# Data Intensive Computing – Handout 3

## **Training Cluster**

You can login to the training cluster *dlrc* via machine ufallab.ms.mff.cuni.cz and port 11422, i.e., using ssh -p 11422 ufallab.ms.mff.cuni.cz.

All machines in the cluster use same architecture and share /dlrc\_share directory.

The cluster consists of master dlrc and 5 nodes dlrc-node1 to dlrc-node5. Each node has 2 cores and 4GB ram, but four SGE jobs are allowed to run simultaneously (only because otherwise there would be too little slots; the slot number may be even increased later).

### Splitting Large Files on Lines

To help you with splitting large files on lines, you can use supplied split\_file.py script from /dlrc\_share/data/wiki/split/ directory. When executed as

split\_file.py input\_file start\_offset length

only lines in the file part starting at offset start\_offset and length are printed (allowing the last line to end outside the file part).

The split\_file.py and also split\_file.pm can be used also as modules in your Python and Perl scripts. In that case, you can use the method

next\_file(file, start\_offset, length)

which sequentially returns lines in the file part (including newline characters), returning empty string at the end.

#### Exercises for using Sun Grid Engine

#### Wikipedia Data

In the /dlrc\_share/data/wiki/ there are following data:

- cs-text/cswiki.txt: Czech Wikipedia data (Sep 2009), file size 195MB, 124k articles.
- en-text/enwiki.txt: English Wikipedia data (Sep 2009), File size 4.9GB, 2.9M articles.

Both files are encoded in UTF-8 and contain one particle per line. Article name is separated by \t character from the article content.

#### Example Solution of a Simple Distributed Task

Consider simple example of producing sorted list of article names. The example solution consists of two sources.

The source **articles**.**sh** is the main driver for distributing work and collecting results:

```
#!/bin/bash
\mathbf{set} -e
\# Parse arguments
["$#" -ge 3 ] || { echo Usage: $0 input outdir tasks \[conc_tasks\] >&2;exit 1;}
input_file="$1"
output_dir="$2"
tasks="$3"
\operatorname{conc}_{tasks} = "\$4"
\# Check that input file exists and get its size
[ -f "$input_file" ] || { echo File $input_file does not exist >&2; exit 1; }
input_size='stat -c%s "$input_file"'
\# Check that output dir does not exist and create it
[ -d "$output_dir" ] && { echo Directory $output_dir already exists >&2; exit 1;}
mkdir -p "$output_dir"
# Compute file split sizes
split_size =  ((1 + (sinput_size / stasks)))
\# Run distributed computations
qsub -cwd -b y -sync y -V -t 1-"$tasks" ${conc_tasks:+-tc $concurrent_tasks} \
 ./<u>articles_distributed</u>.sh "$input_file" "$split_size" "$output_dir"/articles.txt
# Merge all results
sort -m 'seq -f "$output_dir/articles.txt.%g" 1 "$tasks" \
 > "$output_dir"/articles.txt
```

The source articles\_distributed.sh is the helper script executed distributively on the nodes:

```
#!/bin/bash
\mathbf{set} -e
\# Parse arguments
["$#" -ge 3] || { echo Usage: $0 input split_length output_file >&2; exit 1;}
input_file="$1"
\operatorname{split}_{\operatorname{length}} = "\$2"
output_file="$3"
# Parse SGE_TASK_ID and compute file offset
[ -n "$SGE_TASK_ID" ] || { echo Variable SGE_TASK_ID is not set >&2; exit 1; }
task="$SGE_TASK_ID"
input_offset=$((($task - 1) * $split_length))
output_file="$output_file.$task"
# Run computation outputting to temporary file
tmp_file=" 'mktemp'"
trap "rm_-f_-" $tmp_file\"" EXIT
/dlrc_share/data/wiki/split/split_file.py "$input_file" "$input_offset" \
  "$split_length" | cut -f1 | sort > "$tmp_file"
# On success move temporary file to output
mv "$tmp_file" "$output_file"
```

## Tasks

Solve the following tasks. Solution for each task is a source code processing the Wikipedia source data and producing required results, while utilizing distributed computation. The solution does not need to recover when one of the computation fails, but it should fail as a whole.

