Abstract

Driven by encouraging results on a wide range of tasks, the field of NLP is experiencing an accelerated race to develop bigger language models. This race for bigger models has also underscored the need to continue the pursuit of practical distillation approaches that can leverage the knowledge acquired by these big models in a compute-efficient manner. Having this goal in mind, we build on recent work to propose a hallucination-free framework for sequence tagging that is especially suited for distillation. We show empirical results of new state-of-the-art performance across multiple sequence labelling datasets and validate the usefulness of this framework for distilling a large model in a few-shot learning scenario.

1 Introduction

Sequence labelling (SL) can be defined as the task of assigning a label to a span in the input text. Some examples of SL tasks are: i) named entity recognition (NER), where these labelled spans refer to people, places, or organizations, and ii) slot-filling, where these spans or slots of interest refer to attributes relevant to complete a user command, such as song name and playlist in a dialogue system. In general, these spans vary semantically depending on the domain of the task.

Despite the strong trend in NLP to explore the use of large language models (LLMs) there is still limited work evaluating prompting and decoding mechanisms for SL tasks. In this paper we propose and evaluate a new inference approach for SL that addresses two practical constraints:

- **Data scarcity:** The lack of vast amounts of annotated, and sometimes even the lack of unlabelled data, in the domain/language of interest.
- **Restricted computing resources at inference time:** LLMs are very effective, but deploying them to production-level environments is expensive, especially in contexts with latency constraints, such as in a live dialogue system.

Data scarcity leads us to consider high-performing encoder-decoder based LLMs. We address deployment concerns by considering distillation of such models into much smaller SL architectures, for instance Bi-Directional Long Short Memory (BiLSTM) (Hochreiter and Schmidhuber, 1997) units, through the use of both labelled and unlabelled data.

A standard distillation approach, knowledge distillation (KD) (Hinton et al., 2015), requires access to the probability that the teacher network assigns to each of the possible output tags. This probability distribution is typically unavailable at inference time for LLMs; thus, distillation of encoder-decoder models needs to resort to pseudo-labels: the student is trained on the one-hot labels that the teacher assigns to examples in an unlabelled dataset. This prevents the student model from learning those relationships among the probabilities of the incorrect classes that the teacher has learned. Similar arguments apply to decoder-only models.

In this paper, we propose SenT', a simple modification of the Simplified Inside Sentinel+Tag (SenT) format by Raman et al. (2022). We combine our target sequence format with a scoring mechanism for decoding, which we collectively call SenTScore. This combination results in an effective framework that allows us to employ a language model to perform sequence labelling and knowledge distillation. We show that SenTScore is an hallucination-free...
Table 1: An example of how an original input text (from the SNIPS dataset) is transformed into the SenT′ input for the model, and the format for the expected output. We use the explicit form of the special token strings used by T5. The addition of the extra token at the end of the input differentiates SenT′ from SentT. Notice the modified BIO scheme (sBIO) that we use in our experiments: a unique I tag is used for each of the output tags; so if the original tag set is $T$, the tags generated by the model are $T' \equiv T \cup \{I, O\}$.

Our proposed SenTScore method defines a sequence of scores over the output tags that can be aligned with those generated by the sequence tagging student network, making KD possible. We find an advantage in terms of performance in using KD as opposed to just pseudo-labels as a distillation objective, especially for smaller distillation datasets.

In sum, our contributions are:

- A new, hallucination-free, inference algorithm for sequence labelling with encoder-decoder (and possibly decoder only) transformer models, SenTScore, that achieves new state-of-the-art results on multiple English datasets.

- Empirical evidence showing an advantage of SenTScore when distilling into a smaller student model. This approach is particularly promising in the few-shot setting, which makes it even more appealing and practical.

2 Related work

Using LLMs to perform sequence tagging is discussed by Athiwaratkun et al. (2020); Yan et al. (2021); Paolini et al. (2021); Qin and Joty (2021); Xue et al. (2022) and Raman et al. (2022). While these previous works have minor differences in the prompting format of the models, all but the last one include input tokens as part of the target sequence. Different from our work, all previous models are prone to hallucinate.

Distillation refers to training a small student model from scratch using supervision from a large pretrained model (Bucilua et al., 2006; Hinton et al., 2015). Distillation of transformer-based models for different NLP tasks is typically discussed in the context of encoder-only models (e.g. Tang et al., 2019; Mukherjee and Hassan Awadallah, 2020; Jiao et al., 2020), with a few exceptions looking at distillation of decoder-only models (e.g. Artetxe et al., 2021).

