1 Details of experimental setup

For the experiments with questions-words and MUSE, we do not remove any questions containing out-of-vocabulary words. We use full test data sets for all modes, therefore our results are consistent.

For MUSE, we use the following tests provided by the library[1].

English:

- EN_YP-130
- EN_SIMLEX-999
- EN_MTurk-771
- EN_RG-65
- EN_VERB-143
- EN_SEMEVAL17
- EN_MTurk-287
- EN_MC-30
- EN_RW-STANFORD
- EN_WS-353-SIM
- EN-TR-3k
- EN_WS-353-ALL
- EN_WS-353-REL

German:

- DE_ZG222
- DE_GUR65
- DE_SIMLEX-999
- DE_GUR350
- DE_SEMEVAL17
- DE_WS-353

For the final MUSE scores, we take arithmetic mean of individual scores obtained from these tests.

For Russian, MUSE provides no tests. Therefore, we download and use the HJ dataset[2].

2 Preparation of data

To prepare German and Russian Wikipedia dumps for training, we modify the wikifil.pl script such that it captures relevant characters and replaces all digits with relevant words in each language. For German, we add *äöüß* to the set of Latin characters. For Russian, we extract Cyrillic characters. Then, we manually truncate the parsed texts to 1 billion characters, and further truncate to the last complete word in the resulting text. For example, if the German enumeration was cut at the 1 billion boundary, such that

```
eins zwei drei vier
```

becomes

```
eins zwei dr
```

[1] https://dl.fbaipublicfiles.com/arrival/wordsim.tar.gz

[2] https://github.com/nlpub/russe-evaluation/blob/master/russe/evaluation/hj.csv

we truncate it to

```
eins zwei
```

Finally, we use `iconv` to ensure UTF-8 format.

For example, for the Russian Wikipedia dump, the order of actions is:

```
wget <wiki dump address>/<ru.dump>
perl wikifil-ru.pl <ru.dump> > ruwiki
head -c 1000000000 ruwiki > ruwiki9
# manually truncate the text
# to the last complete word
iconv -t utf-8 ruwiki9 -o ruwiki9-utf
```

We train the embeddings on the file `ruwiki9-utf`.

3 Compatibility with Hugging Face Tokenizers

We implement tokenization in a way that it is compatible with the files produced by `ByteLevelBPETokenizer` in the Hugging Face Tokenizers library[3]. We apply the same character mapping for UTF-8 characters that use more than one byte, and preprocess the words from the vocabulary such that each begins with a special delimiter character $\dot{G}$.

Note that the number of tokens h must be selected **during** tokenization with the Tokenizers library.

```
bpe = ByteLevelBPETokenizer()
bpe.train([<corpus file>],
          vocab_size=<h>)
bpe.save(<path>, <filename>)
```

(Note the `[]` brackets.)

4 How to run experiments

We provide parameterized code [4] to replicate our experiments. It is a modification of the original fastText library[5]. Note that we disabled the production of .bin file to reduce saving time and save storage space. Our experiments apply to unsupervised training with skip-gram and CBOW.

In order to compile, CMake, Intel ICPC compiler and a CPU with AVX-512 support are required. Please compile with:

```
cmake .
make
```

To run, please use the command:

[3] https://github.com/huggingface/tokenizers

[4] https://github.com/FT-Submit/ft-mod

[5] https://github.com/facebookresearch/fastText

```
fasttext {cbow, skipgram} \
  -input  <corpus file> \
  -output <embeddings file> \
  <arguments>
```

Selecting code optimizations. To run a particular algorithm mode, use the argument `-mode <mode>`. Table 1 explains all available modes.

Table 1: An overview of `-mode` arguments and their connection to our experiments. Parameter s is set with `-shared <s>`.

argument	mode
`-mode normal`	*_code_opt (default)
`-mode batched`	SG_batch (skip-gram only)
`-mode ns`	SG_NS_CT (skip-gram) CBOW_NS_s (CBOW)
`-mode dh`	CBOW_DH (CBOW only)
`-mode dhf`	CBOW_DHF (CBOW only)
`-mode dh-ns`	CBOW_DH_NS_s (CBOW only)
`-mode dhf-ns`	CBOW_DHF_NS_s (CBOW only)

By default, s (the number of words sharing negative samples) is set to $2C+1$, where C is the maximum context window size (set in the original fastText with the argument `-ws <c>`). To set a different s, use `-shared <s>`. Setting this argument to zero will result in the default setting.

Selecting BPE tokens or no_subword. To use BPE tokens instead of subwords, provide paths to the merge and vocab files produced by the Tokenizers library.

```
 -token-merges <path/to/f.txt >
 -token-vocab  <path/to/f.json>
```

Both these arguments must be set to enable the BPE run. The number of tokens h is obtained from the merge and vocab files. For our experiments, we produce these files using our training corpora. Note that these two arguments are incompatible with `-no-subwords`.

Finally, to run the no_subword (word2vec) version of our code, use the argument `-no-subwords`. Note that it is incompatible with `-token-merges` and `-token-vocab`.

We run both the BPE and no_subword experiments using `-mode normal` (default).

The remaining arguments are identical to those used by the original fastText code.

To obtain the results for SG_original and CBOW_original, please run the original library.