Kittajafr-v1-1.0.0

Raoul Blin

Kittajafr-v1-1.0.0

R.Blin, cnrs-crlao, blin@ehess.fr

17 mars 2021

Résumé

Ce guide présente le kit kittajafr-v1. Il s'agit d'un ensemble de ressources et d'outils permettant de construire un petit modèle de traduction neuronal du japonais vers le français. Ce guide dresse un rapide état des lieux de la traduction ja > fr. Il explique comment préparer un corpus d'entraînement, paramétrer le logiciel d'apprentissage et de traduction. Toutes les ressources du kittajafr sont librement distribuées. Des logiciels non fournis dans le kit sont nécessaires mais eux aussi librement disponibles. Le corpus est de petite taille : ≈70K exemples alignés. La qualité des traductions est basse.

This guide presents KittajaFr-V1. It is a set of resources and tools to build a small model of neuronal translation from Japanese to French. This guide draws up a quick statement of the translation ja> fr. It explains how to prepare a training corpus, set the learning and translation software. All the resources of KittajaFR are freely distributed. Software not provided in the kit are needed but also freely available. The corpus is small : *approx* 70k examples aligned. The quality of translations is low.

Table des matières

1 Introduction

Le présent guide explique comment construire un modèle de traduction neuronal du japonais vers le français à l'aide d'un corpus de très petite taille et de logiciels idoines. Les ressources et outils cités sont tous librement distribués et sont mis à disposition dans le kit kittajafr.v1 . Quelques outils ne sont pas fournis mais sont librement disponibles sur l'Internet.

Le guide s'adresse à un lectorat qui dispose du matériel nécessaire pour entraîner des réseaux neuronaux (carte graphique nvidia) et qui peut manipuler des scripts basiques en Bash. Le lecteur peut utiliser les ressources et le guide sans connaissances préalables sur les réseaux neuronaux ni sur la méthodologie en traduction automatique statistique. Quelques informations sont fournies mais ne constituent pas une introduction à la traduction automatique neuronale. Le lecteur averti ne trouvera quant à lui pas d'informations nouvelles.

Le guide s'en tient aux informations strictement nécessaires sur le japonais et le français. Elles sont suffisantes pour qu'un utilisateur qui ne lit aucune de ces langues puisse malgré tout procéder aux manipulations.

2 Contexte

La «traduction automatique neuronale» s'est imposée à partir de 2015 environ comme la technique la plus performante pour faire de la traduction automatique. Aujourd'hui, les principaux acteurs industriels du domaine (Deepl, Google, Systran, etc. [1]) utilisent cette technologie. D'innombrables micro-projets existent par ailleurs. De surcroît, cette technique bénéficie d'un effort de recherche considérable, tant dans le monde industriel que dans le monde académique. On peut même avancer qu'il s'agit de la seule piste suivie aujourd'hui. Les progrès devraient donc se poursuivre.

La démarche pour construire un système de traduction «neuronal» est la même que pour les techniques statistiques utilisées jusqu'alors. Ce sont les algorithmes d'apprentissage qui changent en s'inspirant initialement des neurones et réseaux de neurones. Naïvement présenté, la méthode est la suivante. On rassemble un corpus de textes déjà traduits, et alignés. Par algorithmes on détermine les correspondances entre les composants dans les deux langues. C'est la phase dite «d'entraînement» ou «d'apprentissage». La description des correspondances constituent un «modèle de traduction». Le modèle est ensuite utilisé pour traduire.

La situation de la traduction japonais $\leftrightarrow$ français est très constrastée. D'un côté, les industriels obtiennent des traductions tout à fait exploitables pour se faire une idée du contenu des textes et en comprendre les tenants et les aboutissants. Un exemple est donné dans le tableau Tab.1 ainsi qu'une proposition de traduction manuelle. Il faut noter que le texte d'origine contient de nombreuses structures assez difficiles et qu'il n'est pas surprenant que le traducteur automatique se trompe. Par exemple, le début de la première phrase nécessite une reformulation en profondeur pour être correctement rendue dans un français fluide. De ce point de vue, on peut considérer que Deepl s'en sort plutôt bien. Google Translate a une réaction très inattendue par rapport au vocabulaire : alors qu'il rend correctement le nom du ministère malgré plusieurs écueils morphologiques et malgré la spécialisation du terme, il se trompe sur le terme «anti corruption» pourtant très évident.

