Danish resources

Finn Årup Nielsen

February 20, 2020

Abstract

A range of different Danish resources, datasets and tools, are presented. The focus is on resources for use in automated computational systems and free resources that can be redistributed and used in commercial applications.

Contents

1 Corpora

1.1 Wikipedia

There are several advantages with Wikipedia. The CC BY-SA license means that Wikipedia as a corpus can be used in commercial applications and distributed to third-parties. There is reasonably easy access to the data, either through the API available at https://da.wikipedia.org/w/api.php or by download of the full dump available from https://dumps.wikimedia.org/dawiki/. The relevant dump files with article texts follow a fixed naming pattern and the file for 20 November 2016 is called `dawiki-20161120-pages-articles.xml.bz2`.

One minor issue with the data in text mining applications is that the text is embedded with wiki markup where nested transclusions are possible with the slightly obscure templating language of MediaWiki. The Python module `mwparserfromhell`(the MediaWiki Parser from Hell) is one attempt to parse the wiki markup and usually do a sufficient job at extracting relevant text from the wikipage.

The Danish Wikipedia has, as of November 2016, more than 220.000 articles. Totally there are close to 750.000 pages on the Danish Wikipedia. This includes small pages, such as pages redirecting pages, discussion (talk) and user pages as well as other special pages. In text mining cases it is mostly the article pages that are relevant.

The use of Wikipedia in text mining applications is widespread [1].

1.2 Wikisource

Wikisource is a sister site to Wikipedia and contains primary source texts that are either in public domain or distributed under a license compatible with Creative Commons Attribution-Share Alike licence.

The Danish Wikisource claims to have over 2'000 source texts. The texts include fiction, poetry and non-fiction. A sizeable part of the works of H.C. Andersen is included.

Due to copyright, the major portion of Danish Wikisource are works in public domain where the author has been "sufficiently dead", i.e., dead more than 70 years. It means that the major part of the texts appear with capital letter for nouns, unusual words and old spelling, e.g., double a, "aa", instead of "å". For instance, "Han kjendte Skoven i dens deilige Foraars-Grønne kun derved, at Naboens Søn bragte ham den første Bøgegreen" is a sentence from a story by H.C. Andersen. Here "kjendte", "deilige" and "Bøgegreen" uses old spelling. This issue is similar for the ADL, Gutenberg and Runeberg resources listed below.

Wikidata may have link to a Danish Wikisource work. The linkage is, as of 2016, apparently not complete.

https://dumps.wikimedia.org/dawikisource has links to dumps of the Danish Wikisource. The November 2016 compressed article dump is only 12.2 MB. A text may be split across multiple pages and will need to be extracted and assembled, which is not a straightforward process. A tool for Wikisource text assembling for the Arabic Wikisource has been described [2].

The Python code below extracts the work via the MediaWiki API of the Danish Wikisource. The example is with *Mogens*, a shortstory by J.P. Jacobsen where the entire text is display on one single wikipage.

```
url = 'https://da.wikisource.org/w/api.php'
params = {'page': 'Mogens', 'action': 'parse', 'format': 'json'}
data = requests.get(url, params=params).json()
text = BeautifulSoup(data['parse']['text']['*']).get_text()
```

Some of the few works that are linked from Wikidata can be identified through the following SPARQL query on the Wikidata Query Service:

```
SELECT ?item ?itemLabel ?article WHERE {
  ?article schema:about ?item.
  ?article schema:isPartOf <https://da.wikisource.org/>.
  values ?kind { wd:Q7725634 wd:Q1372064 wd:Q7366 wd:Q49848 }
  ?item (wdt:P31/wdt:P279*) ?kind .
  SERVICE wikibase:label { bd:serviceParam wikibase:language "da,en". }
}
```

1.3 Wikiquote

Danish Wikiquote contains quotes. The quotes may be copyrighted, but due to their shortness they likely to fall under fair use. The number of citation collections in the Danish Wikiqoute is fairly limited. There is only 150 of these pages.

https://dumps.wikimedia.org/dawikiquote/ has links to the dumps of the Danish Wikiquote. The November 2016 article compressed dump is only 667 KB large.

1.4 ADL

Arkiv for Dansk Litteratur (ADL, Archive for Danish Literature) distributes digital texts from the site http://adl.dk. Most (if not all) of the texts are in public domain. ADL claims to have texts from 78 different authors. Authors include, e.g., Jeppe Aakjær, H.C. Andersen and Herman Bang. Each text has an individual URL.[1]

Though the texts are in public domain, ADL puts a restrictive license that prohibits redistribution of the texts they have digitized. This may hinder some text mining applications of the ADL data.