In this paper we will discuss two approaches to distillation: pseudo-labels and knowledge distillation (KD). In the first approach the student model is trained on the hard labels generated by the teacher on some (unlabelled) dataset. In the second approach additional soft information provided by the teacher is used: typically the probability distribution the teacher assigns to the labels. In the context of sequence labelling, using pseudo-labels allows us to perform distillation on any teacher-student architecture pair. KD, on the other hand, requires access to the teacher’s probability distribution over the output tags. These are not usually available in language models for which the output distribution is over the whole vocabulary of tokens. We are not aware of other works which modify the decoder inference algorithm to generate such probabilities. However, there is recent work distilling internal representations of the teacher model, with the most closely related work to us being Mukherjee and Hassan Awadallah (2020). In that work the authors distill a multilingual encoder-only model into a BiLSTM architecture using a two-stage training process. This two-stage process, however, assumes a large unlabelled set for distilling internal model representations, embedding space, and teacher logits, and another significant amount of labelled data for directly training the student model using cross-entropy loss.

3 Datasets

We select seven English datasets that have been used in recent work on slot labelling: ATIS (Hemphill et al., 1990), SNIPS (Coucke et al., 2018), MIT corpora (Movie, MovieTrivia, and
Restaurant)\(^2\), and the English parts of mTOP (Li et al., 2021) and of mTOD (Schuster et al., 2019). Some statistics about the datasets are shown in Table 2. Some of these datasets (ATIS, SNIPS, mTOP, and mTOD) come from dialogue-related tasks, while the MIT ones have been used for NER.

We use the original training, development, and test sets of the SNIPS, mTOP, and mTOD datasets. For the ATIS dataset we use the splits established in the literature by Goo et al. (2018), in which a part of the original training set is used as the dev set. Similarly, we follow Raman et al. (2022)\(^3\) to obtain a dev set out of the original training set for each of the MIT datasets.

We notice that all datasets, with the exception of MovieTrivia, contain some duplicates. Among these, all apart from Restaurant contain examples in the test set that are also duplicated in the train and dev sets. This happens for fewer than 30 instances, with the exception of mTOD, where more than 20% of the test set examples are also found in the train and dev sets. How these duplicates are handled varies across the literature; we do not remove duplicates from the datasets used in our main results.

However, for mTOD, we also obtained results on a version of the dataset that was deduplicated as follows: If an example is duplicated, we retain it in the highest priority (defined below) split and removed from the others. To ensure the test set is as close as possible to the original test set, we order the splits in ascending order of priority as follows: test, dev, and train. We found that the F1 scores on the deduped mTOD dataset are within 0.5 points those on the original mTOD dataset across all experiments; as such, we do not report the deduped results in the following sections.

In addition to covering different domains, there are noticeable differences across the datasets in terms of the number of tags and the number of labelled examples for evaluation and testing, as can be seen in Table 2. This set of seven datasets allows us to gather robust empirical evidence for the proposed work that we present in what follows.

### 4 Score-based sequence labelling

Using LLMs for sequence tagging requires reframing the problem as a sequence-to-sequence task. In

<table>
<thead>
<tr>
<th>Datasets</th>
<th># tags</th>
<th># train</th>
<th># dev</th>
<th># test</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATIS</td>
<td>83</td>
<td>4478</td>
<td>500</td>
<td>893</td>
</tr>
<tr>
<td>SNIPS</td>
<td>39</td>
<td>13084</td>
<td>700</td>
<td>700</td>
</tr>
<tr>
<td>MovieTrivia</td>
<td>12</td>
<td>7005</td>
<td>811</td>
<td>1953</td>
</tr>
<tr>
<td>Movie</td>
<td>12</td>
<td>8722</td>
<td>1053</td>
<td>2443</td>
</tr>
<tr>
<td>Restaurant</td>
<td>8</td>
<td>6845</td>
<td>815</td>
<td>1521</td>
</tr>
<tr>
<td>mTOP (en)</td>
<td>75</td>
<td>15667</td>
<td>2235</td>
<td>4386</td>
</tr>
<tr>
<td>mTOD (en)</td>
<td>16</td>
<td>30521</td>
<td>4181</td>
<td>8621</td>
</tr>
</tbody>
</table>

Table 2: Number of examples per partition and number of unique tags in the SL datasets we used.

