Peacekeeping Operations Corpus: Corpus design and creation

Elio Amicarelli

PKOC0A03

Contents

1. Summary

The Peacekeeping Operation Corpus (PKOC) Data Project is commissioned by the Research Development Fund, Faculty of Social Sciences, University of Warwick. The objective of this consultancy is to design, create and explore a professional digital corpus from the United Nations' Secretary General reporting on peacekeeping mission to the Security Council.

This technical report provides all the details needed to reproduce the corpus, to update it and to work with it. It describes the pipeline for the corpus generation as well as the structure of the corpus itself. It also provides full documentation of the Python 3 code needed for researcher to create and update the corpus.

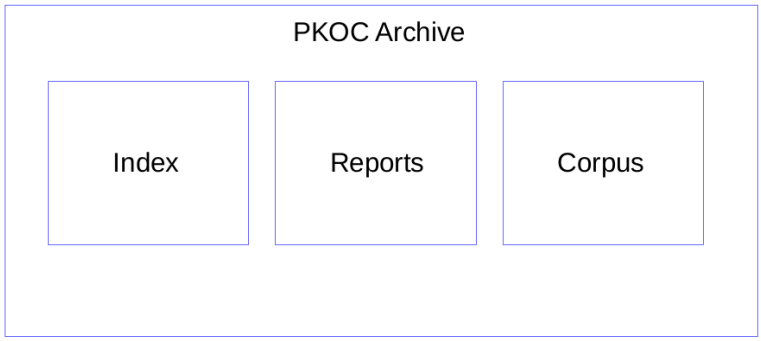
The delivery associated with this document PKOC0A03 includes:

- A professional digital corpus of the United Nations' Secretary general reports on peacekeeping missions to the Security Council from 1994 to 2019. The corpus is delivered to the client in the following formats: *a)* raw texts (a collection of .txt files and .pdf files), *b)* a set of 3 metadata-enriched python dictionaries corresponding to the plain, reduced and tagged version of the corpus (see section *2.Corpus design*), *c)* an R builder for the corpus.
- Python 3 code automating the corpus creation. This code allows the client to fully reproduce the corpus creation and to update it when needed.
- Full documentation on corpus design and creation (this document PKOC0A03).
- Full code documentation (this document PKOC0A03).

2. Corpus design and creation

2.1 The Archive

The creation and maintenance of PKOC relies on the Archive. The Archive is a folder structure underlying the entire automation for the corpus generation. The Archive is divided in the three main folders, that is the 'index', the 'reports' and the 'corpus'.



The Archive tripartite structure showed in the figure above mirrors the three stages for the corpus creation which are described below.

2.2 Stages of corpus creation

Stage 1: Index generation and update

The creation of the corpus starts with a bot collecting a comprehensive list of URLs to peacekeeping missions documents from the United Nations website. This list, called 'index', is saved in the **Archive/index** and represent the reference to be used for two main purposes:

- Automatically download the United Nations peacekeeping mission reports
- Automatically verify in any given moment if new reports where uploaded on the United Nations website and, if so, download only the new documents added on the website.

Stage 2: Reports download, conversion and sanity check

The index is then used by a bot to download all the peacekeeping mission reports in pdf and word formats. The pdf reports are downloaded in **Archive/reports/reports_archive**. The system has a function to flag which documents included in the index were not correctly downloaded.

When the reports_archive is finally populated with all the peacekeeping reports in .pdf (or .doc) format, they are converted to a .txt and saved in **Archive/reports/texts_archive** format in order to allow subsequent manipulations. After the conversion, there is a sequence of sanity checks and filtering stages operated by the system in order to ensure that the .txt documents that will move to the next stage are *correct* and *relevant*:

1. Identify and fix documents not correctly converted from .pdf to .txt
2. Identify and isolate documents which are not relevant for the scope of this corpus (e.g. reports that are not mission-specific, political reports etc)

Stage 3: Corpus raw texts, metadata generation and corpus dictionaries

Raw texts

After the relevant texts are separated from the irrelevant ones, the system moves the relevant texts in **Archive/corpus/corpus_texts_archive**. This is considered as the first version of the corpus that is the *raw texts corpus*. At the moment of writing the raw texts corpus contains 1080 reports in .txt and .pdf format.

