```
2PhasePrediction ⌄ [
minItems: 2
maxItems: 2

The 2-phase prediction results, the first entry represents the binary prediction
TAG/NOTAG, while the second entry represents the multi-class prediction.

2PhasePrediction ⌄ {
   description:
                       The prediction results.

   scores               > {...}

 }]
```

Université de Montréal

État d'avancement du contrat entre le CNRC et le RALI

Contact :
Philippe Langlais +1 514 343 61 11 ext: 47494
RALI/DIRO felipe@iro.umontreal.ca
Université de Montréal http://www.iro.umontreal.ca/~felipe/

Résumé

— Les travaux menés sur le nettoyage sont terminés (un rapport reste à produire).

— Les travaux sur la structuration du MegaCorpus à l'aide de métadonnées ont été volontairement mis de côté en partie faute de métadonnées facilement exploitables, et dans le contexte où le Bureau souhaite changer de solution logicielle pour gérer sa mémoire de traduction.

— Les travaux sur la traduction neuronale ont en revanche été plus intenses et le Rali a développé des systèmes neuronaux sur une partie des données du MegaCorpus (5.8M de paires de phrases jugées correctes par le nettoyeur développé), ainsi qu'un système capable de reconnaître une traduction produite automatiquement.

— En accord avec le CNRC, aucun outil n'a été livré. Certains des outils utilisés ou développés au Rali tournent en effet sur des grappes de GPUs sous environnement Linux au CNRC et ne pourraient pas être facilement déployés dans un autre environnement.
Le Rali reste ouvert à une discussion quant à l'exportation possible de certains outils développés dans le cadre de ce contrat. Une démonstration a d'ailleurs été développée qui intègre un aligneur de phrases à l'état de l'art jumelé à des détecteurs de mauvaises paires de phrases (nettoyeur).
Il existe enfin des ressources développées au Rali qui peuvent s'avérer utiles au Bureau.

Livrables

Les livrables figurant dans l'entente entre le Rali et le CNRC sont au nombre de sept (plus un rapport d'étape). Quatre sont des rapports :

① impact des erreurs d'alignement phrastique
☛ *en cours (avant la rentrée universitaire).*

③ évaluation des détecteurs embarqués dans le nettoyeur
☛ *réalisé, voir l'article de 9 pages "Cleaning a(n almost) Clean Institutional Translation Memory" soumis à COLING 2020.*

⑤ étude comparative du *matcheur* du Bureau et celui livré en ④
☛ *pas abordé pour le moment*

⑦ scenarios d'intégration de la traduction dans les outils du Bureau
☛ *Une intégration suggérée : identifier des traductions produites automatiquement. Voir à ce sujet l'article "Human or Machine" soumis à COLING 2020.*

et trois sont des programmes :

② `cleaner`, le nettoyeur du MC
☛ *une solution implémentée et recommandée*

④ `matcheur`, un programme qui retourne des paires de phrases du MC en exploitant les méta-données du MC et le texte à traduire de manière holistique
☛ *rien de développé*

⑥ système de traduction neuronale entraîné sur le MC nettoyé
☛ *systèmes neuronaux développés sur un sous-ensemble de 5.8M de paires de phrases du* MegaCorpus *jugées bonnes par notre nettoyeur.*

En accord avec le CNRC, aucun outil n'a été livré. À noter cependant qu'une démonstration a été développée au Rali qui prend en entrée une paire de documents, qui les segmente en phrases, aligne les phrases et classifie les alignements résultants de bon ou mauvais. La sortie prend la forme d'une page Web consultable depuis un simple navigateur et qui permet de visualiser le bruit présent dans une telle chaine de traitement. Nous avons convenu avec le CNRC qu'une telle interface pourrait être pertinente au Bureau. La démonstration est décrite dans un article de 5 pages intitulé "An Open Source Translation Memory Populator" soumis à la conférence COLING 2020 (session démonstrations).

Nous détaillons dans la suite le travail réalisé et les travaux en cours. Il est à souligner que dans le contexte actuel de la Covid, les travaux menés au Rali sont ralentis.

Nettoyage

Détection de paires de phrases erronées

Nous avons développé et comparé plusieurs types d'approches permettant de nettoyer le MegaCorpus :

— des méthodes heuristiques recherchant des anomalies particulières comme la présence de faux amis, de calques, d'erreurs dans les données numériques, des problèmes d'encodage (qui ne sont pas négligeables). Un total de 13 heuristiques ont été implémentées et appliquées au MegaCorpus.

— différents classifieurs supervisés informés de traits calculés spécifiquement comme par exemple le nombre de mots de la phrase source et de la phrase cible, le nombre de correspondances selon un lexique bilingues, etc.

— des méthodes d'apprentissage profond dont les traits sont appris de manière supervisée, comme l'approche décrite dans (Grégoire et Langlais 2018).

— une méthode d'apprentissage profond non supervisée (Laser) qui met à l'usage des représentations de mots multilingues précalculées pour 93 langues (Schwenk et al. 2017).

— le système non supervisé Tmop (Jalili Sabet et al. 2016) qui utilise des heuristiques, des traits calculés à partir de modèles statistiques de type IBM et des représentations de mots pré-entraînées.

La comparaison de ces méthodes s'est faite sur un corpus de 2021 paires de phrases annotées manuellement (erreur ou pas et type d'erreur), ainsi qu'en entraînant des systèmes de traduction neuronaux à l'état de l'art sur un extrait du MEGACORPUS avant et après nettoyage par l'une de ces méthodes et en prenant les gains des systèmes entraînés sur le corpus filtré comme une mesure de la qualité de la technique de filtrage. Les deux évaluations ont montré d'une part l'inadéquation du système existant TMOP, et les bonnes performances globales de l'approche LASER. Ce dernier système était capable d'identifier 84% des paires des 2021 paires de phrases du corpus de référence.

Une évaluation à postériori menée sur 100 paires de phrases détectées bonnes (par notre nettoyeur) a révélé que 16 paires de phrases étaient en fait mauvaises. Une autre conduite sur 100 paires de phrases détectées mauvaises a montré 33 paires jugées bonnes. Bien que menée sur de petits échantillons, cette évaluation montre que le nettoyage est un problème complexe.

Détection de traductions produites automatiquement

En constatant la qualité des traductions produites par les systèmes de traduction neuronale que nous avons entrainés, nous avons étudié dans quelle mesure nous pouvions reconnaître des traductions produites automatiquement de celles produites par des traducteurs professionnels. Cela pourrait par exemple servir à diagnostiquer des pratiques auprès des traducteurs contractés par le Bureau, comme par exemple des traducteurs qui s'appuieraient trop fortement sur la traduction automatique pour produire leur traduction.

Nous avons à cet effet considéré un sous-ensemble d'environ 500K paires de phrases du MEGACORPUS jugées sans problème par le nettoyeur développé, puis nous avons traduit les phrases anglaises vers le français à l'aide des systèmes neuronaux développés.

Ce corpus a ensuite été utilisé pour entrainer un système à reconnaître les traductions automatiques des traductions professionnelles. Nous avons comparé 18 modèles (neuronaux ou pas) dans différentes conditions, en faisant notamment varier le domaine des textes, et les systèmes (GOOGLE, DEEPL ainsi que nos systèmes entraînés sur le MegaCorpus.[1])

Nous avons montré que bien que la tâche soit compliquée, il est possible de reconnaître avec une fiabilité de l'ordre de 80% des traductions produites par un système neuronal état de l'art (existant ou entraîné).

À faire ________

Un rapport décrivant le type de bruit présent dans le MegaCorpus et sa proportion. La chose est complexe dans la mesure où nous ne disposons pas d'annotations en quantité suffisante. Dans ce rapport (en cours) une estimée de la portion de bruit liée à la chaîne de traitement du système de TM du Bureau (segmentation en phrases, alignement de phrase) sera effectuée.

Un tel rapport est complexe à rédiger dans la mesure où nous ne disposons pas d'annotations en quantité suffisante et où les annotations sont difficiles à apposer.

Il est à noter que le Rali n'a pas déployé ses algorithmes de nettoyage sur le MegaCorpus au complet. Une telle entreprise requiert de nombreux calculs et il conviendra le cas échéant de vérifier dans quel format le Bureau souhaite récupérer le résultat d'un tel nettoyage.

Traduction Neuronale

Nous avons entraîné des systèmes de traduction neuronale état de l'art sur 5.8M de paires de phrases jugées bonnes par notre nettoyeur. Bien que n'utilisant qu'une petite portion du MegaCorpus, ces systèmes requièrent plusieurs dizaines d'heures d'entraînement sur 4 cartes graphiques (GPUs) rapides (Tesla V100-SXM2) utilisées en parallèle. Traduire 550k phrases sur la même architecture informatique à l'aide de ces systèmes requiert de 10 à 26

1. Aucun élément du MegaCorpus n'a dans cette étude été soumis à des systèmes susceptibles de garder trace des phrases à traduire. Nous n'avons donc soumis aucune phrase du MegaCorpus au système de Google, et avons utilisé le système DeepL interne au Bureau auquel nous avons eu accès via le CNRC.

heures de calcul selon le système. Ces quelques faits illustrent que l'utilisation de la traduction neuronale n'est pas une activité que l'on peut mener sans une infrastructure matérielle adéquate.

À faire

Les expériences réalisées ont été faites sans regard au domaine des données du MEGACORPUS. Nous souhaitons étudier les bonnes pratiques quant au fait qu'au BUREAU, de l'ordre de 200 domaines (clients principaux) sont gérés.

À noter

Le RALI développé une ressource contenant des enregistrements comme celui-ci où l'on peut voir une phrase source (src), la traduction en provenance du MEGACORPUS (ref), et les traductions produites par différents systèmes neuronaux (trans) :

Src	The Parties will mutually ensure that any information of a confidential nature will be treated as such.
Trans	Les parties veilleront mutuellement à ce que toute information de nature confidentielle soit traitée comme telle.
Ref	Les parties veilleront mutuellement à ce que tous les renseignements de nature confidentielle soient traités comme tels.

Une telle ressource pourrait être utile au BUREAU.

Recalibrage d'objectifs

Nous avons mis de côté les activités sur la structuration du MegaCorpus visant à en optimiser l'usage. Il était en effet prévu de tester différentes politiques de recherche dans la mémoire favorisant la cohésion des appariements retournés (comme trouver des appariements extraits d'un même document ou d'un même domaine, ou encore produit par un même traducteur).

Tel qu'indiqué dans le contrat, cet objectif prend tout son intérêt si nous disposons au Rali d'une description de la façon de fonctionner du *matcheur* au Bureau (ou à tout le moins de listes d'appariements retournées par le système de mémoire au Bureau) et si nous disposons de méta-données permettant de structurer le MegaCorpus (comme la personne/groupe ayant traduit, la date de traduction, etc). Nous disposons d'un accès au système de mémoire du Bureau, aussi pouvons-nous probablement produire des appariements retournés par le système du Bureau pour des phrases d'un document d'intérêt.

Reste à voir dans un contexte où le Bureau souhaite migrer de système de gestion de sa mémoire de traduction si une telle étude revêt encore un intérêt.