Raman et al. (2022), the strategy that proved the most effective, at least when applied to the mT5 encoder-decoder architecture, was the \textit{Simplified Inside Sentinel+Tag} (SenT in this paper). In this format (see Table 1), the original text is first tokenized according to some pretokenization strategy (whitespace splitting for all the datasets considered), and each of the tokens is prepended with one of the extra token strings provided by mT5 (the sentence tokens). The resulting concatenation is then tokenized using the mT5 tokenizer and fed to the encoder-decoder model. The output that the decoder is expected to generate is the same input sequence of special token strings, which are now alternated with the tags corresponding to the original tokens.

Given the set \( T \) of string labels to be used to annotate a span of text, the scheme used to associate tags across tokens is a modification of the standard BIO scheme: we use \( t \in T \) for any token that starts a labelled span, a single tag I for each token that continues a labelled span, and O to tag tokens that do not belong to labelled spans. We refer to this scheme as \textit{Simplified Inside BIO} (sBIO), and we indicate with \( \overline{T} \equiv T \cup \{I,O\} \) the tag set associated to it.

Raman et al. (2022) argue that the success of SenT can be attributed to two factors: 1) on the one hand, the use of sentinel tokens mimics the denoising objective that is used to pretrain mT5; 2) on the other hand, when compared to other decoding strategies, SenT does not require the decoder to copy parts of the input sentences and also produces shorter outputs. Both these facts supposedly make the task easier to learn and reduce the number of errors from the decoder (hallucinations, as they are often referred to in the literature).

We remark however that any output format among those described in the literature can be made completely free of hallucinations by constraining

\(^2\)The MIT datasets were downloaded from: https://groups.csail.mit.edu/sls/

\(^3\)Private communication with authors.
decoding (either greedy or beam search based) through a finite state machine enforcing the desired output format (see for instance De~Cao et al., 2020). In what follows we describe our proposed decoding approach that builds on this previous work.

4.1 SenTScore

Regardless of possible constraints imposed during generation, both SenT and the other algorithms described in Raman et al. (2022) use the decoder autoregressively at inference time to generate the target sequence. Since generation proceeds token by token and the textual representation of a tag is a variable length sequence of tokens, it is nontrivial to extract the scores and probabilities that the model assigns to individual tags.

We propose a different approach to inference, one in which the decoder is used to score sequences of tags. For this purpose, we consider a sequence tagging task with a label set $T$, and the associated sBIO tag set $\overline{T}$. Given an input sentence $S$, we use a pre-tokenizer (such as whitespace splitting) to turn $S$ into a sequence of token strings $x_1 \ldots x_L$, of size $L$. The SenT format is obtained by interleaving these tokens with special token strings to obtain the input string $S_{\text{in}} = s_0 x_1 s_1 \ldots x_L$. We use juxtaposition to indicate string concatenation. In what follows, we will work with SenT', a modification of SenT in which an additional special token is appended at the end, $S_{\text{in}} \leftarrow S_{\text{in}} s_L$. The reason for doing this will become clear in what follows.

The valid output strings that can be generated by the decoder are the $|T|^L$ sequences of the form $S_{\text{out}} = s_0 t_1 s_1 \ldots t_L s_L \in \mathcal{O}$ where $t \in \overline{T} \equiv T \cup \{ I, O \}$ consistent with the sBIO scheme convention. The encoder-decoder model can be used to calculate the log-likelihood of each of such strings $\log \mathcal{L}(S_{\text{out}}; S_{\text{in}})$, where $\theta$ represents the model parameters, and the best output will be:

$$S^*_{\text{out}} = \arg \max_{S \in \mathcal{O}} \log \mathcal{L}(S; S_{\text{in}})$$

Exact inference is infeasible but can be approximated using beam search as described in Algorithm 1. The outputs of the algorithm are the top-$K$ output strings and the score distribution associated with each of the output tags. As is evident from Table 1, it is simple to map back the final output string $S^*$ to the sequence of output tags and labelled spans.