<ul> <li>unique_words</li> <li>2</li> <li>Create a list of unique words used in the articles. Convert them to lowercase to ignore case. Because the article data is not tokenized, use provided /d1rc_share/data/wiki/tokenizer/{cs,en}_tokenizer, which reads untokenized UTF-8 text from standard input and produces tokenized UTF-8 text on standard output. It preserves line breaks and separates tokens on each line by exactly one space.</li> <li>Compute inverted index – for every lowercased word from the articles, compute ascending (article id, ascending positions of occurrences as word indices) pairs. In order to do so, number the articles using consecutive integers and produce also a list of articles representing this mapping (the article on line i is the article with id i). The output should be a file with list of articles ordered by article id, and a file with one word on a line in the following format: word \t article_id \t space separated occurrences Once again use provided tokenizer.</li> <li>In order to implement word similarity search, compute for each lemma with at least three occurrences. List the contexts in ascending order. Given N (either 1, 2, 3 or 4), the context of a lemma occurrence is N lemmas for a given article, use provided /d1rc_share/data/wiki/lemmatizer/(cs,e,m)_lemmatizer, which words when article boundaries are reached).</li> <li>Wordsim_index</li> <li>3</li> <li>To compute the lemmas for a given article, use provided /d1rc_share/data/wiki/lemmatizer/[cs,e,m]_lemmatizer, which works just like the tokenizer - it reads untokenized UTF-8 text from standard input and produces tokenized at lemma separated by exactly one space.</li> </ul>	Task	Points	Description
<ul> <li>unique_words</li> <li>2</li> <li>them to lowercase to ignore case. Because the article data is not tokenized, use provided /dlrc_share/data/wiki/tokenizer/{cs,en}_tokenizer, which reads untokenized UTF-8 text from standard input and produces tokenized UTF-8 text from standard output. It preserves line breaks and separates tokens on each line by exactly one space.</li> <li>Compute inverted index - for every lowercased word from the articles, compute ascending (article id, ascending positions of occurrences as word indices) pairs. In order to do so, number the articles using consecutive integers and produce also a list of articles representing this mapping (the article on line i is the article with id i). The output should be a file with list of articles ordered by article id, and a file with one word on a line in the following format: word \t article_id \t space separated occurrences Once again use provided tokenizer.</li> <li>In order to implement word similarity search, compute for each lemma with at least three occurrences. List the contexts in ascending order. Given N (either 1, 2, 3 or 4), the context of a lemma occurrence is N lemmas preceding this occurrence and N lemmas follow- ing this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).</li> <li>To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/(cs,en)_lemmatizer, which works just like the tokenizer - it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format: lemma \t context \t counts</li> </ul>			Create a list of unique words used in the articles. Convert
<pre>unique_words unique_words 2 Because the article data is not tokenized, use provided /dlrc_share/data/wiki/tokenizer/{cs,en}_tokenizer, which reads untokenized UTF-8 text from standard input and produces tokenized UTF-8 text on standard output. It preserves line breaks and separates tokens on each line by exactly one space. Compute inverted index - for every lowercased word from the articles, compute ascending (article id, ascending positions of occurrences as word indices) pairs. In order to do so, number the articles using consecutive integers and produce also a list of articles representing this mapping (the article on line i is the article with id i). The output should be a file with list of articles ordered by article id, and a file with one word on a line in the following format: word \t article_id \t space separated occurrences Once again use provided tokenizer. In order to implement word similarity search, compute for each lemma with at least three occurrences all contexts in which it occurs, including their number of occurrences. List the contexts in ascending order. Given N (either 1, 2, 3 or 4), the context of a lemma occurrence is N lemmas preceding this occurrence and N lemmas follow- ing this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached). 3 wordsim_index 3 word sing this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached). 3 words when article boundaries are reached). 3 words when article boundaries are reached). 3 To compute the lemmas for a given article, use provided UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format: lemma \t context \t counts </pre>	unique_words		them to lowercase to ignore case.
<pre>unique_words</pre>		2	Because the article data is not tokenized, use provided