Grégoire, Francis et Philippe Langlais (2018). “Extracting Parallel Sentences with Bidirectional Recurrent Neural Netwrok to Improve Machine Translation”. In : *COLING*. Santa-Fe.

Jalili Sabet, Masoud et al. (2016). “TMop : a Tool for Unsupervised Translation Memory Cleaning”. In : *Proceedings of ACL-2016 System Demonstrations*. Berlin, Germany : Association for Computational Linguistics, pages 49–54. doi : 10.18653/v1/P16-4009. url : https://www.aclweb.org/anthology/P16-4009.

Schwenk, Holger et al. (2017). “Learning Joint Multilingual Sentence Representations with Neural Machine Translation”. In : *CoRR* abs/1704.04154. arXiv : 1704.04154. url : http://arxiv.org/abs/1704.04154.

Cleaning a(n almost) Clean Institutional Translation Memory

Anonymous COLING submission

Abstract

While recent studies have been dedicated to cleaning very noisy parallel corpora for the sake of better Machine Translation, we focus in this work on filtering a mostly clean institutional Translation Memory for the sake of a better internal usage of the memory. This problem of practical interest has not received much consideration from the community. We were provided access to Translation Bureau of Canada proprietary data, which is extensive and multi-domain. We propose two ways of evaluating this task (manual annotation and Machine Translation), and compare five approaches involving deep-, feature-, and heuristic-based solutions. We report significant gains over a state-of-the-art, of-the-shelf cleaning system available.

1 Introduction

The Translation Bureau of Canada handles a very large quantity of translation requests in a large number of domains. In doing so, they call upon a large pool of translators (internal and freelance) whose translations are fed into a huge internal Translation Memory we call the MEGACORPUS. This memory is accessed by translators through a dedicated interface. Given the purpose of the translation memory, it should be as exempt of noise as possible. Unfortunately, because of the high number of translators and their varied levels of expertise, as well as tight deadlines and technological issues, noise inevitably accumulates over the years.

There are two main types of problems that arise in such a translation memory. Bad alignments occur because of the pipeline used to populate the MEGACORPUS, whereby source and target documents are segmented into sentences before being aligned, two processes which may produce errors. Low quality sentence pairs arise for a variety of other reasons, including spelling or morphosyntax that does not meet established norms, missing translation units on the target side, as well as typical translation errors (calques, bad wordings, etc.). Both types of errors reduce the usefulness of a translation memory system, hence the need for cleaning.

In the remainder of this paper, we first discuss related work in Section 2, then we present the MEGACORPUS in Section 3, and present the corpus filtering approaches we implemented in Section 4. Evaluating the quality of a filtering solution is not easy, and we resort to Machine Translation in this work. We therefore present the neural machine translation engines we evaluated in Section 5. We report our results and analysis in Section 6, and conclude in Section 7.

2 Related works

A widely successful method to identify translated sentence pairs (SPs) in a comparable corpus was proposed by Munteanu and Marcu (2005). It relies on a feature-based classifier trained in a supervised way. Different features are exploited including length ratio of the source and the target sentences, bilingual lexicon matches, and a set of features based on IBM word translation models (Brown et al., 1993). The authors showed that the parallel material mined from news extracted over the web improved a downstream statistical translation engine.

The success of deep learning methods has recently led to a number of classifiers trained without feature engineering. Notably, Grégoire and Langlais (2018) describe a siamese recurrent neural network that encodes source and target sentences into vectors that are then fed through a non-linear transformation in order to classify a sentence pair as parallel or not. The authors showed that training such a model yielded better performance than the aforementioned approach, and that adding parallel material extracted from Wikipedia using this model leads to systematic (although modest) gains in both statistical and neural machine translation engines.

While these studies show fairly convincingly that parallel sentences can be mined from comparable corpora, it remains unclear whether the methods used are also useful for filtering out bad sentence pairs from a translation memory.

One early attempt to tackle this issue was presented by Macklovitch (1994), who proposed simple heuristics to detect specific problems they observed in real (professional) translations, such as errors in numerical entities, the presence of calques, and abnormal translation sizes. The work of Barbu (2015) can be seen as an extension of this line of work. He proposed 17 features, some based on formal clues (e.g., the presence/absence of XML tags, emails, URLs, numbers, capital letters or punctuation) and others using external resources (i.e., the Bing translation API and the language detector Cybozu). Using these features, he trained models to recognize bad translations, using a very small training set (of 1243 sentence pairs). The best model (an SVM model) achieved an f-score of 81% on a test set of 309 sentence pairs. The author concluded that applying it on MyMemory (Trombetti, 2009) would filter out too many good sentence pairs.

Jalili Sabet et al. (2016) introduced a fully unsupervised Translation Memory cleaning tool called TMOP, which uses 25 different features. Some were adapted from features proposed by Barbu (2015), and others are based on the work of de Souza et al. (2014) and focus on estimating the quality of the translation. They also make use of multilingual word embeddings, using the method proposed by Søgaard et al. (2015). Each feature acts as a filter for which a score is returned, then TMOP transforms these scores into a final decision. They tested their system on a subset of the English-Italian MyMemory and it produced results comparable to (Barbu, 2015) while being unsupervised.

Recently, there has been increased interest in filtering very noisy (web-mined) parallel corpora using unsupervised deep learning. Chaudhary et al. (2019) trained multilingual sentence embeddings using LASER and exploited an ensemble of evaluation methods like Zipporah (Xu and Koehn, 2017), Bicleaner (Sánchez-Cartagena et al., 2018), and dual conditional cross-entropy filtering (Junczys-Dowmunt, 2018) to score each sentence pair. We tried this approach in this work, but found LASER alone more efficient.[1] (Wang et al., 2018) is another state-of-the-art online data selection method for de-noising training material and adapting to a specific domain. In our case, we are dealing with over 200 domains, which makes this approach less suited.

3 Datasets

To conduct our experiments, we created a number of subsets of the MEGACORPUS, which we describe below. Their characteristics are summarized in Table 1.

The MEGACORPUS comprises over 1.8M TMX files across more than 200 broad domains (e.g. health_canada, fisheries_oceans), for a total of over 139 million (139 454 913) sentence pairs. In the translation memory system used by the Translation Bureau, translators may flag a problem with a sentence pair, in which case the full TMX file is flagged as problematic and banned from the system. This flagged material represents 7.7% of all sentence pairs. Note that we do not know specifically which sentence pairs were found problematic in the flagged TMX files.

For training our translation engines (see Section 5) we sampled 14M sentence pairs from the MEGACORPUS at random so that we get comparable amounts of SPs in each domain. 4.3M sentence pairs were sampled from the flagged part of the corpus, so that we could monitor if this material impacts translation. We call this corpus MT-TRAIN. We also sampled a test set from the MEGACORPUS, but excluding the flagged corpus and the bad SPs identified by heuristics (see Section 4.1). This was meant to ensure that the test material is not corrupted. We ended up with a subset of 10 000 sentence pairs

[1]On the evaluation reported in Section 6.1 Table 2, we obtained an accuracy of 0.54 with the ensemble approach, while LASER alone got 0.84.

Corpus	#SPs	Types #fr	Types #en
MEGACORPUS	139.5M	1.1M	1.4M
META-H	18.9M	570 896.	680 603
BALANCED	7M	462 703	444 305
MT-TRAIN	14M	387 594	465 603
MT-TEST	10k	10 733	12 884
2021	2021	3 480	4 304

Table 1: Characteristics of the corpora used. We count as types space-separated strings that contain only alphabetical symbols and are no longer than 15 characters. The presence of many token types is due among other things to a large proportion of proper names, as well as issues with the many file formats used to store source and translated documents.

which we call MT-TEST.

We applied the two meta-heuristics described in Section 4.1 on this huge dataset and ended up with 17.7M sentence pairs identified as good, and 1.2M pairs identified as bad. Those two subsets constitute an annotated corpus we call META-H. Because of the nature of the heuristics we deployed, we observed that many of the bad SPs feature obvious errors (presence of gibberish, typos, non-translations, etc.), while there are fewer misaligned SPs and SPs involving subtle translation errors.

We therefore enhanced META-H automatically by: a) taking existing English sentences and pairing them at random with existing French sentences, for a total of 1.15M artificially created misaligned SPs, and b) replacing each (source or target) token containing 4 or more characters with one of its top five nearest neighbours in a space of `fastText` word embeddings (Bojanowski et al., 2016). We produced 1.15M sentence pairs with (often not so) subtle problems, as can be observed in Figure 1, where nearest neighbours of a word are often typos. After this addition, we obtained 3.5 million bad SPs. To create a balanced set of 7 million SPs, we combined these with 3.5M pairs selected randomly among those identified as good. This balanced corpus named BALANCED is used to train our supervised classifiers.

`ori`	If you feel like sleeping , stand up and move to back .	`ori`	The government of Canada will match your contribution dollar for dollar .
`cor`	If you feels just napping, Stand up and moves to abck .	`cor`	The governnment of Quebec will match your Contribution dolllar for dollar.

Figure 1: Examples of original (`ori`) and corrupted (`cor`) sentences.

Finally, in the course of our experiments, we conducted targeted manual evaluations that we compiled into a corpus named 2021, which we use for evaluation purposes. This corpus includes the 1721 SPs used to adjust the meta-heuristics and the 300 SPs used to evaluate them, as mentioned in Section 4.1. We found 1182 (58.5%) good SPs and 839 (41.5%) bad SPs, making 2021 a rather balanced corpus.

4 Approaches

We compared five different approaches to identify noisy sentence pairs, three of them being unsupervised (pre-trained or not trained at all).

4.1 Heuristics

By inspecting the MEGACORPUS, we noticed a number of problems, some of which (we thought) could be detected by specific rules, as illustrated by the examples shown in Figure 2. We developed 13 heuristics that we outline below. They all take as input a sentence pair (SP) and produce a score between 0 (noisy SP) and 1 (good SP). Most of those heuristics are exploited in one way or another in the systems described in Section 2.

①	en	Section 34 verification and certification
	fr	Fiches de spécimen de signature.
②	en	Native Women's Association of Canada.
	fr	James Anaya, Doc. NU A/HRC/9/9, 11 août 2008. Native Women's Association of Canada.
③	en	Since 2005, we have received some \$1.4 billion to purchase 17 vessels.
	fr	Conflicting sovereignty claims to the Arctic are resulting in a race to the North.
④	en	`OP #L O- Sk`
	fr	`i@n [u05ce \x9b}`

Figure 2: Examples of problems in MEGACORPUS. The first involves section titles, which are often poorly aligned. The second shows a partially correct alignment, likely involving a sentence segmentation problem. The third example is a pure misalignment. The forth involves encoding issues, which are rather frequent since many different file formats are handled by the alignment pipeline at the Translation Bureau.

Several heuristics look for matches (i.e. either exact string matches or bilingual matches based on a lexicon) between the two sentences, each looking for specific units such as numerical entities (NUM), cognates (COG), stop words[2] (STOP), URLs (URL), false friends[3] (FRIEND), punctuation and specific symbols (PUNC). Another heuristic (LEX) counts the number of word translation pairs found according to a bilingual lexicon containing 60k entries. Another heuristic (ION) takes into account the fact that in languages whose vocabulary contains many latinate words, words ending with the suffix `-ion` are often translated by words with the same suffix (e.g., `félicitations` / `congratulations`).