1.5 Gutenberg

Project Gutenberg makes digital texts available from https://www.gutenberg.org/. Project Gutenberg states having over 53'000 free books. An overview of the Danish works in the project is available at https://www.gutenberg.org/browse/languages/da. The entire corpus of Danish texts can be mirrored to local storage with a one-line command. As of 2016, there are 63 Danish works with around 23 million characters from more than 230'000 sentences. Some of the Project Gutenberg texts are currently linked from Wikidata via the P2034 property.[2]

[1] http://adl.dk/adl_pub/pg/cv/AsciiPgVaerk2.xsql?nnoc=adl_pub&p_udg_id=7&p_vaerk_id=872, for instance, downloads Herman Bang's novel *Tine*.

[2] Many, if not all of the linked works can be retrieved with the following SPARQL query on the

There may be an issue with copyright in Denmark. For a few of the works the copyright has expired in the United States, but not in Denmark. As of 2016, this is, e.g., the case with *Kongens Fald* by Johannes V. Jensen. *Et Aar* by Antonius (Anton) Nielsen may be another case.

1.6 Runeberg

Runeberg is a Gutenberg-inspired Swedish-based effort for digitizing public domain works with OCR and manual copy-editing. The digitized works are made available from the homepage http://runeberg.org/, where individual digital texts are downloadable, e.g., Hans Christian Ørsted's Aanden i Naturen is available from http://runeberg.org/aanden/. Some of the authors and works are linked from Wikidata by the specialized properties P3154 and P3155.

Not all works on Runeberg are correctly transcribed. OCR errors consitute a problem and indeed a major problem for works with gothic script that are not copy-edited.

Words from Runeberg has been used in a Swedish text processing system [3].

1.7 Europarl

The natural language processing toolkit for Python, NLTK [4], makes easy to access and download of a range of language resources. Among its many resources is a European Parliament multilingual corpus where there is a Danish part. This part contains 22'476 "sentences", 563'358 tokens and 27'920 unique tokens. There are no labels. The sentence tokenization is not done well: many sentences are split due to punctuations around "hr." and "f.eks.". NLTK methods make it easy to load the word list into a Python session.

`europarl-da-sentiment` available from `https://github.com/fnielsen/europarl-da-sentiment` contains a sentiment labeling of a few of the sentences from the European Parliament corpus.

1.8 Leipzig Corpora Collection

The Leipzig Corpora Collection by Abteilung Automatische Sprachverarbeitung, Universität Leipzig is a collection of monolingual corpora for several languages including Danish [5]. Corpora of a range of sizes from different crawls are available from http://corpora2.informatik.uni-leipzig.de/download.html. The text files contain one independent sentence on each line. The largest Danish file has 1 million sentences, while the smallest has 10'000 sentences. The sentence tokenization is not alway correct. Some

Wikidata Query Service:

```
select ?work ?workLabel ?authorLabel where {
  ?work p:P2034 ?gutenberg_statement .
  ?gutenberg_statement pq:P407 wd:Q9035 .
  optional { ?work wdt:P50 ?author . }
  service wikibase:label { bd:serviceParam wikibase:language 'da' }
}
```

[3]http://wortschatz.uni-leipzig.de/use.html

sentences are duplicates. The downloadable corpora are licensed under CC BY.[3]

`lcc-sentiment`, available from https://github.com/fnielsen/lcc-sentiment, annotates a few of the Danish sentence for sentiment.

1.9 Danish Dependency Treebank

Danish Dependency Treebank is distributed under the GNU Public License from http://www.buch-kromann.dk/matthias/ddt1.0/ and "consists of 536 Parole texts, consisting of 5.540 sentences, 100.195 words" [6, 7]. The data has been used in research [8, 9].

1.10 Retsinformation

Laws and regulations are not covered by copyright in Denmark.[4] The Danish laws are digitally available online from https://www.retsinformation.dk/. Function to download and handle this corpus is available in the retsinformation.py module of the *Dasem* Python package. One snapshot of the corpus is also included in Danish Gigaword corpus collection.

1.11 Other resources

Det Danske Sprog- og Litteraturselskab distributes several large Danish corpora: Korpus 90, Korpus 2000 [10] and Korpus 2010. These corpora have been POS-tagged and lemmatized. Password-protected files are available for download at http://korpus.dsl.dk/resources.html. They are not directly available for commercial applications and cannot be redistributed.[5]

CLARIN-DK is a Danish effort to collect Danish texts and other forms of resources and make them available [11]. While some of its material originates from "free" sources (Folketinget and Wikipedia), other parts are taken from texts covered by copyright and with limited licensing.

DSim is a small corpus with 585 sentences that have been aligned for text simplification research [12]. A few more Danish corpora are mentioned by [10].

At `https://visl.sdu.dk/corpus_linguistics.html`, the *Visual Interactive Syntax Learning* project of the University of Southern Denmark lists several Danish corpora, including the already mentioned Europarl and Wikipedia, the latter in a small 2005 version with only 3.7 million words. Large corpora are *Information* with 80 million words and *Folketinget* with 7 million words. The corpora is apparently not immediately available for download.