<ul> <li>which reads untokenized UTF-8 text from standard input and produces tokenized UTF-8 text on standard output. It preserves line breaks and separates tokens on each line by exactly one space.</li> <li>Compute inverted index – for every lowercased word from the articles, compute ascending (article id, ascending positions of occurrences as word indices) pairs. In order to do so, number the articles using consecutive integers and produce also a list of articles representing this mapping (the article on line i is the article with id i). The output should be a file with list of articles ordered by article id, and a file with one word on a line in the following format:</li> <li>word \t article_id \t space separated occurrences Once again use provided tokenizer.</li> <li>In order to implement word similarity search, compute for each lemma with at least three occurrences all contexts in which it occurs, including order. Given N (either 1, 2, 3 or 4), the context of a lemma occurrence is N lemmas preceding this oc agiven article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which words yus like the tokenizer – it reads untokenized uTF-8 text from standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format:</li> </ul>			/dlrc_share/data/wiki/tokenizer/{cs,en}_tokenizer,
<ul> <li>and produces tokenized UTF-8 text on standard output. It preserves line breaks and separates tokens on each line by exactly one space.</li> <li>Compute inverted index - for every lowercased word from the articles, compute ascending (article id, ascending positions of occurrences as word indices) pairs. In order to do so, number the articles using consecutive integers and produce also a list of articles representing this mapping (the article on line i is the article with id i).</li> <li>The output should be a file with list of articles ordered by article id, and a file with one word on a line in the following format:</li> <li>word \t article_id \t space separated occurrences Once again use provided tokenizer.</li> <li>In order to implement word similarity search, compute for each lemma with at least three occurrences all contexts in which it occurs, including their number of courrences. List the contexts in ascending order.</li> <li>Given N (either 1, 2, 3 or 4), the context of a lemma occurrence is N lemmas preceding this occurrence and N lemmas following this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).</li> <li>To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/(cs, en]_lemmatizer, which works just like the tokenizer – it reads untokenized uTF-8 text from standard output, each lemma separated by exactly one space.</li> <li>The output should be a file with one lemma on a line in the following format:</li> </ul>			which reads untokenized UTF-8 text from standard input
<ul> <li>wordsim_index</li> <li>wordsim_index</li> <li>To compute 3</li> <li>To compute the 1, 2, 3 or 4), the context of a lemma occurrence is N lemmas preceding this occurrence and N lemmas following the contexts in ascending order.</li> <li>To compute the lemmas for a given article, use provided difference is not context of a lemma tizer, which works just like the tokenizer – it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text from standard output, each lemma separated by exactly one space.</li> </ul>			and produces tokenized UTF-8 text on standard output. It
<ul> <li>exactly one space.</li> <li>Compute inverted index – for every lowercased word from the articles, compute ascending (article id, ascending positions of occurrences as word indices) pairs. In order to do so, number the articles using consecutive integers and produce also a list of articles representing this mapping (the article on line i is the article with id i).</li> <li>The output should be a file with list of articles ordered by article id, and a file with one word on a line in the following format:         <ul> <li>word \t article_id \t space separated occurrences</li> <li>Once again use provided tokenizer.</li> <li>In order to implement word similarity search, compute for each lemma with at least three occurrences all contexts in which it occurs, including their number of occurrence. List the contexts in ascending order.</li> <li>Given N (either 1, 2, 3 or 4), the context of a lemma occurrence is N lemmas preceding this occurrence and N lemmas following this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).</li> </ul> </li> <li>wordsim_index</li> <li>3 To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer – it reads untokenized uTF-8 text from standard output, each lemma separated by exactly one space.         <ul> <li>The output should be a file with one lemma on a line in the following format:</li> <li>lemma \t context \t counts</li> </ul> </li> </ul>			preserves line breaks and separates tokens on each line by
<ul> <li>Compute inverted index - for every lowercased word from the articles, compute ascending (article id, ascending positions of occurrences as word indices) pairs. In order to do so, number the articles using consecutive integers and produce also a list of articles representing this mapping (the article on line i is the article with id i). The output should be a file with list of articles ordered by article id, and a file with one word on a line in the following format:</li></ul>			exactly one space.