Côté académique, la paire de langues ne fait l'objet d'aucune étude publiée. Le milieu académique n'a pas mis à disposition de systèmes de traduction (logiciel, modèle de traduction). A ce jour, aucune campagne d'évaluation n'a porté dessus. La raison est certainement l'absence de corpus

1. deepl.com, translate.google.com, www.systransoft.com/fr/lp/traduction-en-ligne, etc.)

谷脇康彦・前総務審議官ら同省幹部3人に計18万円を超える接待をしていたNTT。 15日の参院予算委員会に参考人として招致されたNTTの澤田純社長は、社員向けの「贈賄防止ハンドブック」との整合性を問われ、そう答えた。 (src : asahi.com, date : 2021/03/15)
NTT avait diverti trois cadres du ministère, dont Yasuhiko Taniwaki et l'ancien ministre des Affaires intérieures et des Communications, pour un total de plus de 180 000 yens. Le président de NTT, Jun Sawada, qui a été invité à la commission du budget de la Chambre des conseillers le 15 à titre de référence, a répondu à la question de la cohérence avec le «Manuel pour l'anti-brûlure» pour les employés. (Google Translate)
NTT a fourni un total de plus de 180 000 yens de divertissement à trois hauts fonctionnaires du ministère, dont Yasuhiko Taniwaki, ancien vice-ministre des affaires générales. Lorsque le président de NTT, Jun Sawada, a été invité en tant que témoin à la réunion de la commission du budget de la Chambre des conseillers le 15 mars, il a répondu à une question sur la cohérence du manuel anti-corruption destiné aux employés. (Deepl)
NTT a offert un total de plus de 180 000 yens en divertissement à trois cadres du Ministère des Affaires intérieures et Communication, dont l'ancien Directeur général adjoint Yasuhiko Taniwaki. Le président de NTT, Jun Sawada a été invité en tant que témoin à la réunion de la commission du budget de la Chambre des conseillers le 15 <mars>. Interrogé sur la cohérence <entre ces faits> et le «manuel anti-corruption» destiné aux employés, il a répondu ainsi. (une traduction possible)

Table 1 – Traduction d'extrait d'article de presse par deux logiciels commerciaux.

d'entraînement de taille suffisante, librement distribués. Aujourd'hui, seules 70K paires d'exemples de qualité satisfaisante sont librement accessibles (Blin [2018]). Voire une description des corpus dans la section 5.

Pour de la traduction par apprentissage, 70K bi-exemples est très insuffisant. A titre de comparaison, le japonais-anglais bénéficie d'un corpus libre de bonne qualité de plus de 700K segments (Blin [2020]). Les corpus libres pour les langues européennes se situent au delà des 10 millions. Chez les industriels, l'unité de mesure de la taille des corpus serait plutôt le milliard.

Des solutions existent pour pallier le manque de corpus : traduction multilingue (voir la proposition «fondatrice» de cette approche dans le cadre de la traduction neuronale : Melvin et al. [2016]), corpus synthétiques (Currey and Heafield [2019]), backtranslation, etc ... Mais même là, spécifiquement pour le japonais ↔ français, il n'existe présentement aucun dispositif librement accessible. Il faudrait adapter des dispositifs existants pour d'autres langues.

Le dispositif que nous décrivons ici, basé sur les ressources fournies dans le kittajafr, s'inscrit dans ce contexte de pénurie de corpus et se veut tout à fait standard. Nous n'aurons pas recours à ces techniques d'augmentation de corpus.

3 Vue d'ensemble sur les composants et la démarche

Les deux composants de base d'un système de traduction par apprentissage sont le logiciel de traduction et le corpus. De nombreux logiciels sont désormais tout en un, et clef en main : ils permettent de gérer la totalité du processus, de l'entraînement à la traduction. La tâche principale consiste à le paramétrer si la configuration par défaut ne suffit pas. Une fois rassemblé le corpus, celui-ci est traité par le système de traduction qui construit un «modèle de traduction». On utilise ce dernier pour traduire.

Reste l'aspect matériel. Aujourd'hui, la manipulation des réseaux de neurones nécessite d'utiliser des cartes graphiques. A moins de s'en tenir à des corpus ridicules qui malheureusement ne produiront rien d'intéressant, il ne faut pas espérer entraîner un réseau de neurones avec de simples CPU, quel que soit le nombre de coeurs. C'est peine perdue tant cela demanderait de temps (à compter en jours voire semaines). Pour effectuer la manipulation décrite ici, nous avons utilisé une carte graphique NVIDIA V100. C'est un luxe pour traiter notre petit corpus de 70K bi-exemples. L'entraînement dure deux heures. Une carte graphique K80 suffit et réalise le travail en quelques heures.