[4]The specific law (*Ophavsretsloven*, 1144, 2014-10-23) states in Danish "Offentlige aktstykker

§ 9. Love, administrative forskrifter, retsafgørelser og lignende offentlige aktstykker er ikke genstand for ophavsret. Stk. 2. Bestemmelsen i stk. 1 gælder ikke for værker, der fremtræder som selvstændige bidrag i de i stk. 1 nævnte aktstykker. Sådanne værker må dog gengives i forbindelse med aktstykket. Retten til videre udnyttelse afhænger af de i øvrigt gældende regler."

[5]The conditions state: "The language resources may only be used for the indicated purpose(s) and must not be copied or transferred to a third party. The language resources must not without special prior arrangement be used commercially or form part of a commercial product." at

2 Lexical resources

2.1 DanNet

DanNet is the Danish wordnet and contains synsets, relation and sentences as examples for the items [13]. It is freely available from http://wordnet.dk/. It has 57'459 word forms, 65'670 synsets and 236'861 relations.[6] The information may be read into a SQL database and queried.

Part of DanNet is part of the multilingual *Open Multilingual WordNet* [14], see http://compling.hss.ntu.edu.sg/omw/. The current number of Danish merged synsets is 4'476. The wordnet methods in NLTK are able to search this merged corpus, including the Danish part. The code below identifies the WordNet synset for the Danish word for dog ("hund"):

```
>>> from nltk.corpus import wordnet as wn
>>> 'dan' in wn.langs()
True
>>> wn.synsets('hund', lang='dan')
[Synset('dog.n.01')]
```

As the name implies *Extended Open Multilingual Wordnet* [15] extends the *Open Multilingual Wordnet*. Data from Wiktionary and the Unicode Common Locale Data Repository are used to extend the Danish part considerably to 10'328 synsets, see http://compling.hss.ntu.edu.sg/omw/summx.html.

DanNet is used in the http://www.andreord.dk/ webservice, where users can search on words, select synsets and view relations between synsets.

2.2 Wiktionary

Wiktionary is a sister site to Wikipedia and it contains lexical information. The Danish Wiktionary at https://da.wiktionary.org contains over 35'000 words. Not all of these are Danish words. The category system of the Danish Wikipedia lists over 9'000 Danish nouns.[7] This noun count includes the lemma form and derived forms.

The content of Wiktionary is represented in a standard MediaWiki instance, but makes extensive use the MediaWiki templating to represent the structured lexical information, — and thus requires specialized parsing.

http://korpus.dsl.dk/conditions.html.

[6]Using dasem:

```
>>> from dasem.dannet import Dannet
>>> dannet = Dannet()
>>> len(dannet.db.query('select␣w.form␣from␣words␣w;'))
57459
>>> len(dannet.db.query('select␣s.synset_id␣from␣synsets␣s;'))
65670
>>> len(dannet.db.query('select␣*␣from␣relations␣r;'))
236861
```

[7]A listing of the Danish nouns is available via the category page on the Danish Wiktionary: https://da.wiktionary.org/wiki/Kategori:Substantiver_på_dansk

2.3 Wikidata

Wikidata at https://www.wikidata.org is the structured data sister to Wikipedia. It contains almost 55 million items described by labels, aliases, short descriptions and properties with values, qualifiers and references. Many of the items correspond to articles in the different language versions of Wikipedia, Wikiversity, Wikibooks, Wikinews, Wikiquote, Wikisource, Wikivoyage or Wikimedia Commons. Support for lexical information was enabled in 2018, so Wikidata now contains a small set of lexemes and inflected forms. The lexeme may be linked to senses and further on to the ordinary Wikidata items. As of February 2020, Wikidata had over 4,400 Danish lexemes and over 17,000 Danish forms.[8]

Apart from the dumps available at https://dumps.wikimedia.org/wikidatawiki/ and standard MediaWiki API access from https://www.wikidata.org/w/api.php the data in Wikidata is also available in SPARQL result representation with a SPARQL endpoint accessible from https://query.wikidata.org/, — the Wikidata Query Service. The endpoint allows users to query for Danish labels, e.g., conditioned on properties.

The lexical information in Wikidata can be browsed via the Ordia Web application at https://tools.wmflabs.org/ordia/. For an overview of Danish lexemes, see https://tools.wmflabs.org/ordia/language/Q9035.

2.4 OmegaWiki

OmegaWiki at http://www.omegawiki.org is a collaborative multilingual lexical resource. A mysql database dump is available and it contains several thousands Danish words, mostly in lemma form.[9] The semantics of words are structured around a language-independent concept called "defined meaning" and the different defined meanings may be linked. For instance, "bager" is linked via the defined meaning "baker", so baker can be determined to be a kind of profession and that a baker is active in the food industry and works in a supermarket or bakery.