We further developed heuristics that detect specific problems such as the presence of gibberish (GIBB), which we found abundant, or the presence of a source sentence in place of a target one (MONO), most often because parts of a text were not translated, but not filtered out by the pipeline that feeds the MEGACORPUS. We also noticed a lot of problems involving tables of contents (TOC), which often confuses the sentence segmenter, leading to alignments errors. We also check the length ratio (counted in words) of source and target sentences (LEN). Finally, we implemented a rudimentary proxy to spell checking (SPELL), which counts the number of tokens that are correctly spelled (according to a list of words seen at least 1000 times in Wikipedia).

In order to gauge the performance of each heuristic, we created a test set of 1721 sentences. We randomly sampled 1321 SPs from the MEGACORPUS, to which we added 400 sentence pairs that had specific problems (e.g. unbalanced number of cognates). We annotated the resulting 1721 sentence pairs as good or problematic. We used this corpus to adjust the thresholds of our heuristics and select the optimal combination of heuristics in order to distinguish good and bad sentence pairs.

For detecting good SPs, the solution that worked best is a weighted combination of 4 heuristics: NUM, LEX, ION, and PUNC. In contrast, the combination that worked best to predict problematic SPs is a mix of 9 heuristics: LEN, MONO, GIBB, NUM, SPELL, URL, LEX, TOC and PUNC. To further evaluate these two "meta-heuristics" (i.e. weighted combinations of heuristics), we manually inspected a random sample of 150 SPs selected by each (i.e. identified as good or bad respectively). One annotator found 115 good SPs in the former sample (76.7%), and 121 bad SPs in the latter (80.7%).

The fact that we adjusted the meta-heuristics thanks to an annotated corpus makes this approach weakly supervised. In practice however, one could deploy our heuristic system without further adjustments.

[2]We use a lexicon of 93 entries such as `the` / `la`, `le`, `les`.

[3]We use a lexicon of 175 entries such as `fabric` / `fabrique`.

4.2 Feature-based Classifiers

We trained support vector machines and random forests, two well established algorithms. We tested two sets of features to represent sentence pairs. The first contained the score (between 0 and 1) produced by each of the 13 heuristics described in Section 4.1. The second also contained intermediate values produced while computing the heuristics, for a total of 60 scores. In both cases, we added two additional features: the percentage of heuristics that fire an SP as good (resp. bad). The best feature set was by far the larger one, therefore we only report results obtained with it.

We trained the models with `scikit-learn`[4] on standard desktop CPUs. Training took approximately 10 hours per model.

4.3 TMOP

TMOP (Jalili Sabet et al., 2016) is composed of 25 different binary functions meant to capture bad alignments, bad translation quality or "semantic" distance between the source and target. It offers 3 ready-made configurations which control how those functions are aggregated into a final decision. We used the one which classifies an SP as bad if at least 5 functions signal a problem, and good otherwise.[5]

Similarly to (Munteanu and Marcu, 2005), TMOP relies (among other things) on IBM-like features computed through the MGIZA package [6]. It took 13 days to run MGIZA through the 14M sentence pairs of the MT-TRAIN corpus on a 16-core cluster equipped with 70Gb of memory, and 5 more days to run TMOP on it.[7] Therefore, applying TMOP on the full MEGACORPUS would be rather challenging.

4.4 Deep-Learning Classifier

We reimplemented in Keras (Chollet and others, 2015) the model of Grégoire and Langlais (2018), introducing a few variants we found useful. The model architecture consists of two bidirectional LSTMs (Hochreiter and Schmidhuber, 1997), each with 300 hidden units[8] which encode sentences into two continuous vector representations.

The source and target representations are then fed into a Feed-Forward Neural Network containing two hidden layers (with 150 an 75 units respectively), followed by a sigmoid activation function, which outputs the probability that the SP is good.

We trained our model using the Adadelta optimizer (Zeiler, 2012) with gradient clipping (clipped at 5) to avoid exploding gradient and a batch of size 300 (whereas the original implementation uses the Adam optimizer with a learning rate of 0.0002 and a mini-batch of 128).[9] Models were trained using 4 Tesla V100-SXM2 for 10 epochs, which took approximately 2.5 hours.

4.5 LASER

We also compared the LASER (Artetxe and Schwenk, 2018) toolkit[10]. The idea behind this model is to train a single encoder to handle multiple languages such that semantically similar sentences in different languages are close in the embedding space. We used a pre-trained sentence encoder that handles 92 different languages. Sentences from all these languages were mapped to a common embedding space using a 512-dimensional bi-LSTM encoder.

We use the `multilingual-similarity search` (MSS) method available in the toolkit. For each source-language sentence s_i at index i, we find the closest sentence t_j in the target language from

[4] `https://scikit-learn.org/stable/`

[5] The "twenty percent" configuration was by far the better: the "one reject" configuration is producing far too many false negatives, while the "majority vote" configuration hardly detects bad SPs.

[6] `http://www.cs.cmu.edu/~qing/giza/`

[7] We had to run TMOP on a dedicated cluster of 32 CPUs equipped with 300Gb of memory. Training embeddings took only 6 hours of the 5 days required in total.

[8] In the original paper, the authors use 512-dimensional word embeddings and 512-dimensional recurrent states since they learn the word embeddings from scratch. We found it easier (faster) to adapt pre-trained, 300-dimensional `fastText` word embbedings. Also, they tie the parameters of the two encoders, while we do not.

[9] Adadelta does not require us to set a default learning rate, since it takes the ratio of the running average of the previous time-steps to the current gradient.

[10] `https://github.com/facebookresearch/LASER`.

the joint embedding space. If this target sentence has the same index (i.e., $i = j$), the sentence pair is considered good, otherwise not. There is no training involved in this process, since we consume the model as it is. Running this method on one Tesla V100-SXM2 took approximately 14 hours.

5 Neural Machine Translation Models

Evaluating the quality of a translation memory by the performance of a translation engine trained on that memory is rather intuitive. Still, many factors can render this methodology troublesome. In order to draw conclusions that are independent of a specific system, we experimented with two very different neural translation models: XLM, a deep transformer model, and ConvS2S, a convolutional seq2seq model. Both models were trained in parallel on 4 Tesla V100-SXM2. The average time to train XLM was around 22-30 hours depending on the data set, and for ConvS2S, it was 72-96 hours.

5.1 Cross-lingual Language Model

In (Lample and Conneau, 2019), the authors propose three models: two unsupervised ones that do not use sentence-pairs in translation relation, and a supervised one that does. We focus on the third model, named Translation Language Modeling (TLM) which tackles cross-lingual pre-training in a way similar to the BERT model (Devlin et al., 2018) with notable differences. First, XLM is based on a shared source-target vocabulary of sub-words, computed using byte pair encoding (BPE) (Sennrich et al., 2016). We used the 60k BPE vocabulary which comes with the pre-trained language model. [11] Second, XLM is trained to predict both source and target masked words, leveraging both the surrounding words and the other language context, encouraging the model to align the source and target representations. Third, XLM embeds the language of the tokens as well as their position in the sentence, which leads to build a relationship between the related tokens in the two languages.

XLM is implemented in PyTorch and supports distributed training on multiple GPUs. We have modified the original pre-processing code such that, XLM can accept a parallel corpus for training TLM, which is not yet implemented in their GitHub.[12] The translation is produced by a beam search strategy, making use of a beam width of 6 and a unity length penalty.

5.2 Convolutional Sequence to Sequence

The most predominant method to sequence to sequence (seq2seq) learning is to map an input sequence to an output sequence of variable length via a recurrent neural network, e.g. an LSTM. The work of Gehring et al. (2017) demonstrated that convolutional neural networks (CNN) could also be used for seq2seq. The ConvS2S model uses CNNs with Gated Linear Units (Dauphin et al., 2016) for both the encoder and decoder, and includes a multi-step attention layer.

We used the implementation available in the fairseq toolkit (Ott et al., 2019). Similarly to TLM, we use a source and target vocabulary of 60K BPE types. The translation is generated by a beam-search decoder with log-likelihood scores normalized by sentence length.

6 Experiments

6.1 Manual Evaluation

Table 2 shows the accuracy of the classifiers we trained on the BALANCED corpus, as well as the accuracy of unsupervised systems. We observe that training a classifier on top of the features used by meta-heuristics clearly helps. We also observe that TMOP delivers comparable results to our heuristic-infused classifiers, which indicates that it is a good detector on its own, and that combining heuristic signals to word-based translation and embedding features is a good strategy. Further learning how to aggregate those signals would likely improve the performance.

[11] Training TLM without pre-training was rather unstable. We also noticed better results with a back-translation step, but at a high cost in training time.

[12] `https://github.com/facebookresearch/XLM.git`.

The bi-LSTM model, while avoiding feature engineering, leads to much better results overall, which confirms the observations made by Grégoire and Langlais (2018) on artificial data. What comes as a surprise is that the best results are obtained by LASER, and this, without any training to our data. Of course, 2021 is a rather small test set, and results here should be taken with a grain of salt.

Method	**Accuracy**
meta-heuristics	0.42
TMOP	0.60
RF	0.60
SVM	0.63
bi-LSTM	0.79
LASER	0.84

Table 2: Accuracy of approaches trained on BALANCED and tested on 2021. The first line indicates the score of the meta-heuristics developed to identify good and bad SPs (see Section 4.1).

6.2 Machine Translation Evaluation

Table 3 shows the BLEU scores obtained by the different neural translation engines we trained. Removing the flagged material from the training set (see line 2) does not impact BLEU scores significantly, which corroborates the observations we made that the flagged material was generally of good quality. We observe (line 3) that TMOP has problems identifying bad SPs, and is therefore the worst filtering strategy. Clearly, some adaptation to our dataset would in practice be required if we had to deploy it efficiently on MEGACORPUS.

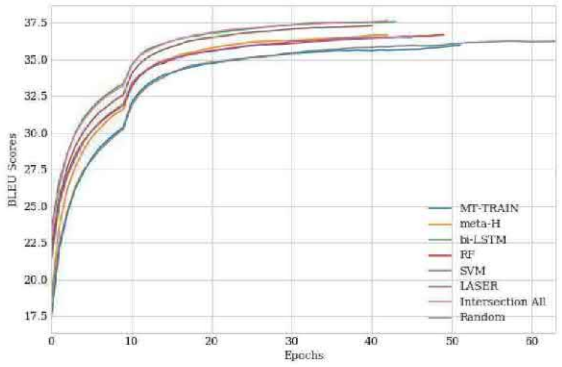
From Table 3, we can observe that cleaning the training material by any of the methods described leads to some gains in BLEU. We were not expecting such a clear outcome, with a corpus as clean as the MEGACORPUS. The largest gains come from the bi-LSTM approach, a supervised approach, but LASER is not far behind. The former approach filters much more, which seems to improve performance further. It should be noted that we applied the supervised filters on the non flagged part (line 2), while the LASER unsupervised method was applied directly to MT-TRAIN.

The differences in BLEU obtained by the two feature-based classifiers do not differ much, with SVM being at a slight edge in the manual evaluation. Lastly, we observe that the gains are consistent over the two translation engines, which is reassuring.