4 Logiciel de traitement

Un phénomène remarquable a accompagné le développement des techniques basées sur les réseaux de neurones. C'est la mise à disposition de très nombreux outils librement accessibles. Parmi les outils, nous avons choisi OpenNMT (Klein et al. [2017]). C'est un outil «tout en un» et «clef en main» qui prend en charge la construction d'un modèle de traduction, la traduction et l'évaluation. Il est utilisé dans les travaux universitaires, comme par exemple pour le japonais → anglais par Sekizawa et al. [2017] mais aussi dans l'industrie (Systran et Ubiqus sont des contributeurs importants).

Il existe plusieurs versions d'OpenNMT. Nous avons choisi OpenNMT-py (plutôt que OpenNMT-tf basé sur Tensorflow) parce qu'elle semblait la plus légère à faire tourner. Une documentation complète pour l'installer est disponible sur : opennmt.net/OpenNMT-py. Sur le même site se trouvent de nombreux tutoriels et des ressources (n'incluant ni le japonais ni le français). La manipulation décrite dans le présent guide a été réalisée avec la version 2.0.0 de OpenNMT-py. Le lecteur est supposé l'avoir installé avec succès.

5 Constitution du corpus

Plusieurs corpus sont nécessaires. On a besoin de deux corpus monolingues pour effectuer des traitements propres à chaque langue. C'est la partie la plus facile. Il suffit de récupérer sur le net un grand corpus de 500K à 1M de phrases pour chaque langue. Idéalement, on choisira des textes dans le domaine de spécialité visé, si il y en a un. Au pire, si aucun corpus monolingue n'est disponible, il est possible d'utiliser le corpus d'entrainement lui-même.

On a par ailleurs besoin d'un corpus bilingue aligné d'entraînement. C'est là que les difficultés se posent pour le japonais - français car comme évoqué précédemment, les corpus de bonne qualité sont rares. On compte environ 70K phrases en tout. Ces corpus sont réunis dans le kit kittajafr-v1. Les corpus de grande taille librement accessibles comme Wikimatrix (Schwenk et al. [2019]), MultiCCaligned (El-Kishky et al. [2020]) ou encore OpenSubtitles (Lison and Tiedemann [2016])

sont malheureusement très bruités [2] et nécessiteraient un prétraitement spécifique. Il est très possible que la partie réellement exploitable soit sensiblement réduite.

En plus de sa petite taille, une faiblesse majeure de ce corpus est qu'il n'est pas originellement en japonais. Une majorité d'exemples sont traduits (manuellement) de l'anglais vers le français et le japonais (voir par exemple le sous-corpus TED (Reimers and Gurevych [2020]). On parle alors de corpus synthétiques à partir de l'anglais comme langue pivot. Il y a quelques rares textes originellement en japonais ou écrits parallèlement en japonais et français. La traduction à partir de textes originellement rédigés en japonais peut souffrir de cette situation car le japonais «natif» peut contenir des structures peu représentées dans le corpus. La construction évoquée dans l'exemple étudié en introduction en est une.

Classiquement, le corpus bilingue («complet») est divisé en trois : un sous-corpus de «réglage fin» («tuning») d'au moins 500 exemples. Le chiffre le plus courant est 1000 exemples mais cela peut monter à 2000. Lorsque la traduction est spécialisée sur un domaine, ce corpus de réglage est choisi aussi proche que possible du domaine visé. Un second corpus «de test». Ce corpus n'est pas utilisé lors de l'entraînement. Il est utilisé pour évaluer la qualité des traductions. Tout le reste du corpus constitue le corpus d'entraînement proprement dit. Par défaut, on construit les corpus de test et de réglage en extrayant de façon aléatoire des bi-exemples à partir du corpus complet.

Pour effectuer des comparaisons avec des expériences antérieures (Blin [2020]), nous nous sommes aussi donné un corpus supplémentaire de test : PUD [3] . Une particularité est qu'il n'a rien à voir avec les textes du corpus d'entraînement. Cela permet d'évaluer les traductions «hors domaine», et de ce fait la capacité de notre système de traduction à généraliser. Comme les autres corpus, ce corpus supplémentaire a pour principal défaut de contenir des textes originellement rédigés en anglais.

6 Traitement du corpus

Le corpus est constitué de plusieurs bi-textes. Il est important de «mélanger» les bi-exemples. Cette opération peut être gérée par la commande «shuffle» d'OpenNMT. Pour chacune des langues, tous les corpus (entrainement, réglage et test) subissent un prétraitement identique. A noter que pour la majorité (voire la totalité) des systèmes de traduction, l'encodage géré est l'utf8. C'est bien sûr notre cas.