<ul> <li>articles, compute ascending (article id, ascending positions of occurrences as word indices) pairs. In order to do so, number the articles using consecutive integers and produce also a list of articles representing this mapping (the article on line i is the article with id i). The output should be a file with list of articles ordered by article id, and a file with one word on a line in the following format:</li> <li>word \t article_id \t space separated occurrences Once again use provided tokenizer.</li> <li>In order to implement word similarity search, compute for each lemma with at least three occurrences all contexts in which it occurs, including their number of occurrences. List the contexts in ascending order. Given N (either 1, 2, 3 or 4), the context of a lemma occurrence is N lemmas preceding this occurrence and N lemmas following this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).</li> <li>wordsim_index</li> <li>3 To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer - it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format:</li> </ul>			Compute inverted index – for every lowercased word from the
<ul> <li>wordsim_index</li> <li>wordsim_index</li> <li>To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatized UTF-8 text for standard output, each lemma separated by eactly one space.</li> </ul>	inverted_index	3	articles, compute ascending (article id, ascending positions of
<ul> <li>inverted_index</li> <li>the articles using consecutive integers and produce also a list of articles representing this mapping (the article on line <i>i</i> is the article with id <i>i</i>). The output should be a file with list of articles ordered by article id, and a file with one word on a line in the following format:         <ul> <li>word \t article_id \t space separated occurrences</li> <li>Once again use provided tokenizer.</li> <li>In order to implement word similarity search, compute for each lemma with at least three occurrences all <i>contexts</i> in which it occurs, including their number of occurrences. List the contexts in ascending order.</li> <li>Given N (either 1, 2, 3 or 4), the <i>context</i> of a lemma occurrence is N lemmas preceding this occurrence and N lemmas following this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).</li> </ul> </li> <li>wordsim_index</li> <li>To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer – it reads untokenized UTF-8 text from standard output, each lemma separated by exactly one space.         <ul> <li>The output should be a file with one lemma on a line in the following format:</li> <li>lemma \t context \t counts</li> </ul> </li> </ul>			occurrences as word indices) pairs. In order to do so, number
<ul> <li>inverted_index</li> <li>of articles representing this mapping (the article on line <i>i</i> is the article with id <i>i</i>). The output should be a file with list of articles ordered by article id, and a file with one word on a line in the following format:         word \t article_id \t space separated occurrences Once again use provided tokenizer.         <ul> <li>In order to implement word similarity search, compute for each lemma with at least three occurrences all <i>contexts</i> in which it occurs, including their number of occurrences. List the contexts in ascending order.</li> <li>Given N (either 1, 2, 3 or 4), the <i>context</i> of a lemma occurrence is N lemmas preceding this occurrence boundaries, use empty words when article boundaries are reached).</li> <li>To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer – it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space.</li> <li>The output should be a file with one lemma on a line in the following format:</li> <li>Iemma \t context \t counts</li> </ul> </li> </ul>			the articles using consecutive integers and produce also a list
<ul> <li>inverted_index</li> <li>3 the article with id i). The output should be a file with list of articles ordered by article id, and a file with one word on a line in the following format: word \t article_id \t space separated occurrences Once again use provided tokenizer.</li> <li>In order to implement word similarity search, compute for each lemma with at least three occurrences all <i>contexts</i> in which it occurs, including their number of occurrences. List the contexts in ascending order. Given N (either 1, 2, 3 or 4), the <i>context</i> of a lemma occurrence is N lemmas preceding this occurrence and N lemmas follow- ing this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached). To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer - it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format: lemma \t context \t counts</li> </ul>			of articles representing this mapping (the article on line $i$ is
