Avg), and its maximal lexicon probability (Ibm1Max):

$$\text{Ibm1Avg}(t_i) = \frac{1}{|s| + 1} \sum_{j=0}^{|s|} P(t_i|s_j) \quad \text{Ibm1Max}(t_i) = \max_{0 \leq j \leq |s|} P(t_i|s_j)$$

(5)
where \( P(t_i|s_j) \) is the model 1 probability, and \( s_0 \) is the empty source word.

We consider history sizes from zero to three. Therefore, we compute a total of 36 \( N \)-best features and two additional model-1-based features, for a total of 38 features. Additionally, note that the set of 11 features described in [3] is a subset of our 38 features: the nine \( N \)-best features with history size \( c = 0 \) plus the two model-1-based features.

4 Partial Least Squares Discriminant Analysis

We formalize CE as a classification problem \( \mathbb{R}^m \to \{0, 1\} \) where we predict a discrete variable \( y \in \{0, 1\} \) (0 denotes an incorrect word and 1 a correct one) given a vector of \( m \) explanatory variables \( x^T = (x_1, \ldots, x_m) \) (the set of features that represents each word). The features described in the previous section are obviously highly redundant, thus the selected learning method has to be robust in the presence of noisy data. We chose to use partial least squares discriminant analysis [4] (PLS-DA). PLS-DA performs an implicit transformation of a set of possibly noisy features into a set of uncorrelated latent variables that still account for the maximum co-variability between the features and the discrete variable \( y \). Formally, given a training set \( \{x_i, y_i\}_{i=0}^n \), let \( X \) be a matrix where each row is the feature vector \( x_i \) of a training sample, and \( y \) a vector with the class \( y_i \) of each sample, PLS-DA builds the following linear model:

\[
y = Xb + f
\]  

The estimation of the regression coefficients \( b \) for PLS-DA is different from the conventional least squares regression. The intuitive idea of PLS-DA is to describe \( y \) as well as possible, hence to make the vector or Gaussian errors \( \|f\| \) as small as possible, and, at the same time, take advantage of the relation between \( X \) and \( y \). To do that, PLS-DA defines two independent transformations \( P \) and \( q \) (for \( X \) and \( y \) respectively) with \( E \) and \( f \) being the corresponding residual errors, and a linear relation \( R \) linking both blocks:

\[
X = TP^T + E \quad y = Uq^T + f
\]

\[
U = TR
\]

where matrices \( T \) and \( U \) are the projections from \( X \) and \( y \) respectively. Specifically, each of the columns of the \( T \) matrix represents one of the latent variables of \( X \). The NIPALS algorithm [4] is used to solve this optimization problem. In this case, \( b \) is estimated as:

\[
b = Rq^T \quad \text{where} \quad R = W(P^TW)^{-1}
\]

where \( W \) is an internal weight matrix used by the algorithm that accounts for the correlation between \( X \) and \( U \). The predictions of the PLS-DA model are in the range \([0, 1]\), thus, each word is finally classified as either correct or incorrect depending on whether its score exceeds or not a given classification threshold \( \tau \).

In the experiments, we use the PLS-DA classifier implemented by the \texttt{pls} library of the R toolkit [8]. The dimension of the intrinsic reduction performed by PLS-DA remains as the only free parameter of the classifier.
5 Experiments

5.1 Experimental Setup
We computed word confidence scores for translations of the English-Spanish news evaluation data used in the quality estimation task of the 2012 workshop on statistical machine translation [9]. Those translations were generated by a log-linear SMT model trained on the Europarl and News Commentaries corpora as provided in the same workshop for the shared translation task. We used this same training data to build the Model 1 model required for $ibm1Avg$ and $ibm1Max$ features. Evaluation data contains 1832 translations for training and 422 translations for test. Additionally, for each translation the corresponding source sentence, and list of 1000-best translation options are also available. We use these to compute features for every word in the lowecased and tokenized version of the corpora. This process results in a training corpus with $\sim 55$ thousand samples and a test corpus with $\sim 11$ thousand samples, each sample being a 38-dimensional float vector of features.

The optimum classification threshold $\tau$, and the optimum values for the free parameters of the models were estimated by ten-fold cross validation on the training corpus. As a baseline, we compare the performance of the PLS-DA classifier to the smoothed naïve Bayes model proposed in [3].

5.2 Confidence Tagging
To evaluate the performance of our confidence estimations, each word has to be tagged as either correct or incorrect. Since manual tagging of the words by human experts is a slow and expensive task, we automatically tag translated words by comparing each translation to the reference translation in the corpus. We follow [3] and consider three different tagging methods:  

Word error rate (WER): Each translated word is tagged as correct if it is Levensthein-aligned [6] to itself in the reference.

Position-independent error rate (PER): The word is searched in the reference, and, if found, it is drawn without replacement and tagged as correct.

Position-independent error rate with replacement (PERR): The word is searched in the reference, and, if found, it is drawn with replacement and tagged as correct.