Metadata

From each report in the *raw texts corpus* the system extracts *metadata* and creates a dictionary mapping each report title to its metadata information. Thanks to metadata we move from a corpus of unstructured data (just a collection of documents) to a structured collection where encoded information available for each document allows to select, subset, filter, combine the documents according with research needs. A full metadata string generated by the system and associated with a document looks like this:

UNMIS-S/2005/579-S2005579-7-2005-September-SDN

The table below summarises the metadata extracted and mapped to each report:

Table 1: Document metadata in PKOC

dimension	example	description
mission name	UNMIS	The official mission acronym
report code	S/2005/579	The official code uniquely identifying the report
report name	S2005579	The name of the file from which metadata were extracted
mission period	7	Number of months since deployment
report year	2005	The year of the report
report month	September	The month of the report
mission iso country code alpha 3	SDN	The country iso code

Plain PKOC, reduced PKOC, tagged PKOC (Python dictionaries)

The metadata and the raw texts corpus are then used to create the following three versions of the corpus:

1. Plain PKOC (pPKOC): This is basically a digital version of raw reports 'as they are' but mapped to their metadata. pPKOC has two main advantages: first, it allows moving across reports and to easily query them in their original format, and second it offers to experienced users the possibility to highly customise the corpus. For example, a sentence in pPKOC looks like this:

   UNPROFOR has been actively negotiating the reopening of the routes across the airport that had brought freedom of movement and a major improvement of living conditions, including the near-disappearance of black-marketeering, to all citizens of Sarajevo.

2. Reduced PKOC (rPKOC): Reduced PKOC (rPKOC) contains a pre-processed version of the reports where stopwords, punctuation, numbers and symbols are removed, words are lowercased and stemmed. These pre-processing steps are commonly applied before performing some types of analyses relying on document-term matrices and bag-of-words models. The same sentence reported before in rPKOC has the following representation:

   'unprofor', 'activ', 'negoti', 'reopen', 'rout', 'across', 'airport', 'brought', 'freedom', 'movement', 'major', 'improv', 'live', 'condit', 'includ', 'neardisappear', 'blackmarket', 'citizen', 'sarajevo'.

3. Tagged PKOC (tPKOC): The tagged version called tPKOC is similar to the plain one in the sense that text is complete but the distinctive feature here is that each word has a grammatical annotation. This means that each word is associated with a grammatical class:

   ('UNPROFOR', 'NN'), ('has', 'VBZ'), ('been', 'VBN'), ('actively', 'RB'), ('negotiating', 'VBG'), ('the', 'DT'), ('reopening', 'NN'), ('of', 'IN'), ('the', 'DT'), ('routes', 'NNS'), ('across', 'IN'), ('the', 'DT'), ('airport', 'NN'), ('that', 'WDT'), ('had', 'VBD'), ('brought', 'VBN'), ('freedom', 'NN'), ('of', 'IN'), ('movement', 'NN'), ('and', 'CC'), ('a', 'DT'), ('major', 'JJ'), ('improvement', 'NN'), ('of', 'IN'), ('living', 'NN'), ('conditions', 'NNS'), (',', ','), ('including', 'VBG'), ('the', 'DT'),

('near-disappearance', 'NN'), ('of', 'IN'), ('black-marketeering', 'NN'), (',', ','), ('to', 'TO'), ('all', 'DT'), ('citizens', 'NNS'), ('of', 'IN'), ('Sarajevo', 'NNP'), ('.', '.').

These versions of the corpus are located in **Archive/corpus/corpus_dictionaries_archive/**.

PKOC in R

An R builder for PKOC is located **Archive/corpus/corpus_r/**.

3. Code documentation

This section documents all the functions created to automate the corpus generation pipeline.

3.1 Index functions

1. **build_index(save, index_archive)**

   build_index() connects to the United Nations website an collects all the links to the Secretary General peacekeeping reports. It returns a list with all the links, this list is what we call index.

- *save = True/False*: If True the index is saved as a text file in a path specified by the argument index_archive. The current timestamp is used as index file name.
- *index_archive = '.../Archive/index/index_archive/'*: A path where the index is saved as a text file if save = True.

2. **import_index(index_archive, index_name)**

   import_index() loads an existing index text file and returns a list.

- *index_archive = '.../Archive/index/index_archive/'*: A path containing indexes as txt files (see *build_index()*).
- *index_name = 'index_2019-07-17.txt'*: The name of the index to be imported from the index_archive.

