NER	Dev. F_1	Test F_1
BERT	95.14	90.76
NN	94.48	89.94
POS	**Dev. Acc.**	**Test Acc.**
BERT	97.56	97.91
NN	97.33	97.64
U-POS	**Dev. Acc.**	**Test Acc.**
BERT	98.34	98.62
NN	98.08	98.36

Table 1: Performance of fine-tuned BERT and nearest-neighbor based labeling (NN) on NER, POS tagging, and universal POS tagging; see text. BERT numbers are from fine-tuning the `huggingface` implementation, and differ slightly from Devlin et al. (2018).

CoNLL $\rightarrow$ Onto NER	Dev. F_1	Test F_1
BERT	58.41	58.05
NN	62.17	62.33
NN (no FT)	54.29	55.35
U-POS $\rightarrow$ POS	**Dev. Acc.**	**Test Acc.**
BERT	61.78	59.86
NN	96.70	96.98
NN (no FT)	87.44	87.13
Chunk $\rightarrow$ NER	**Dev. F_1**	**Test F_1**
BERT	9.55	8.03
NN	78.05	71.74
NN (no FT)	75.21	67.19

Table 2: Zero-shot performance of models trained on CoNLL NER and applied to fine-grained OntoNotes NER, with universal POS tags and applied to standard POS tagging, and on CoNLL chunking and applied to CoNLL NER. "NN (no FT)" indicates BERT was not fine tuned even on the original task.

tuned BERT, often significantly. In particular, we note that when going from universal POS tags to standard POS tags, the fine-tuned label-agnostic model manages to outperform the standard most-frequent-tag-per-word baseline, which itself obtains slightly less than 92% accuracy. The most dramatic increase in performance, of course, occurs on the Chunking to NER task, where the label-agnostic model is successfully able to use chunking-based training information in copying labels, whereas the parametric fine-tuned BERT model can at best attempt to map NP-chunks to PERSON labels (the most frequent named entity in the dataset).

In order to check that the increase in performance is not due only to the BERT representations themselves, Table 2 also shows the results of nearest neighbor based prediction *without* fine-tuning ("NN (no FT)" in the table) on any task. In all cases, this leads to a decrease in performance.

6 Encouraging Contiguous Copies

Although we model token-level label copying, at test time each $\hat{y}_t$ is predicted by selecting the label type with highest marginal probability, without any attempt to ensure that the resulting sequence $\hat{y}$ resembles one or a few of the labeled neighbors $y'^{(m)}$. In this section we therefore consider a decoding approach that allows for controlling the trade-off between prediction confidence and minimizing the number of *distinct* segments in $\hat{y}$ that represent direct (segment-level) copies from some neighbor, in the hope that having fewer distinct copied segments in our predictions might make them more interpretable or accurate. We emphasize that the following decoding approach is in fact applicable even to standard sequence labeling models (i.e., non-nearest-neighbor based models), as long as neighbors can be retrieved at test time.

To begin with a simple case, suppose we already know the true labels y for a sequence x, and are simply interested in being able to reconstruct y by concatenating as few segments $y'_{i:j}$ that appear in some $y'^{(m)} \in \mathcal{D}$ as possible. More precisely, define the set $\mathcal{Z}_{\mathcal{D}}$ to contain all the unique label *type* sequences appearing as a subsequence of some sequence $y'^{(m)} \in \mathcal{D}$. Then, if we're willing to tolerate some errors in reconstructing y, we can use a dynamic program to minimize the number of mislabelings in our now "prediction" $\hat{y}$, plus the number of distinct segments used in forming $\hat{y}$ multiplied by a constant c, as follows:

$$J(t) = \min_{\substack{1 \leq k \leq t \\ z \in \mathcal{Z}_{\mathcal{D}} : |z| = k}} J(t-k) + c + \sum_{j=1}^{k} \mathbf{1}[y_{t-k+j} \neq z_j],$$

where $J(0) = 0$ is the base case and $|z|$ is the length of sequence z. Note that greedily selecting sequences that minimize mislabelings may result in using more segments, and thus a higher J.

In the case where we do not already know y, but wish to predict it, we might consider a modification of the above, which tries to minimize c times