Train set	**#SPs**	XLM	ConvS2S
MT-TRAIN	14	36.25	33.04
¬Flagged	9.67	36.29	33.33
TMOP	13.38	36.49	33.51
RF	7.20	36.70	33.72
SVM	7.50	36.53	33.91
meta-H	8.15	36.80	33.78
LASER	9.65	37.23	33.58
bi-LSTM	6.13	37.52	33.96
∩ALL	5.80	37.57	33.93
random	5.80	36.31	33.00

Table 3: BLEU scores of the XLM and ConvS2S translation engines. Corpus size (#SPs) is in millions of sentencepairs. ¬Flagged is the part of MT-TRAIN that has not been flagged by a professional translator (see Section 3).

In the left part of Figure 3, we plot the BLEU scores obtained on the test set over epochs by the different XLM translation engines we trained. We observe similar training curves (including a notable increase of performance at epoch 10) for all systems, and that filtering the MEGACORPUS leads to gains at each epoch.



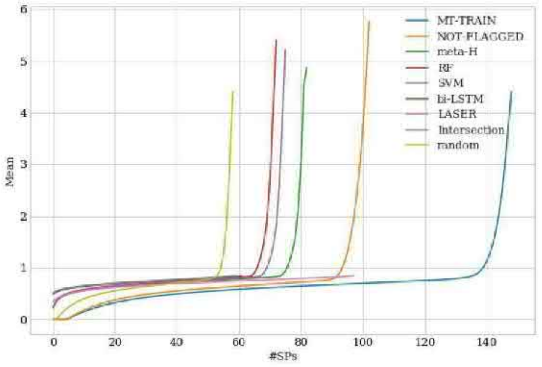


Figure 3: Left plot: BLEU scores of XLM over the epochs for the different training sets we considered. Right plot: average length-ratio of sentence pairs of the different sub-corpora of MT-TRAIN as a function of the number of slices of 100k SPs considered.

6.3 Analysis

Our manual evaluation, although conducted on a limited number of SPs (2021), demonstrates that our cleaning approaches are effective. This is also confirmed by BLEU improvements. The nature of the cleaning performed is however not clear. We investigate this issue in the following.

6.3.1 Domain Specificity

One might think that the cleaning method is only performing some kind of domain adaptation, although we did make sure to have a balanced set of domains (client names) in our test set. We compared the distribution of domains in the MEGACORPUS (P) to the one in MT-TRAIN (Q) and its filtered versions, using the KL divergence:

$$\mathbf{D}_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

We observed small divergence values: all filtered subsets leading to a divergence very close to that between MEGACORPUS and MT-TRAIN (KL=0.0110). This means that the distributions are close and that the gains in BLEU scores are not due to a domain bias.

6.3.2 Combining Methods

We observed that the different filtering methods we tested remove different portions of MT-TRAIN. We measured the percentage of sentence pairs that passed the meta-H, bi-LSTM and LASER filters and observed that 93.4% of SPs detected by LASER as bad were also detected by bi-LSTM, while the agreement of bad SPs between LASER and either META-H or RF is around 65%. If we remove SPs detected bad by these three methods, we end up with a subset of MT-TRAIN which contains 5.8M sentence pairs, the performance of which is reported in line $\cap$ALL of Table 3. BLEU scores are not significantly different from those obtained with bi-LSTM alone, which tends to indicate that the deep supervised classifier does not benefit from other methods. As a sanity check, we randomly selected from MT-TRAIN a subset with the very same number of SPs and observed a drop in BLEU score of -1.26 and -0.93 for XLM and ConvS2S respectively (see the last line of Table 3). This supports the claim that our methods do perform cleaning of the translation memory.

6.3.3 Unknown Words

Because both translation engines are using a fixed set of 60K BPE subword units, some target words in the training material can not be reproduced by the systems. For MT-TRAIN, we measure over 76k such token types for the XLM translation engine. Filtering the training material with feature-based classifiers, bi-LSTM and LASER, lowers this rate to around 3k, 450 and 17k token types respectively, a clear reduction, especially with bi-LSTM. By inspecting why it was so, we observed that the vast majority of types that could not be reproduced by concatenating BPE units were gibberish words (e.g. `t0DÉlmsäoyCœ`). This indicates that our cleaning approaches are taking care of this problem.

6.3.4 Length-ratio

We report in the right plot of Figure 3 the length-ratio of sentence pairs in the different sub-corpora identified from MT-TRAIN. Length is counted in words, and we report the average length-ratio ($|\text{en}|/|\text{fr}|$) computed on increasing slices of 100k sentence pairs. Prior to computing averages, SPs in a corpus are sorted in increasing order of length-ratio.

The (blue) curve is the distribution obtained on the MT-TRAIN corpus. It starts with near zero mean (much larger French sentences), then gradually increases to a ratio slightly lower than one (English sentences are typically shorter than French ones), and finally peaks at around 4, the average over the full corpus. Near zero and high ratio very likely indicate alignment problems.

We observe that applying meta-heuristics or feature-based classifiers reduces near-zero and high ratios to some extent. It is much more noticeable that bi-LSTM and LASER remove most of them, leading to an average length-ratio of around 0.8. This observation further supports the fact that cleaning is being performed.

6.3.5 Other Evidence of Cleaning

We computed the ratio of sentence pairs in which a `www` token is in the English sentence but not in the French part. We have 25 203 such sentence pairs in MT-TRAIN, over 57 925 pairs with `www` tokens that do match, a ratio of 43.5%. Feature-based classifiers reduce this ratio to around 10.5%, while bi-LSTM and LASER lower the ratio to 3.1% and 4.9% respectively.

Finally, we manually annotated 100 sentence pairs belonging to the $\cap$ALL set in Table 3, that is, SPs classified good by all approaches, as well as 100 sentence pairs classified bad by both deep learning methods. We found 16 false positives (SPs wrongly identified as good) in the former set, and 33 false negatives (SPs wrongly identified as bad) in the latter set. This last figure suggests that deep learning methods are too strict noise detectors.

7 Conclusions

In this work, we report on our path to filtering a mostly clean Translation Memory for professional use. Our experiments show that, among the different methods tested, the pre-trained LASER and the in-house trained bi-LSTM are able to discriminate bad from good sentence pairs with high accuracy. The former approach is unsupervised and delivers the best results on our small scale manual evaluation. These two methods outperform heuristics we devised specifically for this task, as well as feature-based classifiers we trained on datasets selected using these heuristics. It also clearly outperform the TMOP system which turned out to be very challenging to deploy.

By filtering a large training set using our methods, we obtain machine translation gains. Similarly to our manual evaluation, deep learning methods lead to larger gains in BLEU. The bi-LSTM classifier could remove over half of the training material while improving BLEU by over one point.

In future research, we would like to revisit a number of choices we made. For instance, when we created a balanced corpus of (supposedly) good and bad sentence pairs, we were not very successful in generating artificial SPs with subtle errors. Also, we overlooked the nature of noise we are trying to identify. It is for instance currently difficult to ascertain whether we are able to identify subtle errors (such as bad wordings), an avenue which deserves more investigation.

References

Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond.

Eduard Barbu. 2015. Spotting false translation segments in translation memories. In *Proceedings of the Workshop Natural Language Processing for Translation Memories*, pages 9–16.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606.*

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Vishrav Chaudhary, Yuqing Tang, Francisco Guzmán, Holger Schwenk, and Philipp Koehn. 2019. Low-resource corpus filtering using multilingual sentence embeddings.

François Chollet et al. 2015. Keras. `https://keras.io`.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks.

José GC de Souza, Marco Turchi, and Matteo Negri. 2014. Machine translation quality estimation across domains. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 409–420.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning.

Francis Grégoire and Philippe Langlais. 2018. Extracting parallel sentences with bidirectional recurrent neural networks to improve machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1442–1453, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November.

Masoud Jalili Sabet, Matteo Negri, Marco Turchi, José GC de Souza, and Marcello Federico. 2016. Tmop: a tool for unsupervised translation memory cleaning. pages 49–54.

Marcin Junczys-Dowmunt. 2018. Dual conditional cross-entropy filtering of noisy parallel corpora. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 888–895, Belgium, Brussels, October. Association for Computational Linguistics.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual Language Model Pretraining. *arXiv e-prints*, page arXiv:1901.07291, Jan.

Elliott Macklovitch. 1994. Using bi-textual alignment for translation validation: the transcheck system. In *First Conference of the Association for Machine Translation in the Americas (AMTA-94)*, Columbia, É-U, oct.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Víctor M. Sánchez-Cartagena, Marta Bañón, Sergio Ortiz-Rojas, and Gema Ramírez. 2018. Prompsit's submission to WMT 2018 parallel corpus filtering shared task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 955–962, Belgium, Brussels, October. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August. Association for Computational Linguistics.

Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual nlp. In *The 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2015)*.

Marco Trombetti. 2009. Creating the world's largest translation memory. In *MT Summit*.

Wei Wang, Taro Watanabe, Macduff Hughes, Tetsuji Nakagawa, and Ciprian Chelba. 2018. Denoising neural machine translation training with trusted data and online data selection. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 133–143, Brussels, Belgium, October. Association for Computational Linguistics.

Hainan Xu and Philipp Koehn. 2017. Zipporah: a fast and scalable data cleaning system for noisy web-crawled parallel corpora. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2945–2950, Copenhagen, Denmark, September. Association for Computational Linguistics.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method.

OSTI: An Open-Source Translation-memory Instrument

Anonymous COLING submission

Abstract

We present OSTI: a free open-source tool to process and visualize a pair of bilingual documents (original and translation) into automatically labeled sentence pairs. This can be used by translation professionals as a human-accessible quality evaluation tool, as a pre-processing step for human annotation as well as an intermediate step to populate a Translation Memory.

1 Introduction

The development of Computer-Assisted Translation (CAT) tools started gaining popularity in the mid 1980s. Since then, the elaboration of new and more sophisticated CAT tools has not ceased to increase. Given that training data is essential to the development of high-end automatic models, there has been great academic and (sometimes) corporate efforts to make large and clean Translation Memories (TM) and translation data sets (bilingual and monolingual dictionaries, terminologies, etc.) publicly available. Works such as (Koehn, 2005; Tiedemann, 2012; Steinberger et al., 2013) have greatly facilitated the elaboration of pioneering CAT tools *freely* available today.

Nevertheless, a problem arises when users have their own proprietary data and want to use it instead of the general and publicly available one. Multiple companies offer to tailor-made an exclusive TM using proprietary data but these services can be black-boxes with untraceable quality, unaffordable to the more humble translators or translation companies, or apprehensive to companies working with sensitive data.

Having this in mind, we designed a simple yet (hopefully) useful open-source tool which takes as input a pair of supposedly parallel documents in English and French.[1] These are segmented, aligned into units whose quality is automatically inspected, labeled, and presented as an easy to consume HTML (an example of which is reported in Figure 1). They can subsequently be distilled into a TMX format.

Our tool can be used as a human-accessible quality evaluation tool, as a pre-processing step for human annotation, as well as an intermediate step to populate a Translation Memory.