3. **update_index(save, index_archive, index_name)**

   update_index() is used to update the corpus. It retrieves the latest list of link to reports available on the United Nations website and compares them with the links from an existing index in order to check if an update is needed. If the existing index needs to be updated and if save = True, it saves the latest list of links as a new index in the index_archive. In addition, the links added in the new index that were not in the old index are saved as an addendum in the index_archive.

- *save = True/False*: If True and if the existing index needs to be updated, it saves the new index as a text file in path specified for the index archive as well as an addendum text file containing only the links that are in the new index but are not in the old index.
- *index_archive = '.../Archive/index/index_archive/'*: A path containing indexes as txt files (see *build_index()*).
- *index_name = 'index_2019-07-17.txt'*: The name of the index to be imported and compared with the latest snapshot of available reports from the United Nations website.

3.2 Reports functions

4. **download_reports(index, driver, doc, reports_archive)**

   dowload_reports() downloads the reports listed in the index.

- *index = [index]*: An index specified as a list (see *build_index()* and *update_index()*).
- *driver = '.../utils/chromedriver'*: A path to a web browser driver to be used by selenium.
- *doc = False*: If False download the pdf otherwise download the word documents.
- *reports_archive = '.../Archive/reports/reports_archive/'*: The path to a folder where all the reports will be downloaded.

5. **match_index_reports(index, reports_archive)**

   match_index_reports() checks that all the reports were correctly downloaded. It compares the files specified in the index with the files actually downloaded. Returns a list of files from the index that were not correctly downloaded.

- *index = [index]*: An index specified as a list (see *build_index()* and *update_index()*).
- *reports_archive = '.../Archive/reports/reports_archive/'*: The path to a folder containing all the downloaded reports.

6. **isolate_reports(reports_archive, texts_archive,isolated_reports_archive, ...)**

   isolate_reports() identifies reports with wrong conversion from pdf to txt and move or copy the pdfs to a separate folder so that they can be examined separately. It returns the list of reports identified as wrongly converted.

- *reports_archive = '.../Archive/reports/reports_archive/'*: The path to a folder containing all the downloaded reports.
- *texts_archive = '.../Archive/reports/texts_archive/'*: The path to a folder containing all the reports converted to a txt format.
- *isolated_reports_archive = '.../Archive/reports/isolated_reports_archive/'*: The path to a folder where the pdfs corresponding to wrong conversions will be copied or moved.
- *min_length = 5*: Number of rows in a file to be used in order to classify it as a wrong conversion. Usually a wrongly converted pdf contains only one or two rows in its txt format.
- *clean_destination = False*: If True all files in isolated_reports_archive are deleted before moving the new ones.
- *move = False*: If True the pdfs with corrupted conversions are moved from reports_archive to isolated_reports_archive.
- *copy = False*: If True the pdfs with corrupted conversions are copied from reports_archive to isolated_reports_archive.

7. **isolate_texts(texts_archive, isolated_texts_archive, min_length, ...)**

   isolate_texts() identifies reports with wrong conversion from pdf to txt and move/copy the txts to a separate folder so that they can be examined separatedly. It returns the list of texts identified as wrongly converted.

- *texts_archive = '.../Archive/reports/texts_archive/'*: The path to a folder containing all the reports converted to txt format.
- *isolated_texts_archive = '.../Archive/reports/isolated_texts_archive/'*: The path to the folder where the txts corresponding to wrong conversions will be copied or moved.
- *min_length = 5*: Number of rows in a file to be used in order to classify it as a wrong conversion. Usually a wrongly converted pdf contains only one or two rows.
- *clean_destination = False*: If True all files in isolated_texts_archive are deleted before moving the new ones.
- *move = False*: If True the txts with corrupted conversions are moved from texts_archive to isolated_texts_archive.
- *copy = False*: If True the txts with corrupted conversions are copied from texts_archive to isolated_texts_archive.

8. **filter_non_missions_texts(texts_archive, isolated_texts_archive, ...)**

   filter_non_missions() checks if the .txt files are missions specific if not (e.g. politcal reports and issues overviews) moves/copies them to a separate folder. It returns two list, the first list contains the name of the files identified as mission specific while the second list contains those identified as non mission specific.