At decoding time the output string is initialized with the first sentinel token $s_0$. At the $i$-th step, $t_i s_i$ with $t \in \overline{T}$,

\begin{equation}
(1)
\end{equation}

picks the highest scoring one, and keeps track of the score distribution. $s_i$ in Eq. 1, the next sentinel token, plays the crucial role of an EOS token at each step. This is needed to normalize the probability distribution: the likelihood of the string $s_0 t_1 \ldots s_{k-1} t_k'$ is always bounded by that of the string $s_0 t_1 \ldots s_{k-1} t_k'$ if $t$ is a prefix of $t'$, and we would never predict $t'$ as a continuation of $s_0 t_1 \ldots s_{k-1}$. This explains why we prefer using SenT' over SenT.

Finally, while SenTScore changes the inference algorithm, the finetuning objective we use throughout is still the original language modelling one.

4.2 Distillation

The main advantage of SenTScore is in the distillation setting. At each inference step, the algorithm assigns a likelihood to each sBIO tag. This distribution can be used to train the student network by aligning it to the teacher’s pre-softmax logits, in a standard knowledge distillation setup.

In detail, given an input sequence $S_{\text{in}}$, let $(y^i_t)_{i=1 \ldots L}$ be the sequence of sBIO output tags (as $|T|$-dimensional one-hot vectors) as inferred by the teacher model, and let $(u^i_t)_{i=1 \ldots L}$ (also $|T|$-dim. vectors) be the associated sequence of log-likelihoods. We indicate with $p^*_i$ the probability obtained by softmaxing $u^i_t$ and by $q_i$ the output of the softmax layer from the student. The contribution of each of the tags to the distillation objective

**Algorithm 1 SenTScore beam search**

\begin{algorithm}
\begin{algorithmic}
\Require Encoder-decoder parameters $\theta$, input $S_{\text{in}}$ with $L$ tokens, sBIO tag set $\overline{T}$, beam size $K$
\Ensure Approximate top-$K$ output sequences $B_{\text{out}}$ and their sBIO tag scores, $B_{\text{scores}}$
\State $B_{\text{out}} \leftarrow \{ s_0 \}_{i=1 \ldots K}$
\State $B_{\text{scores}} \leftarrow [[1]]_{i=1 \ldots K}$
\For{$i = 1$ to $L$}
\State $H \leftarrow \{ z t s_i \}_{z \in B_{\text{out}}, t \in \overline{T}}$ \Comment{Generate hypotheses}
\State $S \leftarrow \{ \log \mathcal{L}_\theta(h; S_{\text{in}}) \}_{h \in H}$ \Comment{Score hypotheses}
\State $\Pi \leftarrow \text{K-argsort} S$ \Comment{top-K args}
\State $B_{\text{out}} \leftarrow \text{take}(H; \Pi)$ \Comment{Update text beam}
\State $\hat{S} \leftarrow \text{reshape}(S; K, |\overline{T}|)$ \Comment{reshape scores}
\For{$k = 0$ to $K - 1$}
\State $k \leftarrow \Pi[k]$ \Comment{Update score beam}
\State $B_{\text{scores}}[k] \leftarrow \text{append}(B_{\text{scores}}[k]; S[k])$
\EndFor
\EndFor
\State \text{return} $B_{\text{out}}, B_{\text{scores}}$
\end{algorithmic}
\end{algorithm}

**SenTScore** uses the model likelihood to score each of the $|\overline{T}|$ possible continuations of the output sequence $t s_i$ with $t \in \overline{T}$, picks the highest scoring one, and keeps track of the score distribution. $s_i$ in Eq. 1, the next sentinel token, plays the crucial role of an EOS token at each step. This is needed to normalize the probability distribution: the likelihood of the string $s_0 t_1 \ldots s_{k-1} t_k'$ is always bounded by that of the string $s_0 t_1 \ldots s_{k-1} t_k'$ if $t$ is a prefix of $t'$, and we would never predict $t'$ as a continuation of $s_0 t_1 \ldots s_{k-1}$. This explains why we prefer using SenT' over SenT.

Finally, while SenTScore changes the inference algorithm, the finetuning objective we use throughout is still the original language modelling one.

4.2 Distillation

The main advantage of SenTScore is in the distillation setting. At each inference step, the algorithm assigns a likelihood to each sBIO tag. This distribution can be used to train the student network by aligning it to the teacher’s pre-softmax logits, in a standard knowledge distillation setup.