6.1 Japonais

Avant toute chose, l'encodage des caractères est uniformisé. En japonais, certains caractères sont encodés sur trois ou un octets. C'est le cas des chiffres arabes, des lettres romaines, et des signes de ponctuation. Nous uniformisons en encodant systématiquement les caractères sur un octet lorsque c'est possible. Par exemple le chiffre encodé sur trois octets [3] est transformé en [3], encodé sur un octet. Faute d'uniformisation de l'encodage, deux occurrences encodées différemment d'un même caractère seront considérées comme deux caractères différents. En plus, le lemmatiseur (MeCab) que nous utilisons traite différemment les caractères selon leur encodage. Deux occurrences d'un même chiffres risquent ainsi d'être traités différemment.

2. Peut contenir des phrases déjà mal traduites, ou qui ne correspondent que partiellement. On trouve aussi des phrases cible sans aucun rapport avec les phrases sources)

3. Corpus de test utilisé à CoNLL 2017 shared task on parsing Universal Dependencies. https ://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2184.

Le seul prétraitement impératif pour le japonais est la segmentation. En effet, le japonais ne sépare graphiquement pas les mots et le corpus non segmenté n'est pas exploitable en l'état par les outils de traduction existant. Il y a principalement trois types de segmentation : par caractères (chaque caractère est séparé par un espace), par «mots linguistiques» ou par unités déterminées selon des critères statistiques. La segmentation par caractères est évoquée dans la littérature mais n'est pas exploitée à grande échelle pour le japonais. Les deux autres segmentations sont fréquemment utilisées et peuvent chacune suffire à elles seules. L'intérêt de la segmentation statistique est qu'elle permet de contrôler la taille du vocabulaire.

La segmentation purement statistique, et plus particulièrement la segmentation avec SentencePiece, semble aujourd'hui suffire à donner les meilleurs résultats pour les traductions (Kudo and Richardson [2018]). Un avantage est qu'elle est indépendante de la langue traitée. Malgré tout, nous procédons à une double segmentation plus traditionnelle : linguistique puis statistique. Nous pensons en effet que pour remédier au manque de données d'entraînement, une solution serait d'injecter dans le texte des informations linguistiques. Pour cela, il faut disposer d'une segmentation basée sur des critères linguistiques. Néanmoins, la taille du vocabulaire joue sur la qualité les performances du système de traduction. Nous avons donc en plus recours à la segmentation statistique pour contrôler cette taille.

La segmentation linguistique est réalisée à l'aide d'un lemmatiseur (ou segmenteur). Pour le japonais, il en existe plusieurs très efficaces. Nous utilisons MeCab (Kudo [2006]). Cromieres et al. [2016] ont utilisé JUMAN par exemple.

Le choix du dictionnaire associé à l'analyseur joue sur le résultat de la traduction. Nous utilisons Unidic (Den et al. [2008]) pour plusieurs raisons. La première est qu'il permet une segmentation linguistiquement cohérente puisque les entrées lexicales sont toutes des morphes simples (non composés). Ces unités sont petites et permettent donc de réduire la taille du corpus, ce qui permet de réduire les temps d'entraînement. Ce sont par ailleurs des unités qui font relativement consensus dans la communauté linguistique et TAL.

On a observé par ailleurs que les traductions étaient meilleures qu'avec IPADIC (Asahara and Matsumoto [2003]). Ces observations sont valables pour nos petits corpus et le modèle de traduction que nous utilisons. Il est possible que dans d'autres configurations (très grand corpus, architecture Transformer), les conclusions soient différentes. Mais avec un système puissant, il est peut-être plus intéressant d'utiliser des unités linguistiques plus grandes telles que proposées par exemple par mecab-ipadic-neologd, qui comprend des morphes mais aussi des formes figées composées (Toshinori [2015]).

Pour segmenter morphologiquement les corpus, nous avons simplement utilisé la commande :

```
mecab F1 -O wakati -output F2 -d .../unidic
```

La segmentation statistique est appliquée au texte segmenté par MeCab (sur F2 donc). Elle est réalisée avec subword_nmt (Sennrich et al. [2016]). Comme l'utilisation du subword_nmt demande un travail préparatoire sur un corpus monolingue, nous décrivons son utilisation dans la section 9. Suivant les recommandations de Sennrich and Zhang [2019] qui se sont intéressé à l'optimisation des systèmes pour la traduction de corpus de petite taille [4], la taille du lexique est fixée à 12000 mots pour la langue source (le japonais).