- *texts_archive = '.../Archive/reports/texts_archive/'*: The path to a folder containing all the reports converted to txt format.
- *isolated_texts_archive = '.../Archive/reports/isolated_texts_archive/'*: The path to the folder where the txts corresponding to wrong conversions will be copied or moved.
- *irrelevant_texts_archive = '.../Archive/reports/irrelevant_texts_archive/'*: The path to a folder where the txts that are not mission specific will be copied or moved.
- *missions_info = '.../utils/missions_master.txt'*: A text file with headless comma separated value on mission name, year start, month start, country, country iso, country iso (e.g. BINUB,2007,1,Burundi,BDI,108).
- *clean_destination = False*: If True all files in irrelevant_texts_archive are deleted before moving or copying the new ones.
- *move = False*: If True the .txt files identified as non mission specific are moved from texts_archive to irrelevant_texts_archive.
- *copy = False*: If True the .txt files identified as non mission specific are copied from texts_archive to irrelevant_texts_archive

9. **reports_metadata(texts_archive, metadata_archive, missions_info, missions_periods)**

   reports_metadata() creates the following metadata for the each report: *mission name, report code, report name, mission period, report year, report month, mission country iso*. It returns two lists, that is the keys list and the exceptions list. Each element of the keys list contains *report code, mission name, report metadata, complete path to the txt file*. The exceptions list contains the complete path to each file for which metadata cannot be completely extracted. The set of files obtained from the exceptions is equivalent to those obtained from *isolated_text_archive()* plus *irrelevant_texts_archive()*.

- *texts_archive = '.../Archive/reports/texts_archive/'*: The path to a folder containing all the reports converted to txt format.

- *metadata_archive = '.../Archive/corpus/metadata/'*: The path to a folder where the metadata will be saved.
- *missions_info = '.../utils/missions_master.txt'*: A text file with headless comma separated value on mission name, year start, month start, country, country iso, country iso (e.g. BINUB,2007,1,Burundi,BDI,108).
- *missions_periods = '.../utils/missions_periods.txt'*: A text file with headless comma separated value on mission name, date, mission period, rowid, country iso, country iso (e.g. BONUCA,2000-02-01,1,BONUCA2000February,CAF,140).

note: to generate missions_master.txt and missions_periods.txt files use the script create_periods.R that can be found in utils.

3.3 Corpus functions

10. **corpus_raw_texts(texts_archive, exceptions, corpus_texts_archive, clean_destination)**

    corpus_raw_texts() copies files from the texts archive to the corpus text archive allowing to specify exceptions.

- *texts_archive = '.../Archive/reports/texts_archive/'*: The path to a folder containing all the reports converted to txt format.
- *exceptions = [texts_removed]*: List with file names not to be copied. This list is the second list returned by *reports_metadata()*.
- *corpus_texts_archive = '.../Archive/corpus/corpus_texts_archive/'*: The path to a folder where files from texts archive minus exceptions will be copied.
- *clean_destination = False*: If True remove all files in corpus_text_archive before the copy operation.

11. **corpus_create_ppkoc(metadata = reports_metadata)**

    corpus_create_ppkoc() uses metadata to create the dictionary for the plain peacekeeping operation corpus.Returns a dictionary where each key:value is defined as report metadata:full report.

- metadata = [reports_metadata]: A list of reports metadata as the first list returned by *reports_metadata()*.

12. **corpus_create_rpkoc(plain_corpus = ppkoc)**

    corpus_create_rpkoc() uses metadata to create the dictionary for the reduced peacekeeping operation corpus. Returns a dictionary where each key:value is defined as report metadata:reduced report.

- *plain_corpus = {ppkoc}*: A dictionary containing the plain peacekeeping operation corpus as returned by *corpus_create_ppkoc()*.

13. **corpus_create_tpkoc(plain_corpus = ppkoc)**

    corpus_create_tpkoc() uses metadata to create the dictionary for the tagged peacekeeping operation corpus. Returns a dictionary where each key:value is defined as report metadata:tagged report

- *plain_corpus = {ppkoc}*: A dictionary containing the plain peacekeeping operation corpus as returned by *corpus_create_ppkoc()*.