<ul> <li>wordsim_index</li> <li>3</li> <li>The output should be a file with list of articles ordered by article id, and a file with one word on a line in the following format: word \t article_id \t space separated occurrences Once again use provided tokenizer.</li> <li>In order to implement word similarity search, compute for each lemma with at least three occurrences all <i>contexts</i> in which it occurs, including their number of occurrences. List the contexts in ascending order. Given N (either 1, 2, 3 or 4), the <i>context</i> of a lemma occurrence is N lemmas preceding this occurrence and N lemmas following this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).</li> <li>To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer – it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format: lemma \t context \t counts</li> </ul>			the article with $id i$ ).
<ul> <li>article id, and a file with one word on a line in the following format:</li> <li>word \t article_id \t space separated occurrences Once again use provided tokenizer.</li> <li>In order to implement word similarity search, compute for each lemma with at least three occurrences all <i>contexts</i> in which it occurs, including their number of occurrences. List the contexts in ascending order.</li> <li>Given N (either 1, 2, 3 or 4), the <i>context</i> of a lemma occurrence is N lemmas preceding this occurrence and N lemmas following this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).</li> <li>To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer – it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format:</li> </ul>			The output should be a file with list of articles ordered by
<ul> <li>format:</li> <li>word \t article_id \t space separated occurrences Once again use provided tokenizer.</li> <li>In order to implement word similarity search, compute for each lemma with at least three occurrences all <i>contexts</i> in which it occurs, including their number of occurrences. List the contexts in ascending order.</li> <li>Given N (either 1, 2, 3 or 4), the <i>context</i> of a lemma occurrence is N lemmas preceding this occurrence and N lemmas follow- ing this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).</li> <li>To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer – it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format: lemma \t context \t counts</li> </ul>			article id, and a file with one word on a line in the following
<ul> <li>word \t article_id \t space separated occurrences Once again use provided tokenizer.</li> <li>In order to implement word similarity search, compute for each lemma with at least three occurrences all <i>contexts</i> in which it occurs, including their number of occurrences. List the contexts in ascending order.</li> <li>Given N (either 1, 2, 3 or 4), the <i>context</i> of a lemma occurrence is N lemmas preceding this occurrence and N lemmas follow- ing this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).</li> <li>To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer – it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format: lemma \t context \t counts</li> </ul>			format:
Once again use provided tokenizer.In order to implement word similarity search, compute for each lemma with at least three occurrences all contexts in which it occurs, including their number of occurrences. List the contexts in ascending order. Given N (either 1, 2, 3 or 4), the context of a lemma occurrence is N lemmas preceding this occurrence and N lemmas follow- ing this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).wordsim_index3To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer - it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format: lemma \t context \t counts			word \t article_id \t space separated occurrences
<ul> <li>In order to implement word similarity search, compute for each lemma with at least three occurrences all contexts in which it occurs, including their number of occurrences. List the contexts in ascending order. Given N (either 1, 2, 3 or 4), the context of a lemma occurrence is N lemmas preceding this occurrence and N lemmas following this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).</li> <li>wordsim_index</li> <li>To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer – it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format: lemma \t context \t counts</li> </ul>			Once again use provided tokenizer.
<ul> <li>each lemma with at least three occurrences all contexts in which it occurs, including their number of occurrences. List the contexts in ascending order.</li> <li>Given N (either 1, 2, 3 or 4), the context of a lemma occurrence is N lemmas preceding this occurrence and N lemmas following this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).</li> <li>wordsim_index</li> <li>3 To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer – it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space.</li> <li>The output should be a file with one lemma on a line in the following format:</li> <li>lemma \t context \t counts</li> </ul>		3	In order to implement word similarity search, compute for
