Automatic Bilingual Markup Transfer

Thomas Zenkel and **Joern Wuebker** and **John DeNero**
Lilt, Inc.
first_name@lilt.com

Abstract

We describe the task of bilingual markup transfer, which involves placing markup tags from a source sentence into a fixed target translation. This task arises in practice when a human translator generates the target translation without markup, and then the system infers the placement of markup tags. This task contrasts from previous work in which markup transfer is performed jointly with machine translation. We propose two novel metrics and evaluate several approaches based on unsupervised word alignments as well as a supervised neural sequence-to-sequence model. Our best approach achieves an average accuracy of 94.7% across six language pairs, indicating its potential usefulness for real-world localization tasks.

1 Introduction

Machine translation (MT) has two primary use cases: *fully automatic* MT and *assistance* for human translators. Fully automatic translation is used widely by consumers, while professional human translation with machine assistance remains the preferred method for translations that require a guarantee of publication quality. In both use cases, markup of particular spans of the source text that encodes formatting, hyperlinks, and other extralinguistic information must be transferred to corresponding spans of the target translation. Prior work on neural machine translation has focused on the problem of simultaneous translation and markup for the *fully automatic* use case (Hashimoto et al., 2019). This work describes approaches to the complementary problem for the *assistance* use case.

A common and effective workflow for professional translators is to first produce the text of a translation, then transfer markup into this text. This paper describes approaches to automating the second step in this workflow by automatically transferring source markup into a fixed reference translation. This fixed reference may not be preferred by a machine translation model, for example because it was written by a human, and therefore the correspondence between source and target may be challenging to infer. In this way, markup transfer is similar to word alignment, which is typically applied to authentic human translations rather than machine translations. Indeed, Hanneman and Dinu (2020) describe an algorithm for using word alignments to perform markup transfer.

This work contains three novel contributions:

- An improved algorithm for markup transfer via word alignments;
- A supervised approach to markup transfer, which benefits from word alignments;
- An evaluation methodology and two metrics for comparing approaches to bilingual markup transfer that can be applied to the structured document translation corpus released by Hashimoto et al. (2019).

In experiments across six language pairs, we find that neural word alignments increase markup transfer accuracy over FastAlign by 5.2% using prior markup transfer methods, our improved transfer algorithm increases accuracy by an additional 7.3%, and our supervised approach further increases accuracy by 9.9%. Our best approach has an average accuracy of 94.7%, compared to a baseline of 72.3% from applying the markup transfer algorithm of Hanneman and Dinu (2020) to word alignments from FastAlign (Dyer et al., 2013). This improved performance indicates potential usefulness in a professional localization setting. NLP practitioners may also benefit from this reliable method of transferring span annotations to new languages.

2 Related Work

Recent work in neural word alignment has indicated that markup transfer is a downstream task,

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 3524–3533
November 7–11, 2021.

though evaluations of word aligners have not included an explicit evaluation of markup transfer (Garg et al., 2019; Nagata et al., 2020; Jalili Sabet et al., 2020). Experiments in this paper are the first to quantify the amount by which the improved alignment quality of a neural aligner compared to FastAlign also improves markup transfer accuracy.

Markup can be represented using XML tags (Hashimoto et al., 2019). Previous work describes two approaches to markup transfer for fully automated machine translation, where the goal is to place each XML tag from the source into the target translation in a way that produces well-formed XML. The first approach is to include markup while training the translation model, such that the translation model takes as input a source sentence with XML markup and directly generates a translation that includes XML tags. A translation training set that includes markup can either be created by human translators (Hashimoto et al., 2019) or synthesized by adding markup to an existing unformatted bitext (Hanneman and Dinu, 2020). A translation model that generates both text and markup may prefer an output sequence for which the XML markup is invalid (e.g. there might be an opening tag that is not closed). This problem can be addressed through XML-constrained beam search (Hashimoto et al., 2019). This approach requires training data that contains XML markup.