4. Vignette

```
# 1. Build or import an index.

links_index = build_index(save = True, index_archive = index_archive_url)

links_index = import_index(index_archive = index_path, index_name = index_title)

# 2. Download the United Nations reports.

download_reports(index = links_index, driver = driver_path, doc = False,
                 reports_archive = reports_path)

# 3. Check mismatch between the index and the downloaded reports.
#    With this info you can check why these reports were not downloaded.

match_index_reports(index = links_index, reports_archive = reports_path)

# 4. At this point we convert files from pdf to txt in bash using pdftotext (omitted).

# 5. Optional: isolating and identifying irrelevant text

isolated_reports = isolate_reports(reports_archive = reports_path,
                                   texts_archive = texts_path,
                                   isolated_reports_archive = isolated_reports_path,
                                   min_length = 10, clean_destination = True,
                                   move = False, copy = True)

isolated_texts = isolate_texts(texts_archive = texts_path,
                               isolated_texts_archive = isolated_texts_path,
                               min_length = 10, clean_destination = True,
                               move = False, copy = True)

positives, negatives = filter_non_missions_texts(texts_archive = texts_path,
                                    isolated_texts_archive = isolated_texts_path,
                                    irrelevant_texts_archive=irrelevant_texts_path,
                                    missions_info = missions_master_path,
                                    clean_destination = True, move = False, copy = True)

# 6. Create reports metadata

mykeys, exc = reports_metadata(texts_archive = texts_path,
                               metadata_archive = metadata_path,
                               missions_info = missions_master_path,
                               missions_periods = missions_periods_path)

# 7. Create raw corpus

exclusions = [a.split('/')[-1] for a in exc]

corpus_raw_texts(texts_archive = texts_path, exceptions = exclusions,
                 corpus_texts_archive = corpus_texts_path, clean_destination = True)
```

```
# 8. Create plain PKOC

ppkoc = corpus_create_ppkoc(metadata = mykeys)

# 9. Create reduced PKOC

rpkoc = corpus_create_rpkoc(plain_corpus = ppkoc)

# 10. Create tagged PKOC

tpkoc = corpus_create_tpkoc(plain_corpus = ppkoc)
```

5. Descriptives

At the time of this delivery, every version of the Peacekeeping Operations Corpus contains 1414 documents from 67 peacekeeping missions. The plot in Figure 1 shows the number of reports currently available by mission. As can be seen reporting coverage varies across missions.