Notre chaîne complète de traitement des textes japonais est donc :

uniformisation de l'encodage > segmentation linguistique > segmentation statistique

4. Les auteurs travaillent notamment sur la traduction coréen-anglais

6.2 Français

La morphologie du français constitue une difficulté supplémentaire pour l'apprentissage car les nombreux figements «cachent» les régularités. Par exemple, l'apostrophe fait qu'un même mot apparaît sous deux formes différentes *qu'* et *que*. Il existe des «tokenizers» qui standardisent les formes. Nous avons utilisé un outil standard : github.com/OpenNMT/Tokenizer.

En supplément, pour les langues occidentales, un traitement classique consiste à transformer les majuscules en minuscules. Nous avons là aussi utilisé un outil standard, Truecaser [5] . Comme son utilisation nécessite un travail préparatoire sur un corpus monolingue, nous le présentons à part dans la section 10.

Le corpus est ensuite segmenté statistiquement avec subword_nmt, en utilisant un dictionnaire de 8000 entrées. Il faut bien sûr entraîner Subword_NMT sur un corpus monolingue en français.

La chaîne complète de traitement du français est donc :

tokenisation > élimination des majuscules > segmentation statistique

7 Entrainement

Doté du corpus d'entraînement (train) et de réglage fin (val), l'entraînement peut commencer. La procédure se fait en fait en deux temps : constitution du lexique, puis entrainement proprement dit. Le tout est géré par OpenNMT. Il faut juste apporter quelques informations et paramétrer.

Pour construire le lexique, avec OpenNMT, il faut rédiger un fichier de configuration CONFIG au format yaml comme ci-dessous.

```
data:
    train_txt:
        path_src: .../train.ja
        path_tgt: .../train.fr
    valid:
        path_src: .../val.ja
        path_tgt: .../val.fr

save_data: ...

## "Where the vocab(s) will be written"
src_vocab: .../vocab/kittajafr-v1.vocab.ja
tgt_vocab: .../vocab/kittajafr-v1.vocab.fr
```

Puis lancer la procédure avec la commande ci-dessous. L'exécusion est rapide.

```
onmt_build_vocab -config .../CONFIG
```

Nous procédons ensuite à l'entraînement lui-même. Il faut décider des caractéristiques du réseau de neurones, les «hyperparamètres». Les effets de ces choix sont difficiles à prévoir avec précision. Il est fréquent de mener en parallèles plusieurs entraînements avec différentes valeurs pour les hyperparamètres, et de comparer leurs effets sur la qualité des traduction. Nous avons privilégié le paramétrage recommandé par Sennrich and Zhang [2019] pour les petits corpus.

5. https ://github.com/moses-smt/mosesdecoder/tree/master/scripts/recaser

Sennrich and Zhang [2019] montre lorsque les données sont peu nombreuses, contrairement à ce qui se fait souvent, il vaut mieux privilégier des petits modèles avec beaucoup de régularisation. Les réglages par défaut d'OpenNMT correspondent à des modèles assez petits mais ils peuvent encore être optimisés. L'encodeur par défaut est remplacé par un RNN bidirectionnel :

```
-encoder_type brnn
```

On utilise le «label-smoothing» :

```
-label_smoothing 0.1
```

L'algorithme d'entraînement adam est préféré à sgd proposé par défaut.

```
-optim adam
-learning_rate 0.0005
```

L'article recommande aussi des «tied embeddings», qui correspondent à l'option :

```
-share_decoder_embeddings
```

Le nombre de couches (layer) est fixé à 1.

La configuration de l'entraînement est notée dans un fichier yaml :

```
data:
    train_txt:
        path_src: $DCORPUS/train.$L1
        path_tgt: $DCORPUS/train.$L2
    valid:
        path_src: $DCORPUS/val.$L1
        path_tgt: $DCORPUS/val.$L2

src_vocab: $DVOCAB/vocab.$L1
tgt_vocab: $DVOCAB/vocab.$L2
save_model: $DMODEL/model
gpu_ranks: 0
save_checkpoint_steps: 10000
log_file: $DRESU/logtrain
encoder_type: brnn
label_smoothing: 0.1
optim: adam
learning_rate: 0.0005
src_word_vec_size: 512
tgt_word_vec_size: 512
enc_layers: 1
dec_layers: 1
enc_rnn_size: 1024
dec_rnn_size: 1024
```

L'entraînement dure environ 2 heures avec une carte NVIDIA V100, plusieurs heures avec une carte NVIDIA K80. OpenNMT crée plusieurs modèles, de plus en plus performants.