In detail, given an input sequence $S_{\text{in}}$, let $(y^i_t)_{i=1 \ldots L}$ be the sequence of sBIO output tags (as $|\overline{T}|$-dimensional one-hot vectors) as inferred by the teacher model, and let $(u^i_t)_{i=1 \ldots L}$ (also $|\overline{T}|$-dim. vectors) be the associated sequence of log-likelihoods. We indicate with $p^*_i$ the probability obtained by softmaxing $u^i_t$ and by $q_i$ the output of the softmax layer from the student. The contribution of each of the tags to the distillation objective

\begin{algorithm}
\begin{algorithmic}
\Require Encoder-decoder parameters $\theta$, input $S_{\text{in}}$ with $L$ tokens, sBIO tag set $\overline{T}$, beam size $K$
\Ensure Approximate top-$K$ output sequences $B_{\text{out}}$ and their sBIO tag scores, $B_{\text{scores}}$
\State $B_{\text{out}} \leftarrow \{ s_0 \}_{i=1 \ldots K}$
\State $B_{\text{scores}} \leftarrow [[1]]_{i=1 \ldots K}$
\For{$i = 1$ to $L$}
\State $H \leftarrow \{ z t s_i \}_{z \in B_{\text{out}}, t \in \overline{T}}$ \Comment{Generate hypotheses}
\State $S \leftarrow \{ \log \mathcal{L}_\theta(h; S_{\text{in}}) \}_{h \in H}$ \Comment{Score hypotheses}
\State $\Pi \leftarrow \text{K-argsort} S$ \Comment{top-K args}
\State $B_{\text{out}} \leftarrow \text{take}(H; \Pi)$ \Comment{Update text beam}
\State $\hat{S} \leftarrow \text{reshape}(S; K, |\overline{T}|)$ \Comment{reshape scores}
\For{$k = 0$ to $K - 1$}
\State $k \leftarrow \Pi[k]$ \Comment{Update score beam}
\State $B_{\text{scores}}[k] \leftarrow \text{append}(B_{\text{scores}}[k]; S[k])$
\EndFor
\EndFor
\State \text{return} $B_{\text{out}}, B_{\text{scores}}$
\end{algorithmic}
\end{algorithm}

**SenTScore** uses the model likelihood to score each of the $|\overline{T}|$ possible continuations of the output sequence $t s_i$ with $t \in \overline{T}$, picks the highest scoring one, and keeps track of the score distribution. $s_i$ in Eq. 1, the next sentinel token, plays the crucial role of an EOS token at each step. This is needed to normalize the probability distribution: the likelihood of the string $s_0 t_1 \ldots s_{k-1} t_k'$ is always bounded by that of the string $s_0 t_1 \ldots s_{k-1} t_k'$ if $t$ is a prefix of $t'$, and we would never predict $t'$ as a continuation of $s_0 t_1 \ldots s_{k-1}$. This explains why we prefer using SenT' over SenT.

Finally, while SenTScore changes the inference algorithm, the finetuning objective we use throughout is still the original language modelling one.

4.2 Distillation

The main advantage of SenTScore is in the distillation setting. At each inference step, the algorithm assigns a likelihood to each sBIO tag. This distribution can be used to train the student network by aligning it to the teacher’s pre-softmax logits, in a standard knowledge distillation setup.

In detail, given an input sequence $S_{\text{in}}$, let $(y^i_t)_{i=1 \ldots L}$ be the sequence of sBIO output tags (as $|\overline{T}|$-dimensional one-hot vectors) as inferred by the teacher model, and let $(u^i_t)_{i=1 \ldots L}$ (also $|\overline{T}|$-dim. vectors) be the associated sequence of log-likelihoods. We indicate with $p^*_i$ the probability obtained by softmaxing $u^i_t$ and by $q_i$ the output of the softmax layer from the student. The contribution of each of the tags to the distillation objective
that we use to train the student sequence tagger is
\[
- \sum_k \left( y^*_k \log (p^*_k) + \lambda KL(p^*_k||q_i) \right). \tag{2}
\]

The first term is the standard cross-entropy contribution from the pseudo-labels, while the second is the knowledge distillation term, implemented with a KL divergence with \( \lambda KL \) its associated positive weight.

We stress that we are allowed to write the second term only because \texttt{SentTScore} provides us with the tag scores. This is not the case for any of the formats proposed in Raman et al. (2022) or, as far as we know, elsewhere.\(^4\)

5 Experimental settings

We evaluate the models by computing the \texttt{F1} score on the test set of each dataset. \texttt{F1} is calculated following the CoNLL convention as defined by Tjong Kim Sang and De Meulder (2003), where an entity is considered correct iff the entity is predicted exactly as it appears in the gold data. We show micro-averaged \texttt{F1} scores.