2 System overview

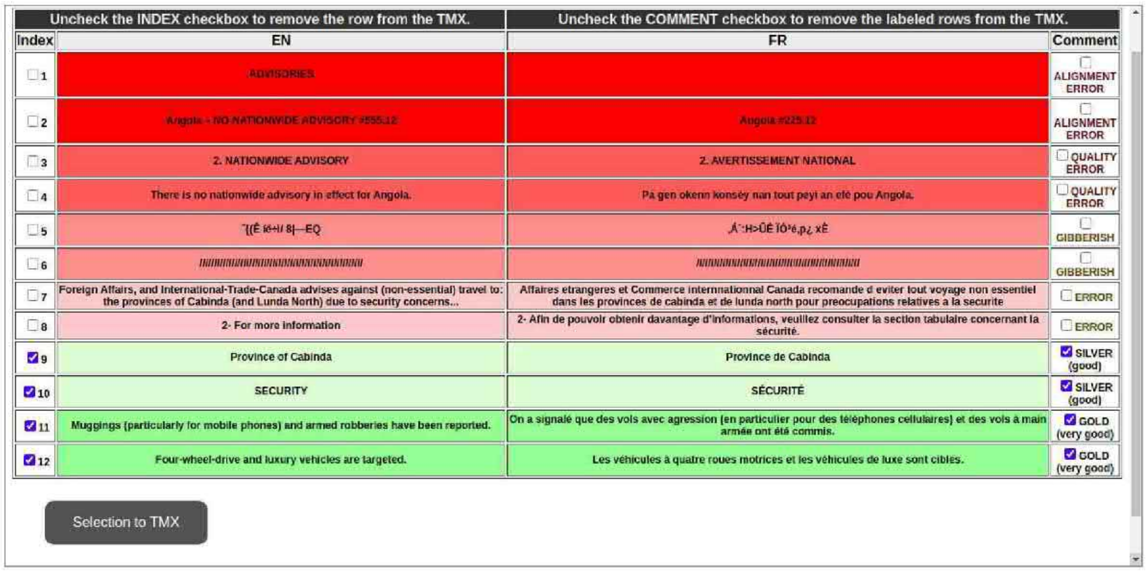
We conceived OSTI in a modular fashion, trying to make the integration of new components, tools or language pairs as easy as possible. Its has been currently tested on the French-English language pair (our use case), but it should apply with minor or no adaptation to other language pairs.

The overall pipeline, depicted in Figure 2, contains 4 modules: a) we first segment each text into sentences, that b) we align at the sentence level; c) the sentence pairs (SPs) are then classified into good or bad SPs; d) the SPs are further labelled into 6 classes for the sake of human readability on an HTML interface that allows to export the automatically or manually selected SPs into a TMX file.

2.1 Sentence Segmenter

Even though the task of sentence segmentation is not a very enticing one, it is crucial to have a clean sentence segmentation to start with in order to reduce the subsequent tasks. We use the NLTK sentence tokenizer as a default, although we also considered Spacy (spaCy, 2017) and the `Mediacloud` Sentence

[1]We targeted English and French languages, but arguably many components could be used for other language pairs. To keep it simple, we also assume that documents are converted into text prior to using OSTI.



Uncheck the INDEX checkbox to remove the row from the TMX.		Uncheck the COMMENT checkbox to remove the labeled rows from the TMX.	
Index	EN	FR	Comment
☐ 1	ADVISORIES		☐ ALIGNMENT ERROR
☐ 2	Angola - NO NATIONWIDE ADVISORY #555.12	Angola #225.12	☐ ALIGNMENT ERROR
☐ 3	2. NATIONWIDE ADVISORY	2. AVERTISSEMENT NATIONAL	☐ QUALITY ERROR
☐ 4	There is no nationwide advisory in effect for Angola.	Pa gen okenn konsèy nan tout peyi an efè pou Angola.	☐ QUALITY ERROR
☐ 5	˜[(Ê ié÷I/ 8|—EQ	‚Á¨:H>ÛÊ ÏÓ³é,p¿ xÊ	☐ GIBBERISH
☐ 6	//	//	☐ GIBBERISH
☐ 7	Foreign Affairs, and International-Trade-Canada advises against (non-essential) travel to: the provinces of Cabinda (and Lunda North) due to security concerns...	Affaires etrangeres et Commerce internnationnal Canada recomande d eviter tout voyage non essentiel dans les provinces de cabinda et de lunda north pour preocupations relatives a la securite	☐ ERROR
☐ 8	2- For more information	2- Afin de pouvoir obtenir davantage d'informations, veuillez consulter la section tabulaire concernant la sécurité.	☐ ERROR
☑ 9	Province of Cabinda	Province de Cabinda	☑ SILVER (good)
☑ 10	SECURITY	SÉCURITÉ	☑ SILVER (good)
☑ 11	Muggings (particularly for mobile phones) and armed robberies have been reported.	On a signalé que des vols avec agression (en particulier pour des téléphones cellulaires) et des vols à main armée ont été commis.	☑ GOLD (very good)
☑ 12	Four-wheel-drive and luxury vehicles are targeted.	Les véhicules à quatre roues motrices et les véhicules de luxe sont ciblés.	☑ GOLD (very good)

Figure 1: Screenshot of OSTI's HTML visualization. Each sentence pair is presented in different colours according to their estimated quality (see text for details). As a default, `Gold` and `Silver` (right column) sentence pairs are selected for conversion to TMX, but it is up to the user to change this selection by removing/adding individual sentence pairs (*Index* checkbox on the left) or by removing/adding all sentence pairs having the same label (*Comment* checkbox on the right).

Splitter.[2] A small empirical analysis showed that NLTK outputs the best out-of-the-box results for our specific language pair, we therefore selected it as our default segmenter.

2.2 Bilingual Sentence Aligner

We benchmarked two very different tools that have both shown to be accurate and robust: YASA (Lamraoui and Langlais, 2013) and VECALIGN (Thompson and Koehn, 2019). The former system is very similar to `HunAlign` (Varga et al., 2007), that is, a sentence-length score (Gale and Church, 1993) enhanced with a cognate-based one (Simard et al., 1992), and has been reported faster and more accurate than more elaborated systems such as BMA (Moore, 2002). The VECALIGN system uses a more innovative scoring function specially made to work with sentence embedding vectors. Although this system is said to accommodate embeddings from various toolkits, we used the recommended multilingual

[2]https://github.com/berkmancenter/mediacloud-sentence-splitter (based on the implementation by (Koehn, 2005))

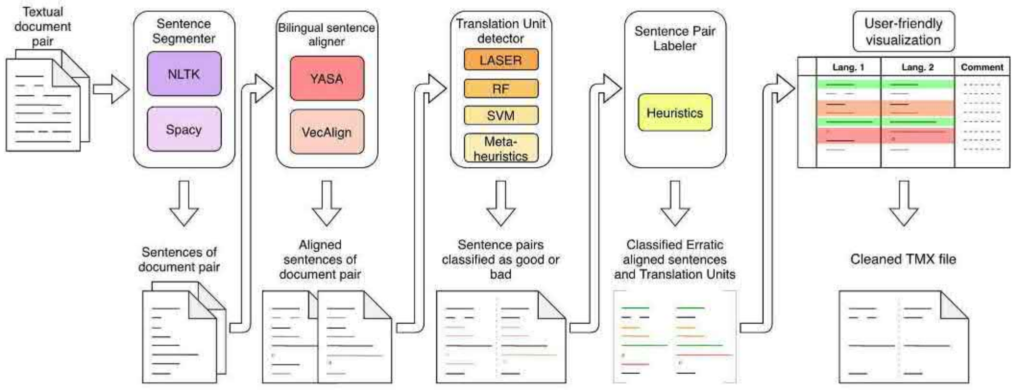

Textual document pair
Sentence Segmenter
NLTK
Spacy
Sentences of document pair
Bilingual sentence aligner
YASA
VecAlign
Aligned sentences of document pair
Translation Unit detector
LASER
RF
SVM
Meta-heuristics
Sentence pairs classified as good or bad
Sentence Pair Labeler
Heuristics
Classified Erratic aligned sentences and Translation Units
User-friendly visualization
Lang. 1
Lang. 2
Comment
Cleaned TMX file

Figure 2: OSTI pipeline, from document pair to visualization. Top (darker) components are defaults.

Language-Agnostic SEntence Representations (LASER) from Artexte and Schwenk (2019).

Since our target language pair is French-English, we compared both aligners using the BAF corpus benchmark (Simard, 1998) which contains 11 document pairs of 4 different genres (Literary, Institutional, Scientific, and Technical) segmented into approximately 25k sentences (in each language) aligned and manually checked. To evaluate the aligners, we used the benchmark's metrics of precision, recall and F1 at the alignment pair level and at the sentence level. See (Langlais et al., 1998) for a description of those metrics.

		Literacy		Institutional		Scientific		Technical		**All doc.**	
		YASA	VECA	YASA	VECA	YASA	VECA	YASA	VECA	YASA	VECA
align.	Prec.	0.59	0.61	0.94	0.95	0.86	0.86	0.85	0.86	0.86	**0.87**
	Rec.	0.74	0.71	0.95	0.95	0.93	0.91	0.96	0.95	**0.92**	0.91
	F_1	0.65	0.65	0.94	0.95	0.89	0.88	0.90	0.90	**0.89**	**0.89**
sent.	Prec.	0.88	0.87	0.98	0.98	0.99	0.98	0.99	0.98	**0.98**	0.97
	Rec.	0.79	0.84	0.94	0.95	0.86	0.86	0.06	0.06	0.81	**0.82**
	F_1	0.83	0.86	0.96	0.96	0.92	0.91	0.11	0.11	**0.85**	**0.85**

Table 1: Alignment and sentence Precision, Recall and F_1 scores of YASA and VECALIGN as a function of text genre on the BAF benchmark. The last columns shows the average over all document pairs, over all genres.

Results are reported in Table 1. Two observations can be made. First, both systems deliver very similar performances. Second, the performance varies substantially depending on the text genre, the worst setting is when aligning literary texts.[3] We refer the reader to (Xu et al., 2015) for extensive comparisons of alignment techniques on literary texts. Because both systems perform on par, we selected YASA as our default sentence aligner since is it much lighter than VECALIGN in terms of hardware (CPU versus GPU).[4]

2.3 Translation Unit Detector

This component decides whether a given sentence pair contains a problem or not. Sentence pairs that are classified as good are promoted to *Translation Units* (TU) and can be saved in TMX format for further consumption. In order to do so, we implemented 5 families of classifiers, described in details elsewhere (Anonymous, 2020): a) a heuristic-based approach (META-HEURISTICS in Figure 2) involving 13 heuristics also described in Sub-section 2.4; b) statistical classifiers using 62 features (SVM and RF in Fig. 2) and c) a cleaning method devised in-house on top of the LASER model developed and pre-trained by (Artetxe and Schwenk, 2019).

We compared those classifiers in terms of accuracy on a manually annotated proprietary corpus of 2021 sentence pairs and found LASER to largely outperform the feature-based classifiers (84% accuracy vs. 63%) and meta-heuristics (42%). Considering that LASER is unsupervised, and relatively fast to use, we selected it as our default detector.