2.5 Retskrivningsordbogen

Retskrivningsordbogen (RO) is the official Danish spelling dictionary assembled by Dansk Sprognævn. RO is available in various digital formats under its own special license. As of January 2018, the word lists is freely available, but cannot be used in independent dictionaries, — only as an integrated part of language technological products such as games and search engines.

The smallest list contains only the lexeme and its word class, — not the inflected forms, and the 2012 version has 64896 lexemes. It is available from https://dsn.dk/retskrivning/om-retskrivningsordbogen/ro-elektronisk-og-som-bog. A longer word list contains the inflected forms, and a XML-formatted dictionary also has grammatical information, hyphenation and usage examples.

[8]Statistics for Danish and other languages are available in Ordia [16] at https://tools.wmflabs.org/ordia/language/.

[9]The number of Danish words in OmegaWiki can be counted from the database dump file:

```
select count(*) from uw_expression where language_id = 103;
```

europæisk	eurovision
europæiske	eurovisionen
EuroCity-toget	eurovisionens
eurotilhængere	eurovisioner
eurocheck	eurovisionerne
eurochekenes	eurovisionernes
eurovalget	eurovisioners
	eurovisions

Table 1: Excerpt from *The Comprehensive Danish Dictionary.*

2.6 The Comprehensive Danish Dictionary

The Comprehensive Danish Dictionary (Den store danske ordliste, DSDO) is a large list of Danish words published by Skåne Sjælland Linux User Group and licensed under GNU General Public License, version 2 or later. It was available from `http://da.speling.org/` (but apparently no longer) and is contained in Debian-derived systems as the package `aspell-da`. The `aspell` program can manipulate the distributed binary file, and the command for a dump the content to a text file with one word per line reads

```
aspell -l da dump master > danish-words.txt
```

One version of the file contains 313'148 words. The words may come in inflected forms, including genitive forms, see Table 1 for an excerpt. Not all forms may be available, see, e.g., "eurotilhængere" in the table where the 7 other forms are missing ("eurotilhænger", "eurotilhængeren", "eurotilhængerne", "eurotilhængers", etc.). There may even be spelling variations which are questionable, e.g., "eurochekenes".

2.7 Other lexical resources

There is a Danish word list associated with Runeberg. It contains 12'958 words and is available at http://runeberg.org/words/. The words in the list are mostly spelt as before the language reform, i.e., with double-a and initial capital letters for nouns.

A number of resources from *Det Danske Sprog- og Litteraturselskab* (DSL) are available for download at http://korpus.dsl.dk/resources.html. These corpora may not be part of a commercial product without prior agreement with DSL, see http://korpus.dsl.dk/-conditions.html. ePAROLE from DSL contains sentences with over 300'000 words. Each word is characterized by a text identifier, a sentence identifier, the word form, the lemma of the word, the POS tag and several "markers". These markers indicate singular/plural, definiteness, case, gender, tense, voice, and other aspects.

NLTK has a small list of 94 Danish stopwords. They may be loaded by:

```
>>> from nltk.corpus import stopwords
>>> words = stopwords.words('danish')
>>> len(words)
94
```

2.8 Wikidata examples with medical terminology extraction

An example of the use of Wikidata in a lexical applications is a SPARQL query on disease with a Danish label, where the diseases are identified by their ICD-9 and ICD-10 linkage:

```
select distinct ?disease ?disease_label ?icd9 ?icd10 where {
  ?disease wdt:P493|wdt:P494 ?icd .
  optional { ?disease wdt:P493 ?icd9 . }
  optional { ?disease wdt:P494 ?icd10 . }
  ?disease rdfs:label|skos:altLabel ?disease_label .
  filter (lang(?disease_label) = 'da')
} order by ?icd9 ?icd10
```

With the above query the Wikidata Query Service returns just over 3'000 labels. Here it should be noted that Wikidata has no guaranty of correctness or completeness of the information. Similar queries for German or Swedish labels return over 23'000 labels and 10'000, respectively.

3 Natural language processing tools

3.1 NLTK

NLTK is a natural language processing (NLP) toolkit written in Python [4, 17]. There is some support for the Danish language: Sentence tokenization (sentence detection) and word stemming.

3.2 Polyglot

Polyglot is a comparatively new Python package with multilingual natural language processing capabilities, and among the many language supported is also Danish [18]. A Python package is available in the Python Package Index and thus installable with

```
pip install polyglot
```

The package will need data files to operate. These files are downloaded under the directory `~/polyglot_data`. From the command-line the data may be downloaded with commands such as

```
polyglot download embeddings2.da
polyglot download pos2.da
polyglot download morph2.da
```

Polyglot implements a range of natural language processing methods: language detection, tokenization, word embedding operations, POS-tagging, named entity extraction, morphological analysis, transliteration and sentiment analysis. These methods are not implemented for all languages.