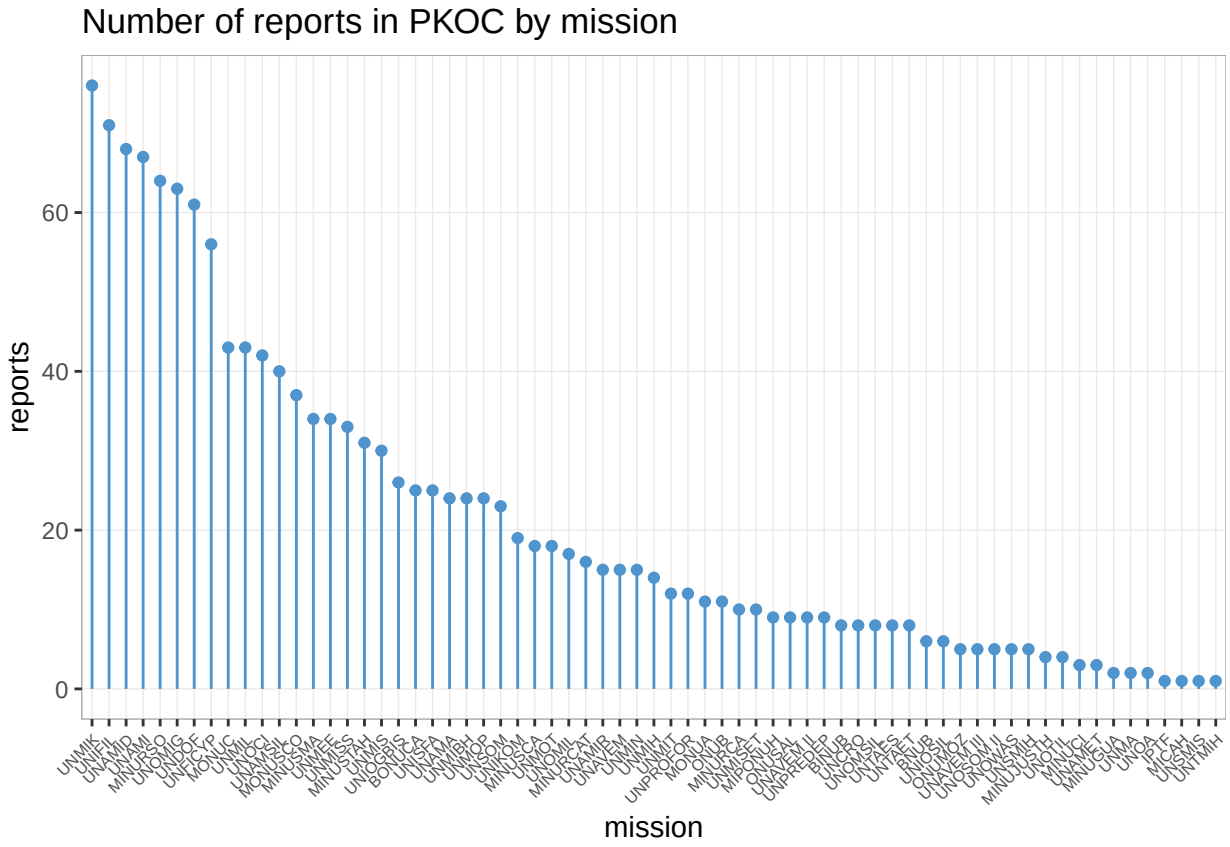


Figure 1: Number of reports by mission

The plot in Figure 2 shows the reporting coverage by period of deployment (see *Section 2.2, Stage 3, Metadata* for an explanation of deployment period metadata). Only missions started during or after 1990 are included in this plot. Orange tiles are deployment periods where a report was released, blue tiles are deployment periods without reports.