2.4 Sentence Pair Labeler

There are many reasons why sentence pairs can be detected erroneous, including bad sentence alignment (often caused by sentence segmentation issues), errors in translations (calques, false friends, etc.), as well as encoding issues (which happen in complex organisations due to numerous format manipulations). All of these are considered errors but not all are equally important to the translation professional. Therefore, we take an extra step into further labelling sentences pairs into 6 labels described below. We do this by taking advantage of the 13 heuristics used in the META-HEURISTIC component of the TU detector. Each heuristic detects a specific type of error.

[3]There is one pair of texts in BAF belonging to this category which is a novel of Jules Vernes where the English version is abridged, which confuses the dynamic programming optimization driving each approach.

[4]VECALIGN accommodates CPU computations, at the expense of much slower response time.

`Aligt` qualifies a major mismatch, such as numerical entity issues (row 2 in Figure 1), sentence length (row 1 in Figure 1), etc. In total, 8 heuristics are used.

`Quality` characterizes mistakes identified by 5 heuristics: misspelling issues (row 4 in Figure 1), false-friends, sentences that are part of table of content or indexes and are often misaligned (row 3 in Figure 1), and non-translated target sentences (which happens when documents are partly translated).

`Gibberish` qualifies sentence pairs that contain mainly gibberish (row 6 in Figure 1), sometimes due to encoding issues (row 5 in Figure 1).

`Error` is used when an alignment problem is detected but can not be attributed to a specific cause. Row 7 (punctuation mismatch and misspelling issues) and row 8 (table of content detection and length mismatch) in Figure 1 are examples, where both `Aligt` and `Quality` compete.

`Silver` is used when the TU detector classifies a sentence pair as good but at least one heuristic indicates the presence of a problem (row 9 and 10 in Figure 1).

`Gold` qualifies SPs that are classified good by the detector, and for which no heuristic indicates any problem (row 11 and 12 in Figure 1).

2.5 Technical details

Our Github project[5] requires the installation of a small amount of modules (such as NLTK, NUMPY, FAISS or PYTORCH) that are specified in our requirements file and are easily installed using the standard `pip` package manager or by executing the setup file. It also requires the installation of more complex tool-kits (i.e., VECALIGN and LASER). The YASA tool is provided in compiled form and has only been tested to work on (multiple) Debian-based Linux distributions. Finally, it also includes all the in-house algorithms/implementations written in Python as well as some pre-trained models mentioned in Section 2.3. This allows to run OSTI without having installed LASER, only using a functional yet lesser classifier.

To ensure its viability, we fully tested OSTI on a computer with 30Gb of RAM, a 12-core CPU and a GeForce GT 1030 GPU (used by LASER and VECALIGN) using a Debian-based Linux Operating System.

We also measured its response time of on 2 document pairs containing around 900 sentences each (1 800 sentences/160k characters in total). On average, sentence segmentation took less than 0.5 seconds. Sentence alignment took 13 seconds with YASA (on CPU) and 17 seconds with VECALIGN (with GPU). For the TU detector, the fastest is the heuristic-based approach (112 seconds, CPU), followed by LASER (117 seconds, GPU) and the 2 feature based classifiers (SVM: 131 seconds, RF: 239 seconds, both using a single CPU). Finally, to save time, the sentence pair labeler is run concurrently to the TU detector.

3 Conclusion

We presented OSTI, an open-source tool which detects good from bad sentence pairs in a French-English pair of (supposedly) parallel documents. These are then further labelled into a set of 6 labels that can be inspected with a simple Web browser and easily transformed into a TMX file. OSTI can be used as a human-accessible quality evaluation tool, as a pre-processing step for human annotation, as well as an intermediate step to populate a Translation Memory. We are aware of proprietary solutions, but there is a striking absence of open-source and peer reviewed such systems.

Currently, we targeted the English-French langage pair, which we plan to revisit. We do not anticipate much difficulties since most components involved in OSTI are arguably language agnostic. Also, we distribute a simple batch pipeline, while for professional use, a client-server application may be more appropriate. In benchmarking embedded components, we were surprised by the fact that a simple sentence aligner was performing on par with a more recent one relying on sentence embeddings. We plan to revisit this benchmarking on other language pairs and conditions.

[5]Anonymized URL.

References

Anonymous. 2020. Cleaning a(n almost) clean institutional translation memory. submitted.

Mikel Artetxe and Holger Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610.

William A. Gale and Kenneth W. Church. 1993. A program for aligning sentences in bilingual corpora. *Comput. Linguist.*, 19(1):75–102, March.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit X*, pages 79–86.

Fethi Lamraoui and Philippe Langlais. 2013. Yet another fast, robust and open source sentence aligner. time to reconsider sentence alignment? In K. Sima'an, M.L. Forcada, D. Grasmick, H. Depraetere, and A. Way, editors, *MT Summit XIV*, pages 77–84.

Philippe Langlais, Michel Simard, and Jean Véronis. 1998. Methods and Practical Issues in Evaluating Alignment Techniques. In *36th Annual Meeting of the Association for Computational Linguistics (ACL) and 17th International Conference on Computational Linguistic (COLING)*, pages 711–717, Montreal, Canada, Aug.

Robert C. Moore. 2002. Fast and accurate sentence alignment of bilingual corpora. In *Proceedings of the 5th Conference of the Association for Machine Translation in the Americas on Machine Translation: From Research to Real Users*, AMTA '02, pages 135–144.

Michel Simard, George Foster, and Pierre Isabelle. 1992. Using cognates to align sentences in bilingual corpora. In *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 67–81.

Michel Simard. 1998. The BAF: a corpus of english-french bitext. In *First International Conference on Language Resources and Evaluation*, pages 489–494.

spaCy. 2017. Industrial-strength natural language processing in python. `https://spacy.io`. (accessed 24-jun-2020).

Ralf Steinberger, Andreas Eisele, Szymon Klocek, Spyridon Pilos, and Patrick Schlüter. 2013. DGT-TM: A freely available translation memory in 22 languages. *arXiv preprint arXiv:1309.5226.*

Brian Thompson and Philipp Koehn. 2019. Vecalign: Improved sentence alignment in linear time and space. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1342–1348, Hong Kong, China, November. Association for Computational Linguistics.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey, May. European Language Resources Association (ELRA).

Dániel Varga, Péter Halácsy, András Kornai, Viktor Nagy, László Németh, and Viktor Trón. 2007. Parallel corpora for medium density languages. *Amsterdam Studies In The Theory And History Of Linguistic Science Series 4*, 292:247.

Yong Xu, Aurélien Max, and François Yvon. 2015. Sentence Alignment for Literary Texts. *Linguistic Issues in Language Technology*, 12:1–25.

Trends, Limitations and Open Challenges in Automatic Readability Assessment Research

Anonymous COLING submission

Abstract

Readability assessment is the task of evaluating the reading difficulty of a given piece of text. Although research on computational approaches to readability assessment is now two decades old, there is not much work on synthesizing this research. This article is a brief survey of contemporary research on developing computational models for readability assessment. We identify the common approaches, discuss their shortcomings, and identify some challenges for the future. Where possible, we also connect computational research with insights from related work in other disciplines such as education and psychology.

1 Introduction

Automatic Readability Assessment (ARA) refers to the task of modeling the reading and comprehension difficulty of a given piece of text (on some pre-defined scale), for a given target audience. This has a broad range of applications in both machine-facing and human-facing scenarios. Some examples of human facing scenarios are: choosing appropriate reading materials for language teaching (Collins-Thompson and Callan, 2004), supporting readers with learning disabilities (Rello et al., 2012) and self-directed learning (Beinborn et al., 2012). In machine facing scenarios, ARA is used in research on information retrieval for ranking search results by their reading level (Kim et al., 2012), in generating translated text controlled for reading level (Marchisio et al., 2019; Agrawal and Carpuat, 2019), and evaluating automatic text simplification (Alva-Manchego et al., 2020), to name a few. A well-known real-world application of ARA is TextEvaluator ™[1], used to determine whether a reading material is appropriate for a grade level in classroom instruction. Apart from such uses around and within the field of NLP, the general idea of readability assessment is used in a range of other scenarios. A common case is of medical research, where it was used for assessing patient education materials (Sare et al., 2020) and consent forms (Perni et al., 2019; Lyatoshinsky et al., 2019), for example. Even though a lot of ARA research in NLP is skewed towards educational applications, this broad application range highlights ARA as one of the important applications of NLP.

Research into measuring how difficult (or easy) is a text to read is almost a century old (e.g., Thorndike (1921), Lively and Pressey (1923), Vogel and Washburne (1928)). Such early research focused on creating lists of difficult words and/or developing a "formula" for readability which is a simple weighted linear function of easy to calculate variables such as number/length of words/sentences in a text, percentage of difficult words etc. This resulted in several readability formulas such as Flesch Reading Ease (Flesch, 1948), SMOG (McLaughlin, 1969), Dale-Chall readability formula (Dale and Chall, 1948) etc. (see Dubay (2007) for a detailed survey of such formulae).

NLP researchers started taking interest in this problem only in the past two decades. Si and Callan (2001) were the first to use statistical approaches for ARA, to our knowledge. Since then, from statistical language models and feature engineering based machine learning approaches to more recent deep neural networks, a range of approaches have been explored so far. Despite this, to our knowledge, a lot of scenarios involving the use of ARA rely on traditional formulae even within NLP. For example, Marchisio et al. (2019) uses the "traditional formulae" such as Dale-Chall, Flesch Reading ease etc. as a measure

[1] `https://textevaluator.ets.org/TextEvaluator/`

of readability to control the reading level of machine translated text. In the scenarios outside NLP, such as the use cases in medical research mentioned earlier too, one would notice the strong domination of traditional formulae. Possible reasons for this situation could be a lack of awareness of the state of the art in ARA or difficulty in using it easily for their purpose.

Analyzing the reasons for this scenario would require a detailed survey of ARA research to understand the limitations in its adaptability. To our knowledge, there has only been one comprehensive ARA survey (Collins-Thompson, 2014) so far. There have been a lot of newer approaches to ARA since then, and researchers in other disciplines such as education have also published their perspectives on validation and evaluation of ARA approaches (e.g., Hiebert and Pearson (2014)). In this background, this paper aims to take a fresh look at ARA considering inputs from other disciplines where needed, and also cover recent research on various aspects of ARA.

We start with an overview of the topic (Sections 1 and 2) and summarize contemporary ARA research in NLP by identifying some common trends (Section 3). We then discuss their shortcomings (Section 4) in an attempt to understand why this large body of research is not reflected in its usage in various application scenarios. Finally, we identify some challenges for future research (Section 5). Where possible, we will draw insights from work done in other disciplines as well. Note that we use the terms readability and text complexity interchangeably in this paper, as is common in NLP research, although one can see more fine grained difference between the usage of these terms in education or psychology literature (e.g., Valencia et al. (2014)). We hope that this survey would be especially useful for three kinds of readers:

1. NLP Researchers specifically working on ARA and other related problems (e.g., text simplification) may find this survey useful to understand current trends and identify challenges for future research.

2. Other NLP researchers can get a general overview of ARA and how to incorporate it into their systems.

3. Researchers from other disciplines looking to use ARA for their research can get an overview of the state of research in the field and what they can use easily.

