Facilitating Annotation of Collective Agreements with Keywords Extraction

(Update 14/10/2019)

Daniela Ceccon

WageIndicator Foundation, Amsterdam – CLARIN, Utrecht
danielaceccon@wageindicator.org

Casper Kaandorp

University of Amsterdam

"Social Sciences and Humanities Open Cloud", has received funding from the European Union's Horizon 2020 project call H2020-INFRAEOSC-04-2018, grant agreement #823782

WageIndicator.org

You Share, We Compare

THE WAGEINDICATOR DATABASES

MINIMUM WAGES

LABOUR LAW

WAGES IN CONTEXT:

- **Real wages**
- **Living wages**

COLLECTIVE AGREEMENTS DATABASE

- Established in 2012
- Currently contains 1000 collective agreements from 52 countries
- Agreements are coded, annotated (749 variables and nine main topics) and published in the national sites run by WageIndicator. For each topic/question, the relevant part of text is also selected.

WageIndicator.org

You Share, We Compare

THE WAGEINDICATOR D

WageIndicator is a foundation based in the Netherlands and running websites in around 100 countries. Through its national websites, it collects, compares and shares information about Wages, Labour Law and Career.

MINIMUM W

WAGES IN CONTEXT:

- **Real wages**
- **Living wages**

DATABASE

Established in 2012

- Currently contains 900 collective agreements from 52 countries
- Agreements are coded, annotated (749 variables and nine main topics) and published in the national sites run by WageIndicator

WageIndicator.org

You Share, We Compare

PROBLEMS AND SOLUTION OFFERED BY TEXT AND DATA MINING (TDM)

PROBLEMS

- We can't have one annotator per language, so sometimes we need to be able to annotate agreements in languages we don't speak
- The annotator has to go through the text many times to find where one topic is addressed
- Annotation is very time consuming as texts are often very long

SOLUTION

A set of keywords (per language) would spot where one topic is addressed, so the annotator could just exclude the rest of the text and look for information in one area only.

WageIndicator.org

You Share, We Compare

HOW?

- All the selected clauses are in a .csv file, together with the relevant topic ID.
- We select one language where we have at least 30 agreements.
- We do text cleaning and lemmatization on 60% of the clauses, we write a script to isolate keywords that are more frequent in each topic.
- We test those keywords on the remaining 40% of our sample (cross validation).
- If the keywords work, we can enter them in our system to spot automatically the clauses where the topics are addressed, to speed up annotation.
- If possible, we will try to go deeper with questions to spot more specific answers.

Work Package 3.3. - Text and Data Mining

Keywords extraction from annotated collective agreements clauses to facilitate annotation of new collective agreements (using the Cobra system of the WageIndicator Collective Agreements Database - wageindicator.org/cbadatabase)

Starting point:
WageIndicator started in 2012 a Collective Agreements Database.
The collective agreement (CBA) text is uploaded in html format. H1 (title), H2 (section) and H3 (article) are assigned by the annotator.
Each CBA is annotated by answering to questions related to 9 topics and by selecting the part of text (clause) where the answer can be found.
Resulting file: .csv file with locale (eg EN_GH for Ghana), bind (id of the question), label (question), clause selected.

State of the art on 14th October 2019

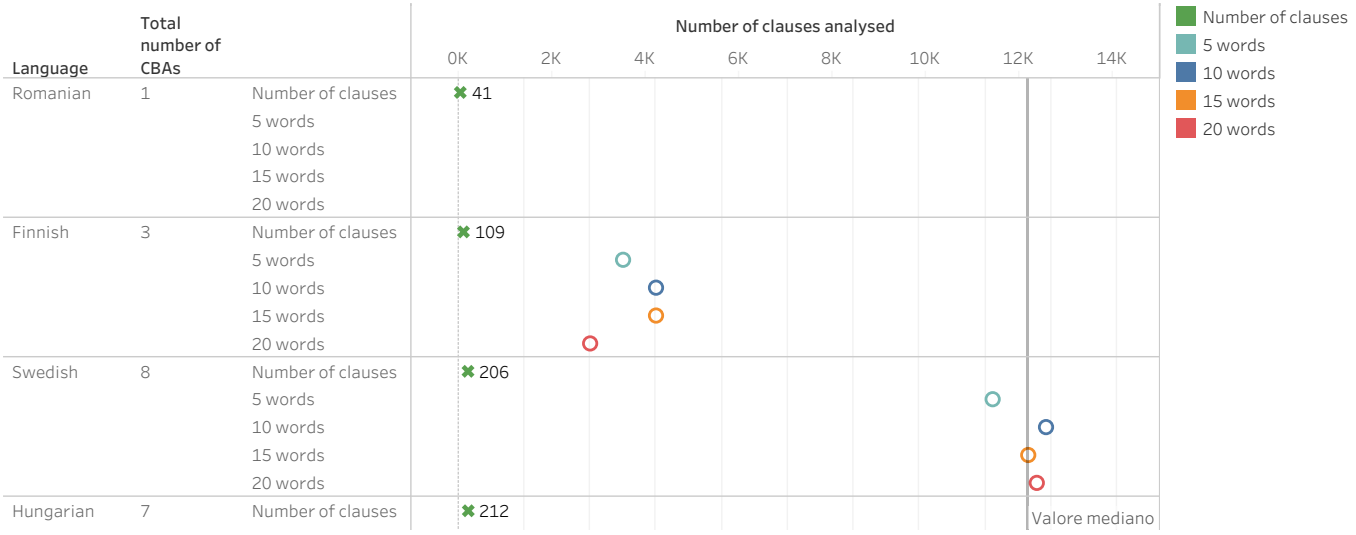
A Python script has been created:

Library used: NLTK (Natural Language ToolKit).
Tools used for lemmatization: 'WordNetLemmatizer' for English, 'SnowballStemmer' for other languages, 'stopwords' to remove stopwords.
Each clause is lemmatized (all downcased, stopwords removed, non-alphanumerics removed, one- and two-letter tokens removed, duplicate words removed, extra whitespace removed, replace numbers with string 'numeric', etc)
The clauses in each language are randomly split in 60% (training set) and 40% (testing set).
The script is run on the training set. A list of keywords is produced. The keywords are then tested against the testing set to check the correspondence (= reliability of the script).

At the moment we are trying to understand:
1. What is the minimum number of CBAs / clauses needed to have a decent percentage of reliability.
2. What is the right number of words to look for to have a better result (5, 10, 15 or 20).

```
tries = [x[0] for x in all_countries if x[1] == langu

eate a dataframe in which we can find all these count
 = df[df['country'].isin(countries)].copy()

t('\n\n', '{}: {} entries'.format(language, str(len(d
t('####################################')

ean all clauses
['cleaned_clause'] = data['clause'].apply(lambda x: c

uresets_training = []
uresets_testing = []
uresets_clauses_testing = []

erate over binds
bind in binds:
# collect all values for this bind
rows = data[data['bind'] == bind]
# if there are more
if len(rows) > 0:

    rows = rows.sample(frac=1)
    train_set, test_set = rows[:round(len(rows)*0.60)
```

Result of training - testing of CBAs keywords recognition (14th Oct 2019)



Observations

- We only have one agreement in Romanian
- The following languages are not supported by the Snowball stemmer (we need a lemmatizer that works) for these: Bahasa, Greek, Khmer, Croatian, Vietnamese, Maltese, Estonian, Polish, Czech, Slovak, Slovenian, Lithuanian, Bulgarian.

- At least 206 clauses (6-7 CBAs) are needed to have some results.
- The number of words can be customized in the script each time for a different language.
- 5 words are enough for languages with 67-203 CBAs (1751-6116 clauses), but not for English, where there are 401 CBAs (10135 clauses), and a minimum of 15 words is needed.
- At least 10 words are needed for languages with 7 to 11 CBAs (206 to 928 clauses).
- In Italy there are 28 CBAs but only 567 clauses because these CBAs have been added to the system but not annotated yet.