The second approach is to train the translation model without markup, separately train a word aligner, and then transfer format using an inference pipeline. After the translation model has generated a text translation, the alignment model aligns the tokens of the source segment to the generated translation. Finally, a deterministic algorithm (labeled Min-Max in Section 4.2) transfers the markup from the source segment into the translation via the word alignments (Hanneman and Dinu, 2020). This approach does not require training data that contains XML markup.

Past work has not measured markup transfer accuracy directly, because when a system generates both a translation and its markup, the translation differs from the reference by more than just markup. Instead, automatic metrics such as XML accuracy check that all source tags appear in the target and are properly nested. XML-based BLEU splits the translation at every formatting tag both for the reference and the translation and calculates the BLEU score (Papineni et al., 2002) on the resulting subsegments (Hashimoto et al., 2019). Past work has also included manual evaluation of the transferred markup information (Müller, 2017; Hanneman and Dinu, 2020), since transfer accuracy could not be assessed directly. In contrast, our goal is to transfer markup directly into the reference translation. Evaluation of markup accuracy is therefore straightforward: a tag is placed correctly if it appears at the correct character position within the reference translation.

Figure 1: Two nested tag pairs that have similar tag positions. Tag pair 1 is the parent of tag pair 2.

3 Bilingual Markup Transfer

In this section we introduce tag pairs, the data structure with which we represent markup information, and define two evaluation metrics.

3.1 Definition

We represent all markup information as tag pairs. A tag pair contains an opening and a closing tag and spans all characters of the sentence between the character position associated with its opening tag and closing tag. When two tag pairs span the same characters, one encloses the other, as in Figure 1. To indicate nesting order, we say that the enclosed pair has the enclosing pair as its *parent*.

Below is a data structure to represent a tag pair:

```
class TagPair:
    opening_tag: Tag
    closing_tag: Tag
    parent: Optional[TagPair]

class Tag:
    position: int
    label: str
```

Each `position` describes the number of text characters that appear before the tag in the sentence, not including any other tags. In contrast to the opening tag, the `label` of a closing tag contains a forward slash (e.g. `</b>`). There are no self-closing tags in this representation. A `TagPair` has a `parent` if there is another `TagPair` that encloses it.

3.2 Metrics

The following two metrics[1] score a proposed set of tags that are well-formed XML (properly nested with each opening tag closed) in which every source tag pair appears exactly once in the target.

Let L be the character length of the reference translation. In the following we denote the character position of a tag as $p \in \{0, ..., L\}$. We start by matching the reference and hypothesis tags by their label. Therefore, let $\mathcal{T} = \{(p_r, p_h)\} \in \{0, ..., L\} \times \{0, ..., L\}$ be the set of tuples of all reference and hypothesis character-level positions, and let $|\mathcal{T}|$ be the number of tags.

To evaluate the quality of the automatically transferred markup tags, we compare the reference character-level position p_r of each tag in the target sentence with its position in the hypothesis p_h.[2] The tag accuracy metric is the fraction of correctly placed tags:

$$C_h = \{(p_r, p_h) \in \mathcal{T} | p_r = p_h\}$$

$$\text{tag accuracy} = \frac{|C_h|}{|\mathcal{T}|}$$

This metric is meant to reflect the human effort saved in the *assistance* use case, as each incorrectly placed tag must be corrected manually.

However, in some cases there may be multiple reasonable tag placements. An example for such a case is provided in Figure 2. Therefore, another useful metric for markup transfer accuracy is the average character distance between reference and hypothesis tag positions.[3]

$$\text{average distance} = \frac{\sum_{(p_r, p_h) \in \mathcal{T}} |p_r - p_h|}{|\mathcal{T}|}$$

Distinction between metrics: The two metrics are designed to evaluate different aspects of the tag placements.