Polyglot is documented at https://polyglot.readthedocs.io. Development version of polyglot takes place at GitHub from https://github.com/aboSamoor/polyglot

3.3 spaCy

spaCy is a newer Python NLP toolkit available from https://spacy.io. It has support, e.g., for English and some support for Swedish and Norwegian Bokmål, see https://spacy.io/docs/api/language-models. As of November 2017, it has limited support for Danish, though the "xx" multilingual model and models trained for other languages may work for some tasks.

3.4 Apache OpenNLP

Apache OpenNLP is a natural language processing toolkit which also can be used for Danish. The command-line program `opennlp` and the supporting Java libraries are available for download at https://opennlp.apache.org/. Pre-trained Danish models are found at http://opennlp.sourceforge.net/models-1.5/. These models enables Danish sentence detection, tokenization and part-of-speech (POS) tagging

3.5 Centre for Language Technology

Centre for Language Technology (Center for Sprogteknologi, CST) at the University of Copenhagen has several tools for Danish natural language processing. Their online tools are displayed at http://cst.ku.dk/english/vaerktoejer/ These include keyword extractor, lemmazier, name recognizer and POS tagger among others. Several command-line tools are available on Github, e.g., a lemmatizer.[10]

3.6 StanfordNLP

StanfordNLP described at https://stanfordnlp.github.io/stanfordnlp/ is a software package with university dependency parsing and interface to Stanford's CoreNLP. The package and its Danish model can be installed with

```
$ pip install stanfordnlp
$ python
>>> import stanfordnlp
>>> stanfordnlp.download('da')
```

3.7 Other libraries

DKIE for Danish tokenization, POS-tagging, named entity recognition and temporal expression annotation was reported in [19].

[10]Installation of cstlemma may be performed with:

```
$ git clone https://github.com/kuhumcst/cstlemma.git
$ cd cstlemma/doc/
$ bash makecstlemma.bash
```

4 Natural language processing

4.1 Language detection

Compact Language Detector 2 (CLD2) maintained by Dick Sites is a language detection Apache-licensed library available from https://github.com/CLD2Owners/cld2. It supports over 80 languages, including Danish. There is Python binding with the library called `pycld2`.

```
>>> from pycld2 import detect
>>> detect('Er␣du␣ok?')
(False, 10, (('Unknown', 'un', 0, 0.0), ('Unknown', 'un', 0, 0.0),
('Unknown', 'un', 0, 0.0)))
>>> detect('Hvordan␣kan␣man␣i␣det␣hele␣taget␣finde␣sproget?')
(True, 48, (('DANISH', 'da', 97, 1220.0), ('Unknown', 'un', 0, 0.0),
('Unknown', 'un', 0, 0.0)))
```

The `pycld2` Python library is used as part of the `polyglot` Python library.

```
>>> from polyglot.detect import Detector
>>> detector = Detector('Er␣du␣ok?')
>>> detector.languages[0].name
u'Dutch'
>>> detector = Detector('Hvordan␣kan␣man␣i␣det␣hele␣taget␣finde␣sproget?')
>>> detector.languages[0].name
u'Danish'
```

4.2 Sentence tokenization

Danish sentence tokenization is available in NLTK with the `sent_tokenize` function:

```
>>> nltk.sent_tokenize(('Hvordan␣går␣det␣f.eks.␣med␣Hr.␣Jensen?␣'
                        'Er␣han␣blevet␣bedre?'), language='danish')
['Hvordan␣går␣det␣f.eks.␣med␣Hr.␣Jensen?', 'Er␣han␣blevet␣bedre?']
```

This function loads a pretrained tokenization model and use it for tokenization of a given text. The model can also be loaded explicitly:

```
>>> tokenizer = nltk.data.load('tokenizers/punkt/danish.pickle')
>>> tokenizer.tokenize(('Hvordan␣går␣det␣f.eks.␣med␣Hr.␣Jensen?␣'
                        'Er␣han␣blevet␣bedre?'))
['Hvordan␣går␣det␣f.eks.␣med␣Hr.␣Jensen?', 'Er␣han␣blevet␣bedre?']
```

The model has been trained on Danish newspaper articles from Berlingske by Jan Strunk.[11] Using the non-Danish sentence tokenizer may yield suboptimal results. For instance, the default English tokenizer produces a wrong tokenization for the example text:

```
>>> nltk.sent_tokenize(('Hvordan␣går␣det␣f.eks.␣med␣Hr.␣Jensen?␣'
                            'Er␣han␣blevet␣bedre?'))
['Hvordan␣går␣det␣f.eks.', 'med␣Hr.', 'Jensen?', 'Er␣han␣blevet␣bedre?']
```

[11]See README of the data. This is usually installed at `~/nltk_data/tokenizers/punkt/README`.