..

```
 1 import nltk
 2 import re
 3 import math
 4 import pandas as pd
 5 from nltk.stem import WordNetLemmatizer
 6 from nltk.stem.snowball import SnowballStemmer
 7 from nltk.corpus import stopwords
 8 import pdb
 9 #import ufal.udpipe
10
11 # cd /Users/dani/Documents/Lavoro/WageIndicator/
   SSHOC in terminal
12 # python3 cba_clauses.py
13 # sudo pip3 install pandas per installare il modulo
    pandas
14
15
16 lmtzr = WordNetLemmatizer()
17
18 # tokenize downcased description, remove stopwords
   and non alphanumeric stuff, remove one-letter
   tokens
19 # lemmatize the result (singular tokens), remove
   duplicate words
20 def clean(s, language='english'):
21
22     # lower the whole thing and remove whitespace
23     result = re.sub('\s+', ' ', str(s)).lower()
24
25     # tokenize, make sure to filter out all non-
   alphabetical and non-numerical stuff
26     result = [token for token in nltk.word_tokenize
   (result) if (token.isalpha() or token.isnumeric())]
27
28     # stem or lemmatize
29     if language == 'english':
30         result = [lmtzr.lemmatize(token) for token
   in result]
31     else:
32         stemmer = SnowballStemmer(language)
33         result = [stemmer.stem(token) for token in
   result]
34
35     # remove stopwords
```

```
36      result = [token for token in result if token
    not in stopwords.words(language)]
37
38      # replace numbers with strings that describe
    the type of number (small, medium, large)
39      result = ['<numeric>' if token.isnumeric() else
     token for token in result]
40
41      # remove 2 letter words or less (in french I )
42      result = [token for token in result if len(
    token) > 2]
43
44      # join into a string again
45      return ' '.join(result)
46
47
48 #check if the list of words can be found in clauses
49 def check_words_in_clause(clause, words):
50      clause_words = clause.split(' ')
51      features = {}
52      for word in words:
53          features['contains({})'.format(word)] = (
    word in clause_words)
54      return features
55
56
57 # read the csv file
58 # ================================================
59 # SUBSTITUTE THE PROPER PATH TO YOUR EXCEL DUMP!!!!
60 # ================================================
61 df = pd.read_csv('cba_clauses_dump_14102019.csv')
62
63 # sorted list of all binds
64 binds = sorted(list(df['bind'].unique()))
65
66 all_countries = [('argentina', 'spanish'), ('gin',
    'french'), ('ghana', 'english'), ('tgo', 'french'
    ), ('ben', 'french'),
67      ('mdg', 'french'), ('ner', 'french'), ('sen', '
    french'), ('uganda', 'english'), ('tanzania', '
    english'),
68      ('mozambique', 'portuguese'), ('malawi', '
    english'), ('kenya', 'english'), ('burundi', '
    french'),
```

```
69      ('rwanda', 'english'), ('ethiopia', 'english'
    ), ('south-africa', 'english'), ('zambia', '
    english'),
70      ('brazil', 'portuguese'), ('peru', 'spanish'),
71      ('costa-rica', 'spanish'), ('el-salvador', '
    spanish'), ('guatemala', 'spanish'), ('honduras',
    'spanish'),
72      ('colombia', 'spanish'), ('mexico', 'spanish'
    ), ('indonesia', None), ('vietnam', None), ('
    cambodia', None), ('pakistan', 'english'),
73      ('estonia', None), ('hungary', 'hungarian'), (
    'spain', 'spanish'), ('croatia', None), ('romania'
    , 'romanian'),
74      ('united-kingdom', 'english'), ('greece', None
    ), ('netherlands', 'dutch'), ('portugal', '
    portuguese'),
75      ('belgium', 'french'), ('czech-republic', None
    ), ('slovakia', None), ('bulgaria', None), ('
    zimbabwe', 'english'),
76      ('finland', 'finnish'), ('france', 'french'
    ), ('germany', 'german'), ('italy', 'italian'), ('
    sweden', 'swedish'),
77      ('lithuania', None), ('slovenia', None), ('
    austria', 'german'), ('denmark', 'danish'), ('
    lesotho', 'english')]
78
79
80 # get all languages
81 languages = list(set([x[1] for x in all_countries
    if x[1]]))
82 print(languages)
83
84 # override languages set
85 languages = ['english']
86
87 # initalise csv for output later
88 csv_results = pd.DataFrame()
89
90 # iterate over languages
91 for language in languages:
92
93     # get all countries which speak this language
94     countries = [x[0] for x in all_countries if x[
    1] == language]
```

```
 95
 96      # create a dataframe in which we can find all
    these countries
 97      data = df[df['country'].isin(countries)].copy
    ()
 98
 99      print('\n\n', '{}: {} entries'.format(language
    , str(len(data))))
100      print('####################################')
101
102      # clean all clauses
103      data['cleaned_clause'] = data['clause'].apply(
    lambda x: clean(x, language))
104
105      featuresets = []
106
107      # iterate over binds
108      for bind in binds:
109          # collect all values for this bind
110          rows = data[data['bind'] == bind]
111          # if there are more
112          if len(rows) > 0:
113
114              # cumulate these values
115              clauses = ' '.join(list(rows['
    cleaned_clause']))
116              # do a frequency count
117              freqs_raw = nltk.FreqDist(clauses.
    split(' '))
118
119              words = [x[0] for x in freqs_raw.
    most_common(15)]
120              clauses = rows['cleaned_clause']
121              for clause in clauses:
122                  features = check_words_in_clause(
    clause, words)
123                  featuresets = featuresets + ([(
    features, bind)])
124
125              # prepare data for csv
126              if hasattr(freqs_raw, 'most_common'):
127                  for x in freqs_raw.most_common(15
    ):
128                      temp = pd.DataFrame([[language
```

```
128 .replace('english','EN').replace('french','FR').
    replace('spanish','ES'),bind,x[0],x[1],len(rows),(
    math.ceil((float(x[1])/float(len(rows)))*100)/100
    )]],columns=['language','bind','word','occurrences
    ','clauses','avg_frequency'])
129                     csv_results = csv_results.
    append(temp, ignore_index=True)
130
131         else:
132             pass
133
134     csv_results.to_csv('csv_results_'+language+'.
    csv')
135
```