2 Related Work

While there was been a lot of work in the NLP community on developing computational models for readability assessment across languages, there has not been much work synthesizing this research. Collins-Thompson (2014) is the most recent, comprehensive survey on this topic, to our knowledge. He gave a detailed overview of the various approaches to ARA and identified the development of user-centric models, data driven measures that can be easily specialized to new domains, and the inclusion of domain/conceptual knowledge into existing models as some of the potential research directions for future. François (2015) presented a historical overview of readability assessment focusing on early research on traditional formulae and identified three challenges for future work - validity of the training data, developing ARA approaches for different domains, and difficulty in estimating readability at different granularities (e.g., words and sentences).

Outside of NLP, Nelson et al. (2012) compared and evaluated a few existing proprietary text difficulty metrics (for English) using a range of reading difficulty annotated corpora and assessed the implications of such measures for education. A few years ago, the Elementary School Journal published a special issue on understanding text complexity (Hiebert and Pearson, 2014), which offered multi-disciplinary perspectives on various aspects of ARA and its implications to education. Concluding that readability involves dimensions other than text and much more research is needed on the topic, the special issue cautioned about the danger of focusing on text readability scores alone. While these are not survey articles, we include them here as they summarize the findings from research on this topic that, we believe, is not common knowledge in NLP research.

In the current survey, we will draw inputs from this existing body of research from NLP and other disciplines, focus on more recent developments in ARA research, understand their limitations and reassess current challenges for ARA. We primarily focus on research in the past two decades, and on papers describing computational approaches. Our goal in this paper is to provide a general overview of the trends in research and not to have an exhaustive listing of all published research on this topic during this period. While we aim to remain language agnostic in this study, we can't avoid an over representation of English.

3 Current Trends in ARA Research in NLP

ARA is generally modeled as a supervised machine learning problem in NLP literature. Hence, a typical ARA approach follows the pipeline depicted in Figure 1.

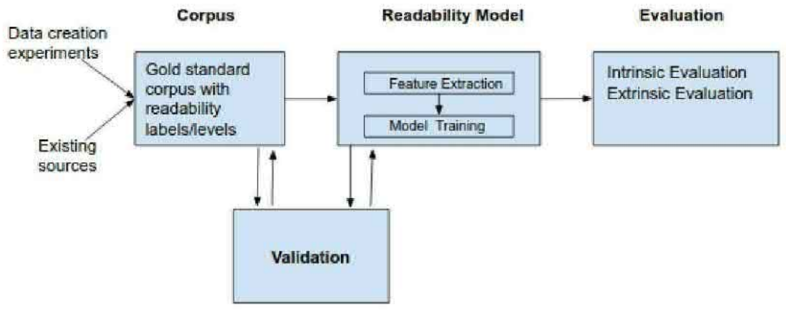


Figure 1: Typical ARA pipeline

ARA approaches rely on a gold standard training corpus annotated with labels indicating reading level categories, or numbers indicating a graded scale, which is either already existing or created specifically for this task (**Corpus**). As with any machine learning problem, the next step consists of feature extraction and training a model (**Readability Model**). The final step in this process is an evaluation of the effectiveness of the model (**Evaluation**). A not so commonly seen, but essential step in this process is **Validation**. Rest of this section discusses each of these steps in detail by giving an overview of representative approaches taken by researchers in handling these stages of ARA, and what changed in the recent few years, compared to Collins-Thompson (2014)'s survey.

3.1 Corpus

Training data in ARA came from various sources. They can be broadly classified into two categories: expert annotated and non-expert annotated. Textbooks, or other graded readers carefully prepared by trained authors targeting audience at specific grade levels can be termed as "expert annotated". These are the most common forms of training data seen in ARA research. On the other hand, some ARA work also relied on available web content, or doing crowd sourcing experiments and user studies to collect data. Figure 2 summarizes the different forms of data sources in ARA research, and we will discuss each of them in detail.

Textbooks: Textbooks have been a common source of training data for ARA research, where available, for several languages such as English (Heilman et al., 2007), Japanese (Sato et al., 2008), German (Berendes et al., 2018), Swedish (Pilán et al., 2016), French (François and Fairon, 2012) and Bangla (Islam et al., 2012), to name a few. They are considered to be naturally suited for ARA research as one would expect the linguistic characteristics of texts to become more complex as school grade increases. Following a similar approach to using textbooks, Xia et al. (2016) collected reading comprehension passages from language exams conducted at different proficiency levels for building ARA models. However, it is not always possible to have a readily accessible dataset of textbooks, as many textbooks are also under copyright or in an undigitized format. Thus, most of the above mentioned corpora are not available for other researchers. A closer alternative is to use graded readers.

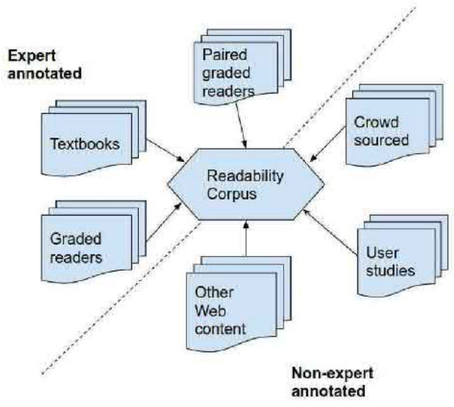


Figure 2: Various kinds of ARA Corpora

Graded Readers: We refer to non-textbook reading materials prepared by teachers or other experts, which are separated into some categorization of reading levels, as graded readers. Typically, these materials are derived from news articles rewritten to suit the target reading level or from encyclopedia articles written for adults and children separately. WeeBit (Vajjala and Meurers, 2012) is one of the widely used graded reader corpus used for English ARA. In the recent past, corpora such as Newsela (Xu et al., 2015) and OneStopEnglish (Vajjala and Lučić, 2018) were created for English, which can be called **Paired Graded Readers**. Instead of having a collection of unrelated documents at each reading level, these corpora have the same documents rewritten to suit different reading levels. Some of them are also aligned at the level of paragraphs and sentences. Newsela corpus was also used to build text simplification systems (Štajner and Nisioi, 2018) and generating machine translated text at varying reading levels (Agrawal and Carpuat, 2019) in the past.

Other web content: In all these cases we saw above, it is assumed that we have an existing source of texts grouped into some kind of reading levels/grades. However, this is not always the case, and especially not when developing an ARA approach for a new language. Hence, it is very common to find other documents from the web which have some form of inherent reading level grouping. Simple Wikipedia[2] was widely used along with Wikipedia to build a easy versus difficult ARA system for English (Napoles and Dredze, 2010). A sentence aligned version of this was also used for modeling text simplification (Hwang et al., 2015). Other such websites have been used in other ARA approaches for English (Vajjala and Meurers, 2013), German (Hancke et al., 2012), Italian (Dell'Orletta et al., 2011) and Basque (Gonzalez-Dios et al., 2014) among others. Taking a slightly different approach, Eickhoff et al. (2011) relied on the topic hierarchy in Open Directory Project to group web articles based on whether they are appropriate to a certain age group. Vajjala and Meurers (2014a) used a corpus of TV program subtitles grouped into three age groups, collected from BBC channels. In the absence of readily available corpora annotated with reading levels, this seems to be the most common way of procuring some form of leveled text corpus.

Crowdsourcing: All the above mentioned approaches relied on some form of an existing data source suitable for training ARA models. De Clercq et al. (2014) described the usefulness of a crowdsourcing for ARA, where non-expert readers/general public are shown two texts (in Dutch) each time and are asked to compare them in terms of their reading difficulty. Comparing these judgements with expert (e.g., teacher) judgements, they concluded that crowdsourcing is a viable alternative to expert annotations for this task.

User studies: Another way to gather an ARA corpus is by conducting user studies. For example, vor der Brück et al. (2008) conducted a user study with 500 German documents from municipal domain,

[2]`https://simple.wikipedia.org/`

and non-expert readers were asked to rate the texts on a 7 point Likert scale (Likert, 1932) and used it to construct an ARA model. Similarly, Pitler and Nenkova (2008) conducted a user study where college students were asked to rate WSJ news articles on a scale, which was then used to build a readability model. Štajner et al. (2017) collected user judgements of sentence level text complexity in the context of text simplification, for original, manually and automatically simplified sentences. Some studies conducted such studies to gather expert annotations as well. For example, Kate et al. (2010) described a dataset collected through a user study, rated separately by experts and naive readers. Shen et al. (2013) used a dataset collected and annotated by experts, in four languages - Arabic, Dari, English, and Pashto. However, user studies are not a common mode of corpus creation for ARA, owing the time and effort involved. They also typically result in smaller datasets compared to other approaches for this task.

Somewhat related to corpus creation is the research on the creation of wordlists annotated with some form of difficulty level (Gala et al., 2013; François et al., 2014; François et al., 2016), which are then used as features for ARA (e.g., % of difficult words in a text). Amongst these different forms of resources, excepting paired graded readers and very few cases from "other web content", the texts/vocabulary at different reading levels in the corpora don't necessarily deal with the same content. For example, in the WeeBit corpus (Vajjala and Meurers, 2012), one of the commonly used corpus for English, articles tagged with different reading levels don't share the same topical content. As we will see in the next subsection, a majority of ARA models do not particularly control for topic variation. This leads us to question what the ARA models learn - is it a notion of text complexity, or topical differences among texts? Further, whether these corpora are validated to be appropriate for the target audience is another important concern, not typically addressed in ARA research. In this background, we can conclude that not much has happened in the direction of corpora creation since Collins-Thompson (2014)'s survey, and many questions remain.

3.2 Readability Model

The second step in ARA pipeline is to build the readability model. Research into building readability models in the past two decades has primarily relied on language models and feature engineering based machine learning approaches. Recent approaches used various deep learning architectures, along with different forms of text embeddings.

Features that are expected to influence the readability of a text come in various forms, from some simple, easy to calculate numbers such as number of words per sentence to more complex ones involving the estimation of a discourse structure in the document. While some of the advanced linguistic features such as coherence and cohesion are potentially hard to extract automatically, shallow variants e.g., noun overlap between adjacent sentences, implemented in Coh-Metrix (Graesser et al., 2004) are commonly used as proxies. Similarly, different kinds of text embeddings, which capture some form of syntactic and semantic properties of texts, do not also need advanced linguistic processing such as parsing, coreference resolution etc. Hence, instead of grouping features based on linguistic categories, as is commonly done, we group them based on the amount of language processing required in this paper. Figure 3 shows a summary of different kinds of features used in ARA research with examples at each step.

Features such as word length (in characters/syllables), sentence length, usage of different forms of word lists (Chen and Meurers, 2018), language models (e.g., Petersen and Ostendorf (2009)), models of word acquisition (Kidwell et al., 2011), measures of morphological variation and complexity (Hancke et al., 2012), syntactic complexity (Heilman et al., 2007; Vajjala and Meurers, 2012), psycholinguistic processes (Howcroft and Demberg, 2017) and other attributes have been extensively used for developing ARA models across languages. Some features relying on advanced processing such as coreference resolution and discourse relations (Pitler and Nenkova, 2008; Feng et al., 2010) have also been explored in the past, more for English, and to some extent for other languages such as French (Todirascu et al., 2013). (Collins-Thompson, 2014) presents a comprehensive summary of different kinds of features used in ARA.