The first set of experiments we performed are intended to investigate whether our proposed \texttt{SentTScore} approach is competitive with respect to recent results on the same datasets (Table 3). Our \texttt{SentTScore} model is a pretrained T5-base model (220M parameters) finetuned on each of the datasets.\(^5\) We trained each model for 20 epochs, with patience 5, learning rate of \(10^{-3}\), and batch size 32. We also want to know how the proposed framework compares against the following strong baselines:

- **BiLSTM**: Our first baseline is a BiLSTM tagger (Lample et al., 2016).\(^6\) The BiLSTM has a hidden dimension of size 200. Its input is the concatenation of 100d pretrained GloVE6B embeddings (Pennington et al., 2014) from StanfordNLP with the 50d hidden state of a custom character BiLSTM. We trained each model for 100 epochs, with patience 25, learning rate of \(10^{-3}\), and batch size 16.

- **BERT**: We finetune a pretrained BERT-base cased model (Devlin et al., 2019) (110M parameters) for the SL task and report results for each of the seven datasets. While we consider BERT a baseline model, we note that this pretrained architecture continues to show good performance across a wide range of NLP tasks, and for models in this size range BERT is still a reasonable choice. In preliminary experiments we compared results from the case and uncased versions of BERT and we found negligible differences. We decided to use the cased version for all experiments reported here. We trained each model for 30 epochs, with patience 10, learning rate of \(5 \times 10^{-5}\), and batch size 64.

\texttt{SentT\textsuperscript{*}}: The pretrained model is the same as that used for \texttt{SentTScore}. The goal of this baseline is to assess improvements attributed to our proposed decoding mechanism. This model is also the closest model to prior SOTA. The main difference between our results and those in Raman et al. (2022) is the pretrained model. They used a multilingual T5 model (Xue et al., 2021) with 580M parameters, whereas we use a smaller monolingual version (Raffel et al., 2020).

All the above models were trained with the AdamW optimizer (Loshchilov and Hutter, 2017). The best checkpoint for each training job was selected based on highest micro-F1 score on the validation set. All pretrained transformer models are downloaded from Huggingface.

5.1 Distillation experiment

We apply \texttt{SentTScore} and the loss function described in Section 4.2, to distill a finetuned T5 model into a BiLSTM architecture to perform sequence tagging. To mimic a low-resource setting, we randomly downsample the train/dev splits of all the datasets. We consider two sets of sizes for these gold train/dev splits: a 100/50 split and a 300/150 one. In both settings the remainder of the original training set is used for the distillation component using pseudo-labels.

We then finetune T5 using the \texttt{SentT\textsuperscript{*}} format on each of these two gold splits. The resulting model is used as the teacher in a distillation setting in which the student is a BiLSTM. The BiLSTM student is trained on the full training set by using the downsampled gold labels, but pseudo-labels and scores generated by the T5 teacher using \texttt{SentTScore} with \(K = 1\) in the rest of the training data. We use a temperature parameter \(\tau\) to rescale the distribution \texttt{SentTScore} defines over \(\mathbb{T}\). We use \(\tau = 10\) in all the distillation experiments.

\[\texttt{SentTScore}(T) = \frac{e^{\tau \texttt{SentTScore}(T)}}{\sum_T e^{\tau \texttt{SentTScore}(T)}}\]

\[\texttt{SentTScore}(T|\mathbb{L}) = \frac{e^{\tau \texttt{SentTScore}(T|\mathbb{L})}}{\sum_T e^{\tau \texttt{SentTScore}(T|\mathbb{L})}}\]

\[\texttt{SentTScore}(T|\mathbb{L}, \mathbb{S}) = \frac{e^{\tau \texttt{SentTScore}(T|\mathbb{L}, \mathbb{S})}}{\sum_T e^{\tau \texttt{SentTScore}(T|\mathbb{L}, \mathbb{S})}}\]

\[\texttt{SentTScore}(T|\mathbb{L}, \mathbb{S}, \mathbb{D}) = \frac{e^{\tau \texttt{SentTScore}(T|\mathbb{L}, \mathbb{S}, \mathbb{D})}}{\sum_T e^{\tau \texttt{SentTScore}(T|\mathbb{L}, \mathbb{S}, \mathbb{D})}}\]

\[\texttt{SentTScore}(T|\mathbb{L}, \mathbb{S}, \mathbb{D}, \mathbb{P}) = \frac{e^{\tau \texttt{SentTScore}(T|\mathbb{L}, \mathbb{S}, \mathbb{D}, \mathbb{P})}}{\sum_T e^{\tau \texttt{SentTScore}(T|\mathbb{L}, \mathbb{S}, \mathbb{D}, \mathbb{P})}}\]
The training schedule we follow is the same we use to train the BiLSTM baseline model, with the only exception that the best checkpoint is selected on the reduced dev set.