8 Evaluation

8.1 Principe

Pour dire les choses simplement, il n'y a pas de traduction automatique sans évaluation. L'évaluation humaine coutant trop chère, elle se fait automatiquement. Il existe de nombreuses méthodes d'évaluation mais aucune n'est complètement satisfaisante. Il n'est pas rare d'en combiner plusieurs. Comme la qualité de nos traductions est de façon très basse, une seule suffira.

La plus utilisée consiste à évaluer la similarité entre ce qui est produit par le système et la traduction par un humain. Le principe est de se doter d'un petit corpus (entre 500 et 2000 exemples, le plus souvent 1000 exemples) traduit par un humain dans la langue cible. Ce sont les traductions de référence (parfois en anglais «golden corpus»). On mesure ensuite la similarité entre les traductions par le système et ces traductions de référence. Cette mesure purement quantitative ne donne aucune indication sur la nature des erreurs de traduction.

Quoique pleine de défauts, cette évaluation est facile à mettre en oeuvre. Moyennant quelques précautions, elle permet de comparer les systèmes entre eux, ou même de comparer plusieurs configurations d'un même système. Les précautions sont les suivantes : la mesure de similarité se fait sur un très grand nombre d'exemples traduits (en général un millier). Pour comparer deux systèmes ou deux configurations d'un même système, le corpus d'évaluation et les langues doivent être les mêmes.

Il existe plusieurs mesures de similarité. La plus utilisée aujourd'hui en traduction automatique est l'incontournable score BLEU (Papineni et al. [2002]). La tendance est désormais à utiliser une seule et même implémentation de cette méthode. Celle qui s'impose est sacreBLEU (Post [2018]) car le logiciel propose de nombreuses fonctions, pour récupérer des tests internationaux, post-traiter etc. Nous nous en tenons à l'implémentation multi-BLEU [6] vu que le japonais→français n'a de toute façon fait l'objet d'aucune campagne d'évaluation internationale.

8.2 La pratique

Comme évoqué lors de la présentation du corpus, le kittajafr-v1 contient deux corpus d'évaluation. Un corpus «interne» est obtenu en extrayant quelques uns des exemples du corpus d'entraînement. Les exemples n'apparaissent donc plus dans le corpus d'entraînement. S'ajoute un corpus «externe», PUD, construit de façon complètement indépendante. Grâce à ces deux corpus, on peut évaluer la capacité du système à traduire des exemples «familiers», et à généraliser pour traduire des séquences complètement nouvelles.

Nous effectuons tout d'abord la traduction du corpus test. Comme pour l'entraînement, le paramétrage est important. L'avantage de la traduction comparé à l'entraînement, c'est que l'exécution est rapide. Il est donc possible de tester de multiples configurations. Dans tous les cas, on aura intérêt à choisir le dernier modèle produit (step=100000 dans notre cas). On peut jouer ensuite sur le maxlength, beamsize et batchsize. Il est possible qu'avec certains paramètres, un ou plusieurs segments ne soient pas du tout traduits.

La commande pour traduire est la suivante, avec les paramètres qui ont produit le meilleur résultat pour le corpus du kittajafr. A noter que la traduction a été effectuée avec une carte graphique mais lorsqu'on utilise un petit modèle comme ici, les CPU suffisent.

```
python translate.py  -max_length 100 -beam_size 1 -batch_size 30 \
```

6. http ://www.statmt.org/wmt06/shared-task/multi-bleu.perl

```
-model  .../model/model_step_100000.pt \
-src    .../corpus/test.ja \
-output .../test.resu.fr \
-gpu 0
```

On peut évaluer les traductions à différents stades de post-traitement. L'évaluation des traductions brutes permet de suivre au plus près le comportement du modèle. Comme ces traductions ne sont graphiquement pas naturelles (majuscules manquantes, présence de marques de segmentation etc), on effectue une seconde évaluation, après normalisation : reconstituer les formes non segmentées par bpe ($A@@\ B \rightarrow AB$), rétablissement des majuscules, etc.

Vient l'évaluation. Il suffit de comparer le fichier de sortie au fichier de référence à l'aide de multi-bleu :

```
perl multi-bleu.perl corpus_reference.fr < fich_traduction_a_evaluer.fr
```

8.3 Nos résultats

Les résultats obtenus avec notre dispositif sont sans surprise très bas...