An example with a command-line opennlp-based Danish sentence detection can look like this:

```
$ echo "Hej,␣der.␣Hvor␣er␣du␣henne?" \
       "Har␣du␣f.eks.␣husket␣Hr.␣Hansen" \
       "og␣en␣ph.d.-studerende?" > text.txt
$ opennlp SentenceDetector da-sent.bin < text.txt > sentences.txt
$ cat sentences.txt
Hej, der.
Hvor er du henne?
Har du f.eks. husket Hr. Hansen og en ph.d.-studerende?
```

Here the three sentences are found. The algorithm may fail on the name "Hr. Hansen" if the last part of the sentence is left out.

Tokenization can be performed with a simple algorithm

```
$ opennlp SimpleTokenizer < sentences.txt > tokens.txt
$ cat tokens.txt
Hej , der .
Hvor er du henne ?
Har du f . eks . husket Hr . Hansen og en ph . d . - studerende ?
```

Here there are several errors. A trained model performs somewhat better:

```
$ opennlp TokenizerME da-token.bin  < sentences.txt > tokens.txt
$ cat tokens.txt
Hej , der .
Hvor er du henne ?
Har du f.eks. husket Hr . Hansen og en ph.d.-studerende ?
```

4.3 Stemming

Danish stemming is available in NLTK via the Snowball stemmer:

```
>>> from nltk.stem.snowball import DanishStemmer
>>> stemmer = DanishStemmer()
>>> stemmer.stem('luften')
'luft'
>>> stemmer.stem('maler')
'mal'
```

The Danish Snowball stemming algorithm is also available in another Python module called `PyStemmer` and once installed it may be called with

```
>>> import Stemmer
>>> stemmer = Stemmer.Stemmer('danish')
>>> stemmer.stemWord('luften')
'luft'
>>> stemmer.stemWords(['maler', 'luften'])
['mal', 'luft']
```

The standalone Snowball stemming program is available from links at http://snowballstem.org/.

4.4 Lemmatization

Lemmy is a Python-based lemmatizer for Danish, see, https://github.com/sorenlind/lemmy. A Python session with Lemmy may read:

```
>>> import lemmy
>>> lemmatizer = lemmy.load("da")
>>> lemmatizer.lemmatize("", "containerskibenes")
['containerskib']
```

Here the plural definite s-genitive inflection has been striped from the compound word. Lemmy has been trained of Dansk Sprognævn's full word list and the Danish Dependency Treebank.

4.5 Decompounding

Polyglot will split a word into morphemes, effectively decompound and remove inflection. After installation of the Danish model with `polyglot download morph2.da`, a morpheme decomposition functionality is available in the `Word` class:

```
>>> from polyglot.text import Word
>>> Word('totalomkostninger').morpheme
WordList(['total', 'omkostning', 'er'])
>>> Word('investeringsforvaltningsholdingvirksomhedernes').morphemes
WordList(['investering', 's', 'forvaltning', 's', 'hold', 'ing',
          'virksomhed', 'erne', 's'])
```

It may be necessary to set the language explicit to obtain good performance:

```
>>> Word('politistation').morphemes
Detector is not able to detect the language reliably.
WordList(['pol', 'it', 'ist', 'ation'])
>>> Word('politistation', language='da').morphemes
WordList(['politi', 'station'])
```

Eckhard Bick's online tools in connection with *Visual Interactive Syntax Learning*, see, e.g., https://visl.sdu.dk/visl/da/parsing/automatic/complex.php, are able to perform decompounding, so that, e.g., *investeringsforvaltning* is recognized as consisting of the words investering+forvaltning.

A small dataset of over 1,300 compounds are available in Dasem at https://github.com/fnielsen/dasem/blob/master/dasem/data/compounds.txt. They are used as the basis for simple lookup-based decompounding in Dasem. A command-line-based decompounding with Dasem can be done with:

```
$ python -m dasem.text decompound "investeringsforvaltning"
investering forvaltning
```

Wikidata has a small number of Danish lexemes, including compounds and their parts. A SPARQL query for the Wikidata Query Service at https://query.wikidata.org can identify the parts of the compound:

```
SELECT * {
  BIND("investeringsforvaltningen"@da AS ?compound)
  ?lexeme ontolex:lexicalForm / ontolex:representation ?compound ;
          wdt:P5238 ?part .
  ?part wikibase:lemma ?lemma .
}
```

The query results in 3 rows corresponding to the compound parts *investering*, *-s-*, and *forvaltning*. Note that *investeringsforvaltningen* is in the definite inflection, while the result is returned as lexemes and lemmas without inflection.

FastText, with its handling of character n-grams, may in some applications make decompounding not necessary.