6 Results

The comparisons between baselines, SenT', and SentScore are shown in Table 3. SentScore is used with a $K=1$ beam size. Larger beams result in very similar performance and a considerable slowdown of inference time. SentScore consistently outperforms SenT' with constrained decoding, and all other baselines. Our intuition is that one advantage of SentScore comes from the fact that decoding happens tag-wise as opposed to token-wise (as in pure beam search). The last column of Table 3 shows the performance of the SenT implementation of Raman et al. (2022). Perfect scores are also reported for completeness. They are evaluated at the sentence level and correspond to the fraction of perfectly predicted examples. However, these results are not directly comparable: Raman et al. (2022) use a different and larger model (mT5-base with 580M parameters) and different optimization details. Nevertheless SentScore achieves better performance in a majority of cases.

6.1 Distillation results

Tables 4a and 4b show the result of the distillation experiments with 100/50 and 300/150 train/dev gold splits, respectively. While a BiLSTM tagger trained on the gold data significantly underperforms a finetuned T5-base model, once the BiLSTM is distilled on the silver data generated using SentScore, it outperforms even the original teacher model. We notice that the difference between student and teacher decreases for larger gold set size, suggesting that the effect is related to regularization properties of the distillation process. A similar phenomenon has been observed elsewhere, for instance in Furlanello et al. (2018) albeit with teacher and student sharing the same architecture.

In order to isolate the benefits of training the teacher model using KD as opposed to just pseudo-labels, we perform a set of ablation studies. For each dataset, we distill a BiLSTM student on a training set $\mathcal{T} = \mathcal{G}\cup\mathcal{S}$, where $\mathcal{G}$ is the original gold set and $\mathcal{S}$ is a random sample from the complement of $\mathcal{G}$. We choose $|\mathcal{S}| = 0, 250, 500$. The student is distilled using Eq. 2 with two choices of the loss.
The results also show that for our choice of teacher and student architectures, and datasets, the gap between KD and pseudo-labels is reduced when more pseudo-labeled data are used. Figure 1 further explores the relationship between amount of pseudo-labeled data and the scores obtained by varying all the random seeds that appear at training time: BiLSTM weight initialization, batch scheduling, and the choice of the silver data set.

Table 5: Distillation experiments with varying silver dataset size and ablation of the KD term in Eq. 2. The gold data split is the same as in Table 4a, with train/dev sizes of 100/50. The numbers in parentheses represent the standard deviation of the scores obtained by varying all the random seeds that appear at training time: BiLSTM weight initialization, batch scheduling, and the choice of the silver data set.

<table>
<thead>
<tr>
<th>Dataset - F1</th>
<th>No silver</th>
<th>250 silver</th>
<th>500 silver</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\lambda_{KL} = 0$</td>
<td>$\lambda_{KL} = 1$</td>
<td>$\lambda_{KL} = 0$</td>
</tr>
<tr>
<td></td>
<td>$\lambda_{KL} = 0$</td>
<td>$\lambda_{KL} = 1$</td>
<td>$\lambda_{KL} = 0$</td>
</tr>
<tr>
<td>ATIS</td>
<td>79.93 (0.85)</td>
<td>82.35 (0.44)</td>
<td>83.09 (1.49)</td>
</tr>
<tr>
<td>SNIPS</td>
<td>51.63 (1.25)</td>
<td>55.63 (1.38)</td>
<td>54.34 (0.71)</td>
</tr>
<tr>
<td>MovieTrivia</td>
<td>48.26 (0.95)</td>
<td><strong>51.86</strong> (1.09)</td>
<td>53.11 (1.26)</td>
</tr>
<tr>
<td>Movie</td>
<td>60.82 (0.67)</td>
<td><strong>64.20</strong> (1.07)</td>
<td>67.04 (0.66)</td>
</tr>
<tr>
<td>Restaurant</td>
<td>47.26 (0.83)</td>
<td><strong>50.19</strong> (0.81)</td>
<td>54.24 (1.17)</td>
</tr>
<tr>
<td>mTOP (en)</td>
<td>43.12 (1.84)</td>
<td><strong>46.43</strong> (1.01)</td>
<td>46.68 (2.31)</td>
</tr>
<tr>
<td>mTOP (en)</td>
<td>68.68 (2.68)</td>
<td><strong>70.40</strong> (0.97)</td>
<td>76.12 (1.07)</td>
</tr>
</tbody>
</table>