Some recent research focused on learning task specific embeddings (e.g., Cha et al. (2017), Jiang et al. (2018)) for ARA. Although not common, there has also been some work on modeling conceptual

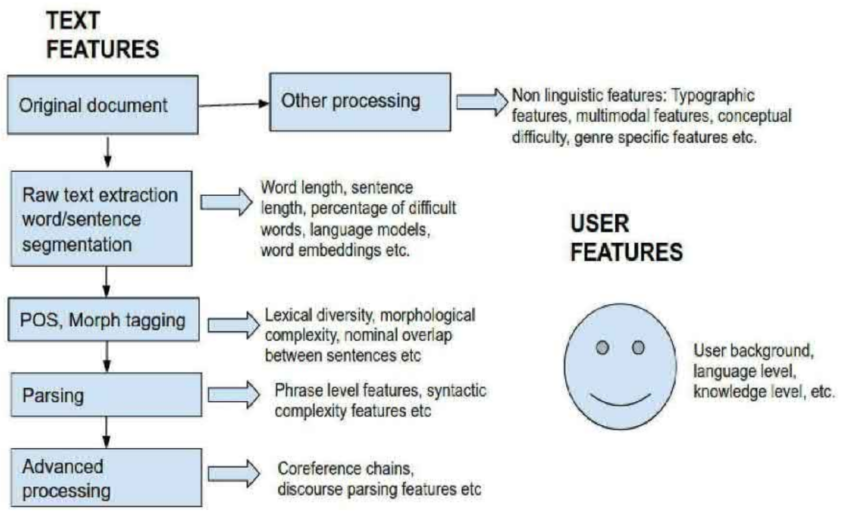

TEXT FEATURES
Original document
Other processing
Non linguistic features: Typographic features, multimodal features, conceptual difficulty, genre specific features etc.
Raw text extraction word/sentence segmentation
Word length, sentence length, percentage of difficult words, language models, word embeddings etc.
USER FEATURES
POS, Morph tagging
Lexical diversity, morphological complexity, nominal overlap between sentences etc
User background, language level, knowledge level, etc.
Parsing
Phrase level features, syntactic complexity features etc
Advanced processing
Coreference chains, discourse parsing features etc

Figure 3: Features used in ARA grouped by the amount of language processing needed

difficulty (Jameel et al., 2012). An often ignored aspect of ARA is the reader. Kim et al. (2012) is one of rare works related to ARA which considers user attributes such as interests, user's language level etc. into their model to rank search results by their reading level. Although not directly about ARA, Knowles et al. (2016) explored the relationship between a word comprehension and a learner's native language. However, though ARA approaches are meant to be for real users in most of the cases, we don't see much work on modeling user features in relation to ARA.

Feature engineering based ARA approaches typically employ feature selection methods to choose a subset of features that best work for the task from a larger set. Apart from generic methods such as information gain, feature correlation etc., genetic algorithm based optimization methods were also explored for this task (De Clercq and Hoste, 2016).

In terms of training methods used, ARA is generally modeled as a supervised learning problem, especially classification. It is, however, not uncommon to see it being modeled as regression (Vajjala and Meurers, 2014b) and ranking (Ma et al., 2012). Heilman et al. (2008) compared different approaches to learn an ARA model and showed that ordinal regression is better suited for the task. Xia et al. (2016) showed that pair wise ranking approach may generalize better compared to classification. Unlike such approaches, Jiang et al. (2019) proposed a graph propagation based approach to ARA, which can potentially consider the inter-relationships between documents while modeling readability.

Like other NLP research, ARA in the past two years has been dominated by neural network based architectures. For example, Mohammadi and Khasteh (2019) proposed a multilingual readability assessment model using deep reinforcement learning and Meng et al. (2020) proposed ReadNet, a hierarchical self attention based transformer model for ARA. Most recently, Deutsch et al. (2020) used a combination of linguistic features and BERT (Devlin et al., 2019) based text representation and showed that linguistic features, which dominated ARA research so far, don't do better than deep learning models.

In general, most readability approaches have been shown to work for one language, or individual models were developed for each language. However, Azpiazu and Pera (2019; 2020) study the development of multilingual and cross-lingual approaches to ARA using deep learning architectures. Finally, while almost all of ARA research has been modeling it as a supervised learning problem, Martinc et al. (2019) compared different supervised and unsupervised approaches to neural text readability. To summarize, we may notice that the past two decades of ARA research closely followed other areas of NLP i.e, traditional feature engineering based methods heavily dominated most of the previous research, whereas recent research seems to see more deep learning based approaches. Compared to the previous survey, most new research seem to have happened in this aspect of ARA.

3.3 Evaluation

Evaluation is the step where we assess how good the ARA model is. Any NLP system can be evaluated either intrinsically or extrinsically. Intrinsic refers evaluating an approach independently, whereas extrinsic refers to evaluating it in a larger system. Intrinsic evaluation is the most common form seen in ARA research. Most of the papers describing ARA models evaluate them in terms of classification accuracy, pearson/spearman correlation (regression/ranking approaches), root mean square error (regression) and other such measures on held-out test data or in a cross-validated setup, as is conventionally done while evaluating supervised machine learning approaches. While it is not a default, we also see multi-corpus evaluation e.g., training on one corpus, testing on many; training and testing on many corpora (Nelson et al., 2012; Vajjala and Meurers, 2014b; Xia et al., 2016). Another way of evaluating if texts predicted by a ARA model to be "simple" result in better comprehension for the target reader group is through a user study. To our knowledge, such an evaluation has not been conducted so far for ARA models.

In terms of extrinsic evaluation, Pera and Ng (2012) and Kim et al. (2012) reported on experiments related to integrating readability approach into a search engine, and applying it for personalized search. Sheehan et al. (2014) deployed ARA models into a real-world tool. However, these examples are more of exceptions than norms, and extrinsic evaluation is rare in ARA research. Thus, we can conclude that intrinsic evaluation is a dominant method of evaluation, and apart from some multi-corpus evaluation experiments, not much has changed since 2014 in this aspect.

3.4 Validation

Validation is the step of assessing the accuracy of a process. We consider validation as a step distinct from evaluation as we are here evaluating the stages before model building, and not the ARA model itself. In the context of ARA research, validation is the step that answers the following two questions:

1. Are the features used to model ARA capable of learning reading level differences in texts?
2. Are the reading level differences annotated in text corpora actually reflected in a reader's experience with the texts? i.e., Does the (annotated) reading level have any relation to reader comprehension?

Although these questions seem obvious, and have been posed many times in non-computational work on text readability in the past (e.g., Cunningham and Anne Mesmer (2014)), there is not much work in this direction in contemporary ARA research. Research related to TextEvaluator (Sheehan et al., 2014; Sheehan, 2017) mentioned earlier has the only detailed analysis in this direction, to our knowledge. However, these are published outside of typical NLP venues, and hence, may not draw the attention of ARA researchers within NLP research. Apart from these, François (2014) conducted a qualitative and quantitative analysis of a French as Foreign Language textbook corpus and concluded that there is a lack of consistent correlation among expert ratings, and that the texts assigned at the same level by the expert annotators showed significant differences in terms of lexical and syntactic features. Berendes et al. (2018) reached similar conclusions using a multidimensional corpus of graded German textbooks covering two school tracks and four publishers.

While there are a few user studies aiming to study the relationship between readability annotations and reader comprehension (Crossley et al., 2014; Vajjala et al., 2016; Vajjala and Lucic, 2019), conclusions have been mixed. The most recent among these, Vajjala and Lucic (2019)'s study concluded that the reading level annotations for texts in a paired graded corpus did not have any effect on reader's comprehension. To summarize, clearly, there is not much work done on validation in ARA research, and this is an area which needs further work.

4 Limitations

Based on this overview of current trends, we identify the following limitations that are potentially preventing the adaption of modern ARA techniques into other research areas within and outside NLP.

1. **Multidimensional and Multimodal ARA models:** - Text readability involves several aspects of text, starting from typographical to linguistic, from conceptual difficulty to deeper pragmatics.

However, contemporary ARA research tens to focus on the surface textual form. Topical or conceptual difficulty is not given much importance. Where it is considered, it is typically not combined with other aspects of readability. Further, texts don't exist in isolation. There are always accompanying non-text data such as tables and/or images in the document. We are not aware of any research that touches upon these aspects in the context of computational modeling. To summarize, there is no framework yet (to our knowledge) that can incorporate a multidimensional, multimodal view of text complexity.

2. **Reader and Task considerations:** Research in education and psychology typically describes text complexity as a combination of text properties, reader (user) characteristics, and task complexity (Goldman and Lee, 2014; Valencia et al., 2014). However, within NLP, ARA research is almost always focused on text, with a small amount of research on reader modeling (Kim et al., 2012). While some research on modeling task complexity started to emerge (Kühberger et al., 2019), We are not aware of any approach that considers task complexity in the context of ARA or combine all the three aspects.
3. **Availability of corpus resources:** While there is clearly a lot of work on ARA across languages, we still don't see a lot of publicly available corpora. Even when available, one has ask whether the corpora suit the target scenario. For example, one cannot use a corpus of textbooks to evaluate ARA models that intend to serve, say, dyslexic readers, as the reading difficulties experienced by dyslexic readers are completely different from first language readers learning subject matter in school. This lack of available (and diverse) corpora can limit the development of ARA models tailored to specific application scenarios.
4. **Availability of ready to use tools:** There is not much of readily usable code artefacts related to building and using ARA models online. While some researchers shared code to reproduce their experiments (e.g., Ambati et al. (2016), Howcroft and Demberg (2017)), there is not much usable code for other NLP researchers or off the shelf tools for researchers from other disciplines. Such tools can potentially be useful for researchers from other disciplines wanting to use readability assessment approaches to answer research questions in their own domains.
5. **Lack of extrinsic evaluation:** Typically, ARA approaches are evaluated intrinsically, using cross validation or held out test set. It is rare to see an extrinsic evaluation when we consider a typical ARA research paper. This makes it particularly hard for practitioners to understand whether an approach works in an applied scenario.
6. **Lack of validation and interpretation:** The most common approach taken in building an ARA model is to take an available corpus, extract various kinds of features, and train different models and compare them. However, there is very little research on whether the corpus is suited for the task, whether the features themselves are actually useful, or if they have a theoretical grounding. Further, it is hard to understand what exactly does a model learn about text complexity. These issues make it difficult for researchers from other domains wanting to adapt modern ARA methods, and they instead turn to traditional formulae, which are relatively straight forward to interpret.
7. **What is the SOTA?** Since papers typically do not report on their experimental settings in detail, it is impossible to compare results on even the same dataset across different publications. Thus, we don't know what exactly is the state of the art right now. This again makes it difficult for all the three groups interested in ARA (mentioned in Section 1).

Amongst these, the first three limitations are of particular concern to NLP researchers, both in terms of using ARA in other NLP problems as well as furthering research on ARA itself. The remaining limitations are more general in nature, and would interest all the three target audience. We believe these are among what probably prevents the research in ARA from being accepted outside ARA.

5 Challenges and Open Questions

In view of the above mentioned limitations and their potential consequences, we identify four major challenge areas where more future work is needed.