Tag accuracy checks whether a tag is at exactly the same position as in the reference. The character distance uses the assumption that, if multiple tag placements are correct, the different correct tag placement will oftentimes be close to each other as in the example of Figure 2. Both metrics will yield a perfect score for reproducing the reference exactly, but for an incorrect placement the character distance gives additional information about the severity of the errors.

Figure 3 provides an example of this situation.

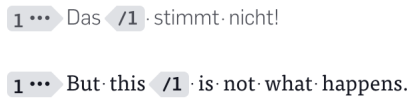

Figure 2: In the source sentence the German word "Das" is formatted. In the translation formatting either "But this" or "this" are both reasonable options.

4 Unsupervised Markup Transfer

For unsupervised markup transfer, we apply a two-step process. First we use an unsupervised aligner to infer the alignments between source and target subwords. The second step uses a deterministic algorithm to place tag pairs based on these alignments. Two advantages of this unsupervised approach are that it does not require training data with markup, and it can leverage any word aligner.

4.1 Alignments

An alignment expresses the token-level correspondence between a source sentence and its target translation. Tokens can be words, individual characters or subwords. Our experiments align subwords to minimize alignment error rate (Zenkel et al., 2020).

Let s_i and t_j represent the ith token in the source sentence and the jth token in its translation, respectively. The number of tokens of the source sentence and its translation are I and J. Additionally, let $A(s_i) \subseteq \{1, \ldots, J\}$ define the alignments of the ith source token to a set of target tokens.

In this work, we compare the popular FastAlign toolkit (Dyer et al., 2013), a statistical aligner, to a state-of-the-art neural alignment approach described by Zenkel et al. (2020) based on the Transformer architecture.

4.2 Min-Max Tag Pair Projection

As a baseline markup transfer algorithm we implement the approach described by Hanneman and Dinu (2020), which we call the Min-Max algorithm. Each tag pair in the source sentence spans multiple contiguous source tokens $s_{i'}, \ldots, s_{i''}$. To project the start and end tags of the tag pair into the translation, we use the union of the target alignments

[1] An implementation of these metrics is available at https://github.com/lilt/markup-tag-evaluation.

[2] In case of ambiguity due to multiple tags with the same label, each reference tag is matched with a unique hypothesis tag in a way that maximizes accuracy.

[3] In case there are multiple reference tags with the same label, for each reference tag we use the closest hypothesis tag with this label to calculate the average character difference.

Reference	But this is not what happens.	**tag accuracy**	**average distance**
Hypothesis 1	But this is not what happens.	50.0%	2
Hypothesis 2	But this is not what happens.	50.0%	10

Figure 3: A reference tag placement and two different hypotheses for the sentence "Das stimmt nicht!". While both hypotheses have a tag accuracy of 50%, the average distance of the first hypothesis (4/2=2) is lower than the average distance of the second one (20/2=10).

of its spanned source words $L = \bigcup_{i=i'}^{i''} A(s_i)$. We project the tag pair to the contiguous target span $t_{min(L)}, \ldots, t_{max(L)}$ that contains all target tokens present in the set of target alignments. This method implicitly maintains nesting order.

4.3 Inside-Outside Tag Pair Projection

The Min-Max approach has the disadvantage that a single incorrect alignment link can lead to a large error in the projected location of the target span. To address this shortcoming, we introduce the Inside-Outside span projection algorithm which is more resilient to spurious alignment links. It works by individually scoring all possible target spans and selecting the span with the highest score. For nested tag pairs, we ensure that nesting order is maintained by projecting the parent first, and restricting the search space of the child to the span of the projected parent pair.

The Min-Max algorithm can be viewed as a special case of this generalization, where the score for a target span is defined as the total number of alignment links between tokens in the source and the target spans, with a penalty for unaligned words at the boundaries.