<ul> <li>which it occurs, including their number of occurrences. List the contexts in ascending order. Given N (either 1, 2, 3 or 4), the context of a lemma occurrence is N lemmas preceding this occurrence and N lemmas following this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).</li> <li>wordsim_index</li> <li>To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer – it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format:</li> <li>lemma \t context \t counts</li> </ul>			each lemma with at least three occurrences all <i>contexts</i> in
<ul> <li>the contexts in ascending order.</li> <li>Given N (either 1, 2, 3 or 4), the context of a lemma occurrence is N lemmas preceding this occurrence and N lemmas following this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).</li> <li>To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer – it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space.</li> <li>The output should be a file with one lemma on a line in the following format: lemma \t context \t counts</li> </ul>	wordsim_index		which it occurs, including their number of occurrences. List
<ul> <li>Given N (either 1, 2, 3 or 4), the context of a lemma occurrence is N lemmas preceding this occurrence and N lemmas following this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).</li> <li>To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer – it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format:</li> </ul>			the contexts in ascending order.
<ul> <li>is N lemmas preceding this occurrence and N lemmas following this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).</li> <li>3 To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer - it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format:</li> <li>lemma \t context \t counts</li> </ul>			Given $N$ (either 1, 2, 3 or 4), the <i>context</i> of a lemma occurrence
wordsim_indexing this occurrence (ignore sentence boundaries, use empty words when article boundaries are reached).3To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer - it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format: lemma \t context \t counts			is $N$ lemmas preceding this occurrence and $N$ lemmas follow-
wordsim_index3words when article boundaries are reached).3To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer - it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format: lemma \t context \t counts			ing this occurrence (ignore sentence boundaries, use empty
wordsim_index3To compute the lemmas for a given article, use provided /dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer - it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format: lemma \t context \t counts			words when article boundaries are reached).
/dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer, which works just like the tokenizer – it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format: lemma \t context \t counts			To compute the lemmas for a given article, use provided
which works just like the tokenizer – it reads untokenized UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format: lemma \t context \t counts			/dlrc_share/data/wiki/lemmatizer/{cs,en}_lemmatizer,
UTF-8 text from standard input and produces tokenized and lemmatized UTF-8 text on standard output, each lemma separated by exactly one space. The output should be a file with one lemma on a line in the following format: lemma \t context \t counts			which works just like the tokenizer – it reads untokenized
lemmatized UTF-8 text on standard output, each lemmaseparated by exactly one space.The output should be a file with one lemma on a line in thefollowing format:lemma \t context \t counts			UTF-8 text from standard input and produces tokenized and
separated by exactly one space. The output should be a file with one lemma on a line in the following format: lemma \t context \t counts			lemmatized UTF-8 text on standard output, each lemma
The output should be a file with one lemma on a line in the following format: lemma \t context \t counts			separated by exactly one space.
following format: lemma \t context \t counts			The output should be a file with one lemma on a line in the
lemma \t context \t counts			following format:
			lemma \t context \t counts
Let S be given natural number. Using the index created in	wordsim_find	2	Let S be given natural number. Using the index created in
wordsim_index, find for each lemma S most similar lemmas.			wordsim_index, and for each lemma 5 most similar lemmas.
The similarity of two lemmas is computed using costne sis computed using costne similarity of t			The similarity of two lemmas is computed using cosine similarity of $C_{A} \cdot C_{B}$ where $C_{A} \cdot C_{B}$ is a vector of accumulation of a computed using cosine similarity of the similarity o
wordsim_find $\begin{bmatrix} 2 \\ L \end{bmatrix}$ and $\begin{bmatrix} uarray as \frac{C_L + C_L}{C_L} \end{bmatrix}$ , where $C_L$ is a vector of occurrences of lemma			$L_{aontovts}$ where $C_L$ is a vector of occurrences of lemma
$\begin{bmatrix} L \text{ contexts.} \\ The output sheld he a file with one lemme on a line in the$			$\mathcal{L}$ contexts.
following format:			following format:
lemma \t most similar lemma \t cosine similarity			lemma \t most similar lemma \t cogine similarity