A "Will it Unblend?": Supplementary

A.1 Annotation

When annotating the PAXOBS schema, the `A` base is defined as the one whose exclusive material precedes all other exclusive materials in the blend, and from then on iteratively through the alphabet.

A.2 Segmentation Experiment

Character Tagger. We manually annotated 550 of the 1,579 blends in the (Gangal et al., 2017) dataset, and passed the rest through a heuristic program to cover whatever linear blends remained. The program flagged 150 blends as suspected non-linear, so we manually annotated them as well. A tagger trained on only the 550-blend set originally annotated did not reach better F1 scores than the one reported in the main text, trained on the full set.

We use a 3-layer Bidirectional LSTM followed by a 4-layer MLP (tuned in the 2–4 range on dev) with ReLU activation (tuned vs. tanh), trained for 30 epochs with early stopping, optimized using Adam (Kingma and Ba, 2014) with learning rate 0.01 and a batch size of 96 (not tuned). The character embedding dimension is 200 and the hidden dimensions for the LSTM and MLP are both 192 (not tuned). Tagging accuracy on the dev set is .457 and on our dataset .462. We translate the resulting PAXOBS tags into segmentations by segmenting on each label change (so ⟨"shoptics", `AAXXBBBS`⟩ becomes "*sh*;*op*;*tic*;*s*"). The model is implemented in PyTorch (Paszke et al., 2019).

Domain models. The vocabulary size of BPE and the Unigram LM are by default automatically inferred, but may be specified. For BPE, we tested several vocabulary sizes: 20,000, 30,000, 30,522 (WP vocabulary size) and 40,000, selected heuristically, and 30,522 performed best. Consequently, we used the same size for Unigram LM.

Results. Strict exact match scores are: All-chars, 0.0%; Seq. tagger, 3.5%; WordPiece, 7.7%; Domain BPE, 10.6%; Domain ULM, 6.3%. This ordering is consistent with the lenient exact match scores reported in Table 2.

A.3 Recovery Experiment Details

A single hyperparameter controlling the minimum length of candidate overlap in linear blends was set to 3 with no tuning. Candidates were selected according to bases' stemmed form. We provide the complete candidate lists in the project repository.

Character RNN. We use 2-layer GRUs with embedding dimension 128 and hidden dimension 256 (all chosen manually with no tuning), and a sample of ∼109,000 documents from the Westbury corpus (Shaoul, 2010), and optimize using Adam with early stopping determined by performance on a held-out development set of 10,000 randomly sampled documents. The model is implemented in PyTorch.

FastText. We used the Engish CommonCrawl 300-dimension vectors available from `https://fasttext.cc`, inferring OOV words using FastText software.

GloVe. We used 300-dimensional GloVe vectors trained on the 840-billion CommonCrawl corpus, obtained from `https://nlp.stanford.edu/projects/glove/`.

BERT. In candidate pair ranking, when multiple candidate pairs have the same initial wordpieces ⟨ l_0=*pre*, r_0=*suf* ⟩, we create a new sentence input "*left_context pre* [MASK] *suf* [MASK] *right_context*" and continue predicting the following piece pair, $\langle l_1, r_1 \rangle$ iterating until there are no more ties. At any point in the process, candidates which run out of wordpieces are floated to the top of the working ranked list by base order (A-ending before B-ending).

We inferred the words in context based on the BERT-BASE-UNCASED BERTFORMASKEDLM module obtained via `https://github.com/huggingface/transformers` (version 2.0.0). This required lowercasing all input prior to processing.