The Inside-Outside span projection algorithm expands this idea by considering alignment links both inside the spans and outside of the spans. The score for each target span is defined as the total number of alignment links inside the source and target spans, plus the number of links outside of the spans. Formally, given a source span $s_{i'}, \ldots, s_{i''}$, the score for the target span $t_{j'}, \ldots, t_{j''}$ is calculated as $s(j', j'') = |L_{in}| + |L_{out}|$ with

$$L_{in} = \bigcup_{i \in \{i', \ldots, i''\}} \{j \in A(s_i) | j' \leq j \leq j''\}$$

$$L_{out} = \bigcup_{i \notin \{i', \ldots, i''\}} \{j \in A(s_i) | j < j' \vee j > j''\}$$

The highest scoring target span for a given tag pair can be computed in quadratic time by a straightforward application of dynamic programming.

4.4 Perfect Match Heuristic

During development of these algorithms we observed that markup tags often span source phrases that appear identically in the target (e.g. "start()", "DefaultWorkflowUser", "Identity Connect"). We define a tag pair as a perfect match if it spans a phrase in the source that appears exactly once in the target, and both the source and target phrase either span full words or both have a tag placed within words. The second condition is necessary to prevent perfect matches for cases like "We all" and "Wir alle". We project tag pairs that span perfect matches by placing the tag around the same phrase in the target segment.

5 Supervised Markup Transfer

When a bitext annotated with markup is available, it is possible to train a supervised markup transfer system. We implement a sequence-to-sequence model using the Transformer (Vaswani et al., 2017) architecture that learns to generate the target sequence with tags given input of the source with tags and the target *without* tags. To perform well in this task, the model must learn to copy the target text, infer the correspondence between source and target tokens, and place the tags present in the source text at corresponding positions in the target.

To encourage the model to learn the correspondence between source and target subwords, we pretrain it for machine translation, translating a source segment without tags into a target without tags. Afterwards, we train the model to project the markup tags into a given target sentence. The input of the model during this stage of training (and during inference) is the source segment with tags, a separator token, and then the target segment without tags. Figure 4 provides an input-output example.

After training we can project markup tags into the target sentence by searching for the most likely output sequence under the model, which will be a target sentence containing markup. We first consider greedy search. While a well-formed output results most of the time, the model does not always

Input	Select Multiple Languages ||| Wählen Sie Mehrere Sprachen aus
Output	Wählen Sie Mehrere Sprachen aus

Figure 4: Example input and desired output for a sequence-to-sequence supervised markup transfer system.

generate the same target sentence that appeared in the input. It also does not always reproduce all tags that appeared in the source segment.

To circumvent these issues, we can constrain the search towards a consistent output. During output sequence generation, we keep track of the text of the produced hypothesis, and constrain the next target token to be either a prefix of the remaining target text or a markup tag. When producing a markup tag, we make sure that only markup tags that appeared in the source segment can be opened, and we track their counts. To enforce a valid tag structure, we ensure that only the most recent opening tag without a corresponding closing tag can be closed. We additionally ensure that all tags appearing in the source are produced in the target exactly once. These constraints can be implemented efficiently using a bias vector that prevents invalid tokens by setting their bias to a large negative value. During every decoding step this bias vector is added to the logits before retrieving the most likely token.

During development of this model we noticed that the output of the unconstrained search provides a signal about its quality. If unconstrained greedy search does not copy the target text or does not reproduce all tags in a well-formed structure, typically the constrained search produces output with incorrect markup tag positions. Therefore, we evaluate an additional method which uses the output of unconstrained greedy search from the sequence-to-sequence model, but with a fallback to unsupervised markup transfer if either the text or tags of the output are inconsistent with the input—the two failure modes described above.

6 Experimental Setup

6.1 Dataset

We base our experiments on the multilingual dataset for structured document translation[4] described by Hashimoto et al. (2019). This dataset is extracted from the online help of an international enterprise software-as-a-service platform that is localized from English into multiple languages. The data is already aligned into segments consisting of one or multiple sentences. These segments contain markup tags that are always consistent between the source segment and its translation, that is the type and number of markup tags is the same across aligned segments.