4.6 Part-of-speech tagging

Polyglot has a POS-tagging.

```
>>> from polyglot.text import Text
>>> blob = "Hej,␣der.␣Hvor␣er␣du␣henne?␣Har␣du␣f.eks.␣husket␣Hr.␣Hansen"
>>> text = Text(blob)
>>> text.pos_tags
[(u'Hej', u'INTJ'), (u',', u'PUNCT'), (u'der', u'PART'),
 (u'.', u'PUNCT'), (u'Hvor', u'ADV'), (u'er', u'VERB'),
 (u'du', u'PRON'), (u'henne', u'VERB'), (u'?', u'PUNCT'),
 (u'Har', u'SCONJ'), (u'du', u'PRON'), (u'f.eks', u'ADV'),
 (u'.', u'PUNCT'), (u'husket', u'ADJ'), (u'Hr', u'PROPN'),
 (u'.', u'PUNCT'), (u'Hansen', u'PROPN')]
```

Here there is automatic language detection involved.

In Apache OpenNLP, there are presently two trained models for part-of-speech (POS) tagging, — one that uses the `da-pos-maxent.bin` pre-trained model:

```
$ opennlp POSTagger da-pos-maxent.bin  < tokens.txt > tagged.txt
$ fold -w 60 -s tagged.txt
Hej_NP ,_XP der_U ._XP
Hvor_RG er_VA du_PP henne_RG ?_XP
Har_VA du_PP f.eks._RG husket_VA Hr_NP ._XP Hansen_NP og_CC
en_PI ph.d.-studerende_AN ?_XP
```

and another that uses the `da-pos-perceptron.bin` pre-trained model:

```
$ opennlp POSTagger da-pos-perceptron.bin  < tokens.txt > tagged.txt
$ fold -w 60 -s tagged.txt
Hej_NP ,_XP der_U ._XP
Hvor_RG er_VA du_PP henne_XS ?_XP
Har_VA du_PP f.eks._RG husket_VA Hr_NP ._XP Hansen_NP og_CC
en_PI ph.d.-studerende_NC ?_XP
```

CST has a version derived from Eric Brill's Part Of Speech tagger. CST's version is availabel from GitHub at https://github.com/kuhumcst/taggerXML. Web services demonstrating its capabilities are running from http://ada.sc.ku.dk/tools/ and http://ada.sc.ku.dk/online/pos_tagger/. Their `taggerXMLs` program requires a POS-tagged lexicon to operate.

With StanfordNLP:

```
>>> import stanfordnlp
>>> nlp = stanfordnlp.Pipeline(lang='da')
>>> text = "Hej,␣der.␣Hvor␣er␣du␣henne?␣Har␣du␣f.eks.␣husket␣Hr.␣Hansen"
>>> doc = nlp(text)
>>> [(word.text, word.upos) for sentence in doc.sentences
                              for word in sentence.words]
[('Hej', 'INTJ'), (',', 'PUNCT'), ('der', 'PRON'), ('.', 'PUNCT'),
 ('Hvor', 'ADV'), ('er', 'AUX'), ('du', 'PRON'), ('henne', 'ADV'),
 ('?', 'PUNCT'), ('Har', 'AUX'), ('du', 'PRON'), ('f.eks.', 'ADV'),
 ('husket', 'VERB'), ('Hr.', 'PROPN'), ('Hansen', 'PROPN')]
```

4.7 Dependency parsing

Danish dependency parsing is implement in the StanfordNLP Python package. With the package and its Danish model installed, sentences can readily be analyzed:

```
>>> import stanfordnlp
>>> nlp = stanfordnlp.Pipeline(lang='da')
>>> doc = nlp("Den␣lille␣mand␣sovser␣ordentligt␣rundt␣i␣kagen")
>>> doc.sentences[0].print_dependencies()
('Den', '3', 'det')
('lille', '3', 'amod')
('mand', '4', 'nsubj')
('sovser', '0', 'root')
('ordentligt', '4', 'advmod')
('rundt', '4', 'obl:loc')
('i', '8', 'case')
('kagen', '6', 'obl')
```

The part-of-speech tags and the grammatical features for the individual words are also available:

```
>>> doc.sentences[0].words[2]
<Word index=3;text=mand;lemma=mand;upos=NOUN;xpos=_;
 feats=Definite=Ind|Gender=Com|Number=Sing;governor=4;
 dependency_relation=nsubj>
```

4.8 Sentiment analysis

Danish sentiment analysis is available with the AFINN Danish word list. An initial English word list has been extended and translated to Danish. Both the Danish and the English version of AFINN associate individual words and phrases with a value (valence) between -5 and $+5$ where -5 indicates strong negative sentiment and $+5$ strong positive sentiment. The Danish word list was constructed from an initial Google Translate translation, followed by a manual inspection and editing and further extension. The sentiment for a text, e.g., a sentence, a tweet or a document may be computed as, e.g., the average or the sum of the individual word and phrases. Various other features may be computed, e.g., a value for ambivalence and separate values for positivity or negativity.