Version	no.expé	PUD	interne
brute	1	2,98	12,02
	2	3,39	11,98
post-traité	1	1,55	7,91
	2	1,84	7,85

TABLE 2 – Scores BLEU des traductions ; en fonction du post traitement ; deux entraînements sur les mêmes données

L'entraînement et l'évaluation ont été effectués deux fois, sur les mêmes données. Les résultats diffèrent de quelques dizièmes de points. Ceci découle de l'initialisation aléatoire du réseau, qui aboutit à un état final un peu différent. Les variations peuvent aller jusqu'à au moins 5 dizièmes de différences.

La traduction du corpus PUD est de qualité moindre. Cela ne surprend pas puisque les structures et le vocabulaire sont peu présents dans le corpus d'entraînement. Quant au test «interne», il atteint un chiffre très en deça même de ce qu'obtient Sennrich and Zhang [2019] avec peu de corpus pour la traduction coréen $\rightarrow$ anglais (aux environs de 12 points BLEU). Comme indiqué précédemment, cette comparaison doit être prise avec une grande prudence vu que le corpus d'entraînement n'est pas le même et que les langues ne le sont pas non plus. Néanmoins, cela donne un ordre de grandeur. Il y a certainement une marge possible de progrès. Par exemple, l'examen du texte source prétraité montre que la normalisation des caractères est loin d'être satisfaisante. De même, du côté français, une analyse morphologique plus poussée pourrait aider le système à généraliser.

9 Segmentation statistique

Il existe plusieurs outils de segmentation statistique des textes. L'outil le plus en vue aujourd'hui est SentencePiece (Kudo and Richardson [2018]). Un de ses points forts est qu'il peut être appliqué à du texte brut non préalablement segmenté en mots.

Pour des raisons pratiques, nous avons utilisé la segmentation BPE (Byte Pair Encoding) avec le classique subword_nmt (Sennrich et al. [2016]) [7]. Dans les lignes qui suivent, nous supposons que l'installation a été déjà effectuée.

L'opération se déroule en deux temps : construction d'un modèle de segmentation sur un corpus monolingue, puis application de ce modèle pour segmenter le corpus de d'entraînement. Subword_nmt se présente comme un outil tout en un qui assure les deux tâches. Il est indépendant de la langue et s'utilise aussi bien pour le japonais que le français.

Pour chaque langue (japonais source et français cible), la procédure consiste à se doter d'un corpus monolingue et de lui appliquer le même prétraitement que celui appliqué au corpus d'entraînement. En l'occurrence uniformisation de l'encodage des caractères, tokenisation et/ou transformation des majuscules en minuscules. Sur le corpus monolingue prétraité, on lance l'apprentissage avec la commande ci-dessous, qui produit un dictionnaire. Ce dictionnaire sera utilisé pour la segmentation statistique du corpus d'entraînement.

```
subword-nmt learn-bpe -s 12000 --input corpus_pretraite --output dictionnaire
```

La commande «-s» permet de fixer la taille du dictionnaire. Conformément aux recommandations de Sennrich and Zhang [2019], la taille est fixée à 12000 pour la langue source (japonais) et 8000 pour la langue cible (français).

10 Conversion majuscules>minuscules

Il existe de nombreux outils de conversion des majuscules en minuscules. Nous utilisons un outil classique, universel (indépendant de la langue traitée) : truecaser. [8] La procédure se fait en deux temps : construction d'un modèle puis application. La procédure est décrite à l'adresse "www.statmt.org/moses/ ?n=Moses_supportTools#ntoc11

Plus précisément, pour nous, la démarche est la suivante. On construit le corpus monolingue français, d'un minimum de 500K exemples. Mais cela peut varier. Il suffit de reprendre le corpus utilisé pour la segmentation. Si aucun corpus monolingue n'est disponible, l'opération peut être effectuée sur le corpus d'entraînement français. Le corpus monolingue doit ensuite subir le même traitement que le corpus d'entraînement (tokenisation).

L'entraînement est lancé avec la commande suivante. Il produira en sortie le modèle MODEL :

```
train-truecaser.perl --model MODEL --corpus CORPUSMONOLINGUE
```

Ce modèle sera ensuite utilisé pour transformer les majuscules en minuscules dans le corpus d'entraînement (par exemple 'train.fr') et de test :

```
truecase.perl --model MODEL < train.fr > train.trucased.fr
```

11 Remerciements

Je remercie Fabien Cromières pour les nombreuses informations fournies, notamment sur le réglage des hyperparamètres.