The data set is split into a training set consisting of approximately 100k segments, a validation set of 2k segments and an unreleased test set. One fourth of the segments in both the training and validation set contain at least one markup tag. We hold out 1k segments of the training set for early stopping, use the remaining segments for training and the validation set for testing.

Only a fixed set of 14 different opening and closing markup tags appear in the dataset, each of these tag pairs spanning one or more characters.

6.2 Tokenization

We use byte pair encoding (BPE) (Sennrich et al., 2016) computed via the SentencePiece toolkit (Kudo, 2018), and follow the setup described by Hashimoto et al. (2019) for subword tokenization. We add all tags and the separator token used for the input of the sequence-to-sequence model as user-defined symbols. In contrast to Hashimoto et al. (2019), we also add all punctuation marks to this set. These symbols will not be split or merged by the SentencePiece toolkit and are always represented as a single token. We learn a joint subword vocabulary of 10k tokens for each language pair and use this tokenization for both the supervised sequence-to-sequence model and the unsupervised alignment systems. Zenkel et al. (2020) showed that subword-level alignment leads to lower alignment error rates than word-level alignment, both for statistical and neural aligners. For the purpose of markup tag transfer, subwords also provide more fine-grained information, for example if a markup tag is used to format a part of a word. Partial word formatting is common for German compound words, for example "<ph>Self-Service</ph>snutzung". We learn a single SentencePiece model on the concatenated training data including markup tags for both languages of each

[4] https://github.com/salesforce/localization-xml-mt

Method	EnDe	EnFi	EnFr	EnJa	EnNl	EnZh	Avg
FastAlign	75.9%	72.9%	81.9%	42.6%	83.9%	81.8%	72.3%
(Min-Max)	7.9	7.2	5.3	5.8	4.0	1.7	5.3
FastAlign	83.1%	80.0%	85.5%	47.1%	89.2%	83.8%	78.1%
(Inside-Outside)	2.7	3.8	1.8	3.2	1.2	1.1	2.3
FastAlign	86.7%	82.8%	88.9%	49.5%	91.2%	86.3%	80.9%
(Inside-Outside + Perfect Match)	2.4	3.3	1.6	3.0	1.0	1.0	2.1
NeuralAlign	77.4%	73.3%	83.8%	57.8%	84.1%	88.6%	77.5%
(Min-Max)	10.2	8.5	8.8	5.2	10.6	1.4	7.5
NeuralAlign	84.7%	81.2%	86.3%	64.5%	90.9%	91.4%	83.2%
(Inside-Outside)	2.3	2.5	3.4	1.3	1.4	0.5	1.9
NeuralAlign	88.2%	84.5%	87.5%	65.0%	92.0%	91.7%	84.8%
(Inside-Outside + Perfect Match)	1.9	2.3	3.4	1.2	1.4	0.5	1.8
Seq2Seq	89.6%	89.1%	91.0%	94.5%	89.0%	95.5%	91.5%
(Constrained Search)	15.8	13.4	16.1	3.0	15.7	0.7	10.8
Seq2Seq + NeuralAlign	91.6%	95.3%	95.2%	94.1%	95.6%	96.4%	94.7%
	2.0	1.3	2.1	0.8	1.1	0.2	1.3

Table 1: Tag accuracy and average distance results on the multilingual dataset for structured document translation. The methods above the double line are not trained using target markup; the methods below do use supervised data.

language pair.[5]

6.3 Unsupervised Markup Transfer: Alignment Systems

To compare unsupervised statistical and neural aligners, we strip all markup tags from the training and validation data and apply the SentencePiece model to obtain tokenized versions of the data.

As our statistical system, we use FastAlign (Dyer et al., 2013; Brown et al., 1993) due to its popularity. We concatenate both training and validation data and train the alignment system using its standard settings.