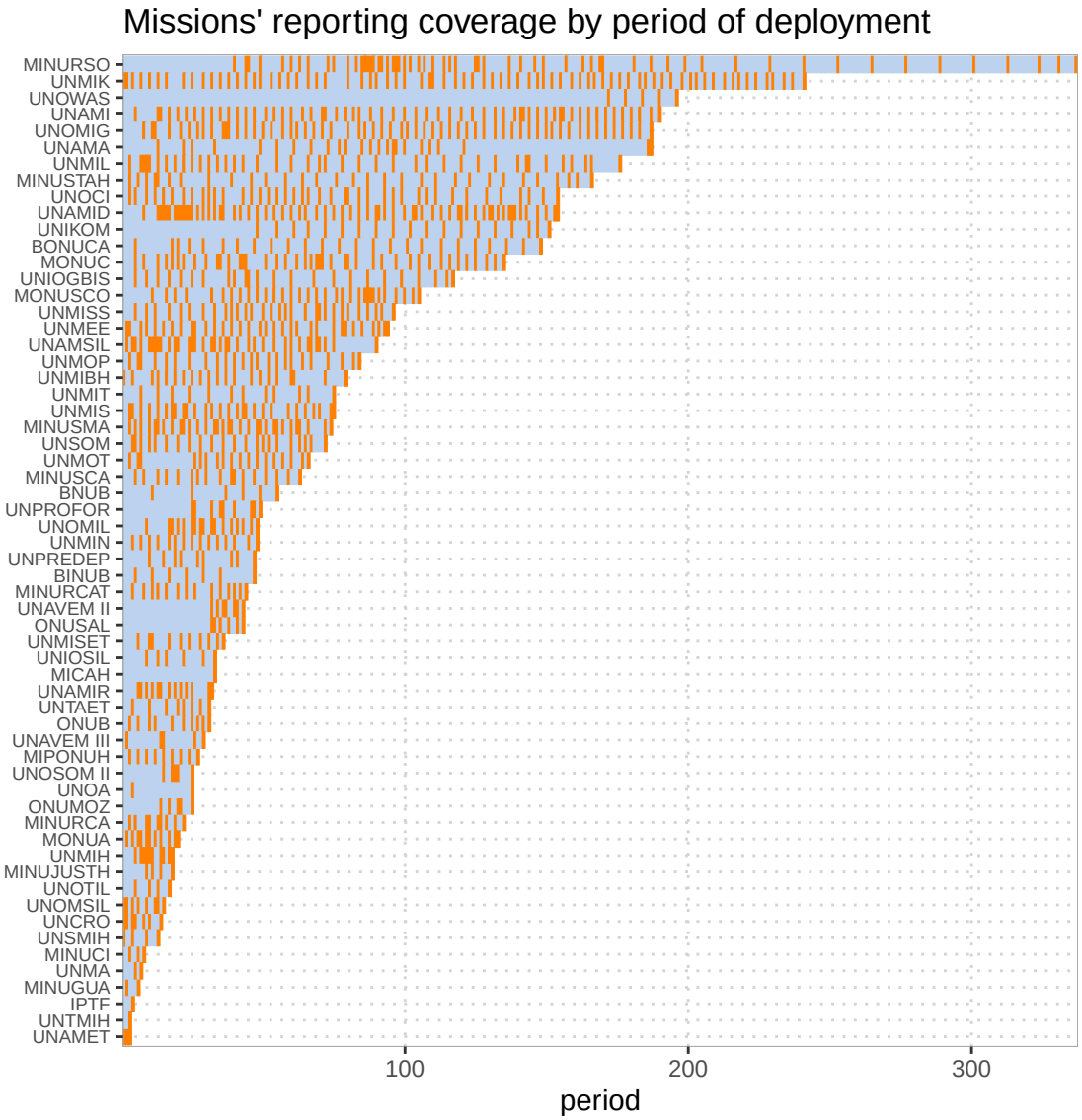


Figure 2: Coverage by period of deployment

Figure 3 summarizes the lenght of reporting by mission as computed using the plain version of the corpus. Over time reporting has increased in length moving from a median length around 2500-5000 tokens in the period before the year 2003 toward a median around 10000 tokens during the last decade.

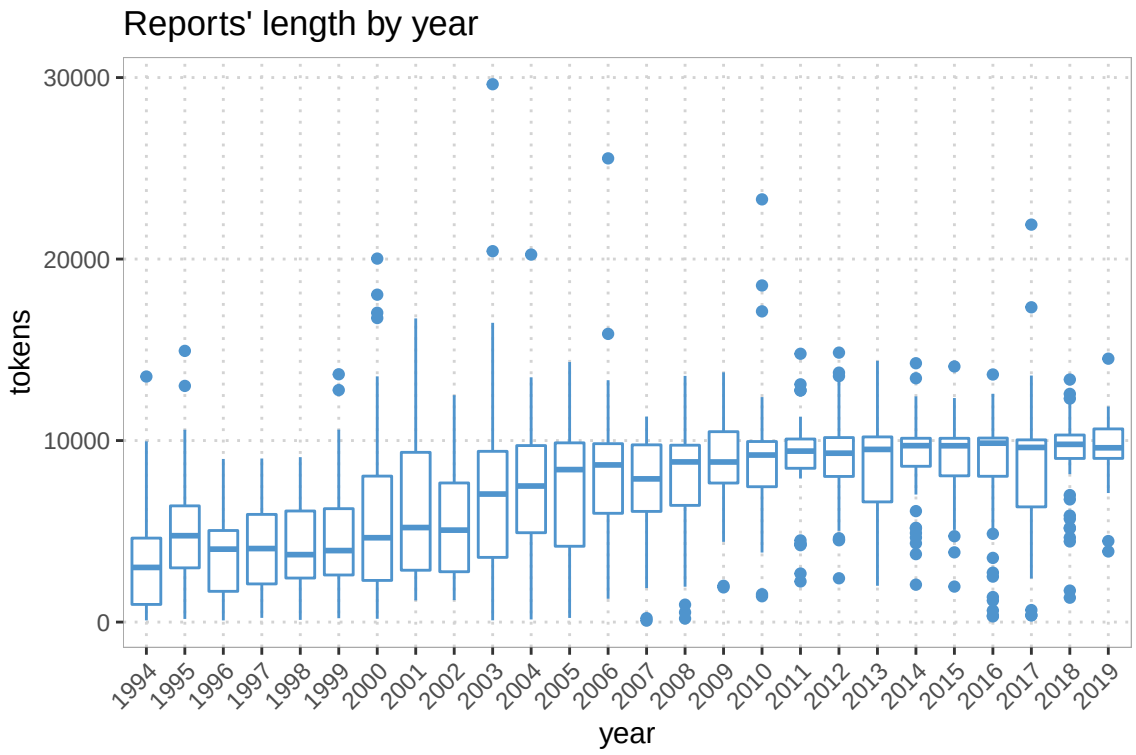


Figure 3: Reports' length over time

6. References

Bird, S., Klein, E. and Loper, E., 2009. `Natural language processing with Python: analyzing text with the natural language toolkit`. O'Reilly Media, Inc.

Jurafsky, Daniel, and James H. Martin. 2009. `Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics`. Prentice-Hall.

Manning, C. et al. 1999. `Foundations of statistical natural language processing`. MIT press.

Manning, C. et al. 2014. `The Stanford CoreNLP Natural Language Processing Toolkit` In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.*