Table 6: Distillation experiments with varying silver dataset size and ablation of the KD term in Eq. 2. The gold data split is the same as in Table 4a, with train/dev sizes given by 300/150. All experimental details are common with Table 5.

multipliers: $\lambda_{KL} = 1$ and $\lambda_{KL} = 0$. The first setting is the same used in Tables 4a and 4b, while the second drops the KD loss and only keeps the pseudo-labels for distillation. Whenever pseudo-labels and scores are used, they are generated by the SenTScore algorithm.

The results are shown in Tables 5 and 6. We see a consistent trend in which KD outperforms training the student using only pseudo-labels. This in particular motivates SenTScore as an inference algorithm. The results also show that for our choice of teacher and student architectures, and datasets, the gap between KD and pseudo-labels is reduced when more silver data are used. Figure 1 further explores the relationship between amount of pseudo-labeled data and gains from KD with $|S| = 0, 250, 500, 2000$. The trend with more pseudo-labeled data remains unchanged.

7 Limitations and future work

A reasonable critique to our focus on real-world constraints is the simple fact the datasets we are using are not real-world ones. From noise to tokenization choices, many issues arise when considering datasets outside of the academic domain. However, we believe our methods are simple enough to be applicable to real-world scenarios and our results to be independent of these various subtleties.

Some issues that could be addressed in future work have to do with the exploration of even larger models and different architectures such as decoder-only ones (Radford et al., 2018, 2019; Brown et al., 2020; Zhang et al., 2022; Chowdhery et al., 2022; Black et al., 2021). We note, however, that in all our experiments we finetune all the weights of the pretrained models we use. When using extremely large models this becomes impractical. Recent work (Turc et al., 2019) suggests that KD with compact encoder-only student models, such as BERT, is a promising avenue for further research. Exploring the pure few-shot scenario, or only finetuning a subnetwork, for instance by using adapters à la Houlsby et al., 2019, would be also interesting.

8 Conclusion

Real-time systems need to find a trade-off between performances and computing resources, the latter constraint coming either from budget or some other service requirement. Such trade-offs become particularly evident with large pretrained transformer...
Figure 1: A graphical representation of the distillation results in Table 4a (100/50 gold train/dev split) as a function of the size of the silver dataset. Knowledge distillation using SenTScore generated scores outperforms pseudo-labels.
models, which achieve SOTA results on many NLP tasks at the cost of being extremely hard and expensive to deploy in a real-world setting.

The standard solution for this is distillation. In this paper we have revisited these issues for the SL task, which is often the first crucial step in many real-world NLP pipelines. We propose a new inference algorithm, SenTScore, that allows us to leverage the performance of arbitrarily large encoder-decoder transformer architectures by distilling them into simpler sequence taggers using KD as opposed to just pseudo-labelling.

Ethical considerations

The intended use of our proposed approach is related to sequence labelling tasks where there are latency constraints and limited labelled data available. While it is not impossible to identify potential misuses of this technology, it is not immediately clear what those malicious uses would be. On the contrary, this paper contributes to the body of work investigating efficient solutions for deployment of live systems.

Computing infrastructure and computational budget

All of our experiments were run on single V100 GPU machines with 32GB. The most expensive experiments relate to finetuning a model, including best checkpoint selection. In this case, the running time is directly related to the dataset size. For the experiments using the full train/dev set, running time varies from 45 minutes (mATIS corpus) to a few hours (mTOD corpus) for a T5-base model. Training a model takes, on average, around 4 iterations per second with batch size 32. For the generation of pseudo-labels, we did not implement batch processing and it takes around 0.15 seconds to annotate each sample.

References


Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata.


Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.


Huggingface. Models - Hugging Face.


Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages.


StanfordNLP. GloVe: Global Vectors for Word Representation.