As our neural alignment system, we generate first-pass alignments and then train a guided alignment model using the generated alignments (Garg et al., 2019). To generate alignments for guided training, we follow Zenkel et al. (2020) and train an alignment layer on top of a Transformer-based machine translation system in the forward and backward direction. We then extract alignments using bidirectional attention optimization. We follow the hyperparameter settings of Zenkel et al. (2020): 6 encoder and 3 decoder layers with a layer dimension of 256. Finally, we train a guided alignment layer on top of the existing translation model in the forward direction. In contrast to Zenkel et al. (2020), we additionally shift the attention by one unit to the right using the "SHIFT-ATT" method described by Chen et al. (2020), which resulted in higher quality alignments. We finally generate attention distributions from the guided alignment layer and extract alignments based on the attention. To extract alignments, for each target token we select the source token with the highest attention value as its alignment link. This method, which is commonly used across neural alignment systems (Garg et al., 2019; Zenkel et al., 2019), does not produce any unaligned target tokens and produces more alignment links than FastAlign.

6.4 Supervised Markup Transfer: Sequence-to-Sequence Model

The sequence-to-sequence markup transfer model also has a transformer architecture with 6 encoder and 3 decoder layers using a embedding size of 256 and 8 attention heads per layer. We first train a translation model on the data with stripped markup tags. We then use this pretrained translation model and continue training to predict the target with tags using the input described in Section 5.

7 Evaluation

Table 1 shows accuracy and average distance results for all language pairs, discussed below.

[5]Scripts to reproduce this setup are available at https://github.com/lilt/markup-transfer-scripts.

7.1 Unsupervised Markup Transfer

All results labeled FastAlign or NeuralAlign are unsupervised in that they do not use the source or target markup in the corpus during model training.

7.1.1 Effect of Transfer Algorithms

Using FastAlign, the choice of markup transfer algorithm does impact tag accuracy. The simple Min-Max algorithm gives a tag accuracy of 72.3% and a character distance of 5.3, averaged across all language pairs. English to Chinese achieves the best average distance with 1.7 characters per tag, which is due in part to its segments containing fewer characters compared to target languages with phonetic alphabets. There is substantial variability in tag accuracy across language pairs, ranging from 42.6% (Japanese) to 83.9% (Dutch).

Compared to the Min-Max algorithm, the Inside-Outside algorithm improves both metrics in all cases. The tag accuracy improves by 5.8% and the character distance per tag reduces by half from 5.3 to 2.3, with the largest gains in German, Finnish, and Dutch. Figure 5 provides an example of a Min-Max projection error that is corrected by Inside-Outside. This example is typical in that a single incorrect link within the source span to a position in the target that is well outside the correct target span will cause a large error in the Min-Max algorithm, but will not cause a similar error for Inside-Outside.

7.1.2 Effect of Alignment Quality

When using the higher quality neural alignment system, the tag accuracy improves on average by 5% for both markup transfer algorithms. The character distance of the projected tags also decreases for the Inside-Outside algorithm, but increases when using the Min-Max algorithm. We speculate that the lack of null alignments in the neural alignment system makes it more likely that erroneous alignment links are off by a large distance, and so the Inside-Outside algorithm is particularly important for projecting markup with neural aligners.

7.1.3 Perfect Match Heuristic

To conclude the analysis of unsupervised markup transfer algorithms we analyse the rule-based transfer of markup tags that span "perfect matches". This simple heuristic increases the average tag accuracy consistently across all language pairs by 1.6% for the Inside-Outside algorithm. We analysed this result further for German, French and Chinese. The perfect match heuristic finds 236, 242 and 178 perfect matches for these three language pairs, respectively, and failed to match the reference tag in only eight cases across all three languages. These errors were largely due to the reference translation containing both the English and the translated word, e.g. "Clear (Effacer)", and the translator placing the tag around both words. In this case, the perfectMatch heuristic differed from the reference tag position by only spanning the English word "Clear".