5.3 Assessment Measures
Let us assume a translation task that contains $N_c$ words tagged as correct and $N_i$ words tagged as incorrect. Then after confidence classification is performed for a certain threshold value, let us assume that $FP(\tau)$ of the words tagged as incorrect are classified as correct $0 \leq FP(\tau) \leq N_i$ (false positives), and $0 \leq TP(\tau) \leq N_c$ words tagged as correct that are classified as correct (true positives). Then, we

\[\text{Note that WER is the most strict tagging while PERR is the most tolerant.}\]
define the false positive rate $\text{FPR}(\tau)$, the true positive rate $\text{TPR}(\tau)$ and the classification accuracy, namely the confidence error rate $\text{CER}(\tau)$:

$$\text{FPR}(\tau) = \frac{\text{FP}(\tau)}{N_i} \quad \text{TPR}(\tau) = \frac{\text{TP}(\tau)}{N_c} \quad \text{CER}(\tau) = \frac{\text{FP}(\tau) + (N_c - \text{TP}(\tau))}{N_c + N_i}$$

The trade-off between $\text{FPR}(\tau)$ and $\text{TPR}(\tau)$ can be represented in a so called receiver operating characteristic (ROC) curve. ROC curves are calculated by using different threshold values $\tau \in [0, 1]$ to classify the words and keeping track of $\text{FPR}(\tau)$ and $\text{TPR}(\tau)$. The area under the ROC curve (AROC) provides an adequate overall estimation of the classification accuracy.

We also compute the statistical significance of the observed performance differences. Specifically, we compute $p$-values using a randomization version of the paired $t$-test [10]. First, we use an evaluation measure, e.g. AROC, to determine the performance difference between the outcomes of two methods. Then, we repeatedly create shuffled versions of the original outcomes, determine the performance difference between them, and count the number of times that this difference is equal or larger than the original difference. Finally, the $p$-value is the proportion of iterations in which the difference was indeed larger for the shuffled version.

5.4 Results

In a first experiment, we studied the performance of each individual feature described in Section 3 as a direct estimation of the quality of the words. Figure 1 displays the cross-validation AROC obtained by each feature for the three tagging criteria. Results show that the model-1-based features obtained the best performance. Regarding posterior probability features, Lev and Any alignment methods consistently outperformed Target alignment for all history sizes. Also, features obtained slightly worse results as larger history sizes were used.

Then, we evaluated the classification accuracy of the PLS-DA model presented in Section 4 in comparison to the naïve Bayes classifier proposed in [3], and the best-performing feature Ibm1Avg. We present results for two sets of features, the 11 features used in [3], and the extended set of 38 features proposed here. Table 1 displays the AROC and CER scores obtained in the test set.
Table 1. AROC and CER results in the test set. Best results are displayed in bold. We use asterisks (*) to denote the statistical significance of the AROC and CER differences observed between PLS-DA and naïve Bayes: *p < 0.01, **p < 0.001.

<table>
<thead>
<tr>
<th>Feature set</th>
<th>Method</th>
<th>WER</th>
<th>PER</th>
<th>PERR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AROC</td>
<td>CER</td>
<td>AROC</td>
</tr>
<tr>
<td>Ibm1Avg (best individual feature)</td>
<td>naïve Bayes</td>
<td>0.70</td>
<td>0.29</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td>PLS-DA</td>
<td>0.70</td>
<td>0.29</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.75</td>
<td>0.28</td>
<td>0.75</td>
</tr>
</tbody>
</table>

(a) Extended 38 features

(b) 11 features as in [3]

Figure 2. ROC curves for results in the test set using WER tagging.

Results show that the proposed PLS-DA model consistently outperformed the baseline naïve Bayes model for all tagging criteria and feature sets. Additionally, results for naïve Bayes clearly deteriorated, e.g. −0.05 AROC for PER tagging, when the extended feature set was used. In fact, naïve Bayes obtained worse AROC results with the extended feature set than the single best-performing feature. This fact indicates that the naïve Bayes model had difficulties in handling highly-correlated features of different quality, and these learning issues get worse as more features are used. I.e., to obtain a good performance, it would require a previous feature selection step to filter out redundancy and low-quality features. In contrast, PLS-DA not only obtained better performance for the “high-quality” 11-feature set in [3], but its performance did not degraded for the highly-redundant extended set of features. These results indicate that PLS-DA is a better-performing and more robust classification model than the baseline naïve Bayes model. Finally, Figure 2 shows for the WER tagging method the test set ROC curves for PLS-DA, naïve Bayes, and the best individual feature.

6 Summary and Future Work

We have presented a new CE method to classify as correct or incorrect the words of the translations generated by an SMT system. For each word, we compute a number of features, and use a partial least squares model to classify the word.
Our results showed that features based on the model 1 lexicon achieve the better performance, followed by those computed from N-best with the Lev and Any alignment criteria. Also, the use of context information did not improve the results of the previously used [2,3] N-best-based features. Regarding the classification models, the proposed PLS-DA model consistently outperformed the smoothed naïve Bayes model proposed in [3] in all test conditions. In fact, PLS-DA had shown to be an effective, scalable, and robust classification model quite adequate for the task.

As future work, we plan to exploit the scalability of the PLS-DA model to study interactions between the features. Additionally, we will explore techniques to automatically estimate the optimum number of latent variables to use.

Acknowledgements

We thank Alberto Sanchis for providing us with the code of the naïve Bayes model. Work supported by the European Union Seventh Framework Program (FP7/2007-2013) under the CasMaCat project (grans agreement nº 287576), by Spanish MICINN under TIASA (TIN2009-14205-C04-02) project, and by the Generalitat Valenciana under grant ALMPR (Prometeo/2009/014).

References