7. https ://github.com/rsennrich/subword-nmt
8. https ://github.com/moses-smt/mosesdecoder/blob/master/scripts/recaser/truecase.perl

Références

Masayuki Asahara and Yuji Matsumoto. Ipadic user manual, 2003.

Raoul Blin. Automatic evaluation of alignments without using a gold-corpus - example with french-japanese aligned corpora. In Shirai Kiyoaki, editor, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, may 2018. European Language Resources Association (ELRA). ISBN 979-10-95546-24-5.

Raoul Blin. Traduire des corpus pour construire 'des modèles de traduction neuronaux : une solution pour toutes les langues peu dotées ? In Christophe Benzitoun, Chloé Braud, Laurine Huber, David Langlois, Slim Ouni, Sylvain Pogodalla, and Stéphane Schneider, editors, *6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Volume 2 : Traitement Automatique des Langues Naturelles*, pages 172–180, Nancy, France, 2020. ATALA. URL `https://hal.archives-ouvertes.fr/hal-02784765`.

Fabien Cromieres, Chenhui Chu, Toshiaki Nakazawa, and Sadao Kurohashi. Kyoto university participation to wat 2016. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*, pages 166–174, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL `https://www.aclweb.org/anthology/W16-4616`.

Anna Currey and Kenneth Heafield. Zero-resource neural machine translation with monolingual pivot data. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 99–107, Hong Kong, November 2019. Association for Computational Linguistics. doi : 10.18653/v1/D19-5610. URL `https://www.aclweb.org/anthology/D19-5610`.

Yasuharu Den, Junpei Nakamura, Toshinobu Ogiso, and Hideki Ogura. A proper approach to japanese morphological analysis : Dictionary, model, and evaluation. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May 2008. European Language Resources Association (ELRA). URL `http://www.lrec-conf.org/proceedings/lrec2008/pdf/258_paper.pdf`.

Ahmed El-Kishky, Vishrav Chaudhary, Francisco Guzmán, and Philipp Koehn. CCAligned : A massive collection of cross-lingual web-document pairs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, November 2020.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT : Open-Source Toolkit for Neural Machine Translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL `http://aclweb.org/anthology/P17-4012`.

Taku Kudo. Mecab : yet another part-of-speech and morphological analyzer., 2006. URL `http://mecab.sourceforge.net`.

Taku Kudo and John Richardson. Sentencepiece : A simple and language independent subword tokenizer and detokenizer for neural text processing. In Eduardo Blanco and Wei Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018 : System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*,

pages 66–71. Association for Computational Linguistics, 2018. doi : 10.18653/v1/d18-2012. URL https://doi.org/10.18653/v1/d18-2012.

Pierre Lison and Jörg Tiedemann. Opensubtitles2016 : Extracting large parallel corpora from movie and tv subtitles. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may 2016. European Language Resources Association (ELRA). ISBN 978-2-9517408-9-1.

Johnson Melvin, Schuster Mike, V. Le Quoc, Krikun Maxim, Wu Yonghui, Chen Zhifeng, Thorat Nikhil, B. Viégas Fernanda, Wattenberg Martin, Corrado Greg, Hughes Macduff, and Dean Jeffrey. Google's Multilingual Neural Machine Translation System : Enabling Zero-Shot Translation. *CoRR*, abs/1611.04558, 2016. URL http://arxiv.org/abs/1611.04558.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU : a Method for Automatic Evaluation of Machine Translation. In *Proc. ACL*, pages 311–318, 2002.

Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation : Research Papers*, pages 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-6319.

Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2020. URL https://arxiv.org/abs/2004.09813.

Holger Schwenk, Vishrav Chaudhary, Shuo Sun, Hongyu Gong, and Francisco Guzmán. Wikimatrix : Mining 135m parallel sentences in 1620 language pairs from wikipedia. *CoRR*, abs/1907.05791, 2019. URL http://arxiv.org/abs/1907.05791.

Yuuki Sekizawa, Tomoyuki Kajiwara, and Mamoru Komachi. Improving japanese-to-english neural machine translation by paraphrasing the target language. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*, pages 64–69, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing. URL https://www.aclweb.org/anthology/W17-5703.

Rico Sennrich and Biao Zhang. Revisiting low-resource neural machine translation : A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221, Florence, Italy, July 2019. Association for Computational Linguistics. doi : 10.18653/v1/P19-1021. URL https://www.aclweb.org/anthology/P19-1021.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 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 2016. Association for Computational Linguistics. doi : 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162.

Sato Toshinori. Neologism dictionary based on the language resources on the web for mecab, 2015. URL https://github.com/neologd/mecab-ipadic-neologd.