	EnDe	EnFr	EnZh
Consistent	88.1%	87.8%	93.4%
Inconsistent Text	8.5%	9.2%	4.1%
Inconsistent Tags	6.7%	7.0%	3.5%

Table 2: Percentage of consistent segments produced by the sequence-to-sequence markup transfer model using unconstrained search and proportion of inconsistencies due to not being able to copy the text or not producing a consistent tag structure.

7.2 Supervised Markup Transfer

The supervised approach, Seq2Seq (constrained search), substantially outperforms the best unsupervised approach, increasing average accuracy by 6.7%. We analyse how often the sequence-to-sequence model correctly copies the provided target text and how often it produces a correctly formatted tag structure when using unconstrained greedy search. We focus on German and French as example phonetic languages and Chinese as an example character-based language. For German and French, greedy search produces a consistent output on 88% of the validation segments, and for Chinese on 93.4%. Failure to copy the target text is a slightly more frequent error mode compared to inconsistent tag structure (8.5% versus 6.7% for German). The two error modes are not mutually exclusive. Table 2 states the distribution of these errors for these three languages.

When using constrained search, we force the model to output the correct text and to copy all tags from the source segment. In comparison to unconstrained greedy search, this only changes the segments with inconsistencies and results in an overall tag accuracy of 89.6%, 89.1% and 95.5%, for German, French and Chinese. These results are consistently better than using the best unsupervised system, but the overall results are considerably lower compared to the subset of segments for which greedy search produced a consistent output.

Source	To see if your formula contains errors, click <u>Check Syntax</u>.
Min-Max	Klicken Sie auf <u>Syntax prüfen, um zu sehen, ob</u> die Formel Fehler enthält.
Inside-Outside	Klicken Sie auf <u>Syntax prüfen</u>, um zu sehen, ob die Formel Fehler enthält.

Figure 5: Example output of two markup transfer algorithms after FastAlign produced the wrong alignment link "Check"-"ob". While the Inside-Outside algorithm is able to recover and select the correct target span, the Min-Max algorithm erroneously selects an excessively large span. The tag <uicontrol> is abbreviated with <u>.

1.	**en**	To show the available values, leave the <uicontrol> Search for values...</uicontrol> box empty and click <uicontrol> Search </uicontrol>.
	ja	選択可能な値を表示するには、<uicontrol>[値を検索...]</uicontrol> ボックスを空のままにして、<uicontrol>[検索]</uicontrol> をクリックします
2.	**en**	Set the territory classification policy to <uicontrol>Highest</uicontrol>.
	ja	テリトリー分類ポリシーを [Highest (最高)] に設定します。
3.	**en**	Make the page the default object record page for specific <ph>Lightning apps</ph>.
	ja	ページを特定の <ph>Lightning アプリケーション</ph>のデフォルトのオブジェクトレコードページにする

Figure 6: Examples for the three patterns we identified in the English-Japanese test set that make word-alignment based tag transfer challenging. In example 3 the highlighted character sequence アプリケーションの constitutes a single subword in the tag-stripped sentence.

	EnDe	EnFr	EnZh
Consistent	98.6% 0.3	99.4% 0.1	98.0% 0.1
Inconsistent Text	53.5% 89.5	52.9% 99.3	78.6% 3.3
Inconsistent Tags	36.3% 116.4	47.7% 104.0	53.7% 10.0

Table 3: Tag accuracy and average distance using constrained search on subsets of segments based on whether unconstrained search produces consistent output. Note that in the "Consistent" case unconstrained and constrained search outputs are identical.

Table 3 summarizes the tag accuracy on different subsets defined by consistency behavior in unconstrained search. When greedy search correctly outputs the target with a consistent tag structure, its performance is close to perfect, achieving a tag accuracy above 98% and an average character distance below 0.3. When the text is inconsistent, the accuracy drops between 20% an 50% absolute. If the tags are inconsistent in the output of greedy search, constrained search places less than half of the tags correctly across the language pairs. The average distance increases to over 100.0 characters per tag for German and French. This large average difference is due in large part to tag pairs being placed at the very end of the target sentence.

7.3 Manual Error Analysis

On the English-Japanese data set there is a substantial gap in accuracy between the unsupervised and supervised approaches. A manual analysis identified three common patterns that make this task challenging for word-alignment based techniques.

1. Tags often span labels of UI elements like buttons, which in Japanese are additionally bracketed. These brackets do not have a correspondence in the English source.
2. Some label names are left untranslated, but with their Japanese translation in brackets.
3. Grammar particles at the end of Japanese words are usually not included in tags, but are not encoded as separate subwords when encoding the target sentence without tags, which makes correct placement through word alignment impossible.

Examples for these patterns are given in Figure 6.

7.4 Seq2Seq + NeuralAlign

Finally, we evaluate a simple approach to combining the output of the best unsupervised system with the output of the supervised system. When the greedy search of the sequence-to-sequence model produced a coherent output, we treat it as a signal that its output is of high quality. For these segments we use the output of the greedy search, otherwise

we use the output of the best unsupervised system. This approach, called Seq2Seq + NeuralAlign in Table 1, leads to both the best accuracy of 94.7% and average character distance of 1.3 character per tag, averaged across all language pairs. The performance gain over Seq2Seq for average distance is particularly large, indicating a substantial reduction in highly misplaced tags. Since the Seq2Seq system does not use word alignments, this improvement in performance is evidence that unsupervised word alignments are indeed useful for the task of bilingual markup transfer, even when supervised examples are available at training time.

8 Conclusion

We introduced the task of bilingual markup transfer into a fixed reference translation. Using two novel metrics, tag accuracy and average character distance, we evaluated both unsupervised and supervised approaches to this task. Both may be useful, depending on the availability of training examples with markup. Our supervised approach provides higher tag accuracy, but at the expense of higher average character distance. Combining supervised and unsupervised approaches corrects for this problematic behavior and provides a reliable and accurate method for markup transfer.

References

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.

Yun Chen, Yang Liu, Guanhua Chen, Xin Jiang, and Qun Liu. 2020. Accurate word alignment induction from neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 566–576, Online. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Sarthak Garg, Stephan Peitz, Udhyakumar Nallasamy, and Matthias Paulik. 2019. Jointly learning to align and translate with transformer models. 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 4453–4462, Hong Kong, China. Association for Computational Linguistics.

Greg Hanneman and Georgiana Dinu. 2020. How should markup tags be translated? In *Proceedings of the Fifth Conference on Machine Translation*, pages 1160–1173, Online. Association for Computational Linguistics.

Kazuma Hashimoto, Raffaella Buschiazzo, James Bradbury, Teresa Marshall, Richard Socher, and Caiming Xiong. 2019. A high-quality multilingual dataset for structured documentation translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 116–127, Florence, Italy. Association for Computational Linguistics.

Masoud Jalili Sabet, Philipp Dufter, François Yvon, and Hinrich Schütze. 2020. SimAlign: High quality word alignments without parallel training data using static and contextualized embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1627–1643, Online. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Mathias Müller. 2017. Treatment of markup in statistical machine translation. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 36–46, Copenhagen, Denmark. Association for Computational Linguistics.

Masaaki Nagata, Katsuki Chousa, and Masaaki Nishino. 2020. A supervised word alignment method based on cross-language span prediction using multilingual BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 555–565, Online. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Thomas Zenkel, Joern Wuebker, and John DeNero. 2019. Adding interpretable attention to neural translation models improves word alignment. *arXiv preprint arXiv:1901.11359*.

Thomas Zenkel, Joern Wuebker, and John DeNero. 2020. End-to-end neural word alignment outperforms GIZA++. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1605–1617, Online. Association for Computational Linguistics.