It is used in the `afinn` Python module available in the Python Package Index and on https://github.com/fnielsen/afinn. This module may perform the matching of a text to the word list in two ways:

1. An initial word tokenization followed by a lookup in the word list dictionary. This method will not identify phrases in the “word” list.
2. Direct matching with a regular expression. This method can identify phrases.

An application of the word list within Python can look like this:

```
>>> from afinn import Afinn
>>> afinn = Afinn(language='da')
>>> afinn.score('Hvis␣ikke␣det␣er␣det␣mest␣afskyelige␣flueknepperi...')
-6.0
```

Together with the word lists in `afinn` is a list of emoticon associated with a score. The Python module can combine the word and emoticon sentiment analysis in one process. The below code finds and scores the smiley emoticon in the end of the sentence:

```
>>> from afinn import Afinn
>>> afinn = Afinn(language='da', emoticons=True)
>>> afinn.score('Mon␣ikke␣han␣kommer␣i␣morgen␣:)')
2.0
```

The below code shows an example with sentiment scoring of multiple sentence with data taken from the Danish part of the European Parliament corpus in NLTK:

```
from afinn import Afinn
from nltk.corpus import europarl_raw

afinn = Afinn(language='da')
sentences = ["␣".join(wordlist) for wordlist in
             europarl_raw.danish.sents()]
scored_sentences = [(afinn.score(s), s) for s in sentences]
print(sorted(scored_sentences)[0][1])
```

The sentiment scored sentences are sorted and the most negative-scored sentence is shown. The result is this sentence:

> Situationen er alvorlig , eftersom der i dag inden for selve Den Europæiske Union er en tydelig sammenhæng mellem arbejdsløshed og fattigdom , som det påvises af den meget bekymrende kendsgerning , at arbejdsløsheden i gennemsnit berører 23,7 % af de regioner , der er hårdest ramt af dette problem , og som samtidig er fattige regioner , mens der i de 25 regioner , der har mindst arbejdsløshed , og som er de rigeste , er en arbejdsløshed på under 4 % .

While the English version of the word list has been validated with the initial articles [20] and several later papers, the Danish AFINN has so far had no major evaluation. A small evaluation has been performed with the `europarl-da-sentiment` sentiment corpus. The result is displayed in Table 2. Here the three-class accuracy of the particular limited data set reached 68%.

Several third-party developers has utilized the open AFINN word list and implemented sentiment analysis version in JavaScript and Perl. These implementation focus on the

afinn valence	-1	0	1
-1	22	16	5
0	3	30	7
1	1	6	29

Table 2: Three-class confusion matrix for `europarl-da-sentiment` afinn sentiment analysis. The rows correspond to manual labels, while the columns are afinn scores. The code for this particular computation is available as a Jupyter Notebook on Github repository associated with `europarl-da-sentiment`.

English version of AFINN, but it might be relatively easy to change the word list to the Danish version of AFINN.

The `polyglot` Python package also contains Danish sentiment analysis [21]. A small preliminary evaluation against a labeled Europarl corpus, `europarl-da-sentiment` showed that the sentiment analysis of the `polyglot` package does not perform as well as `afinn`: a three-class accuracy of 55% versus 68%.

4.9 Semantics

Digital Scholarship Labs at Statsbiblioteket in Aarhus maintains a webservice with a word2vec model trained on texts from Danish newspapers. The webservice is available from http://labs.statsbiblioteket.dk/dsc/.

4.9.1 FastText

FastText is a word and n-gram embedding method and program from Facebook Research. It also includes supervised learning [22, 23]. Pretrained embedding models based on Wikipedias are available for a number of languages, including Danish, see `https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md`. FastText has a command-line interface. Third-party packages, *fasttext* and *Gensim*, enable access through Python. A Gensim-fastText session with Facebook Research's pretrained Danish model may look like the following:

```
>>> from gensim.models.wrappers.fasttext import FastText
>>> model = FastText.load_fasttext_format('wiki.da')
>>> for word, score in model.most_similar('sjov', topn=7):
...     print(word)
sjovt
sjove
sjovere
sjoveste
pigesjov
hyggesnakke
barnlig
```

As fastText handles multiple n-grams it may also—with varying degree of success—be used to embed sentences with no further preprocessing:

```
>>> sentence = "der␣er␣tale␣om␣landbrug␣og␣maskinstation"
>>> for word, score in model.most_similar(sentence, topn=7):
...     print(word)
maskinstation
landbrugsmaskiner
landbrugsvirksomheder
jordbrugsmaskiner
landbrugsvirksomhed
maskinsnedkeri
industripark
```

Out-of-vocabulary misspellings are also handled:

```
>>> word = 'bankasistent'
>>> word in model.wv.vocab
False
>>> for word, score in model.most_similar(word, topn=7):
...     print(word)
bankassistent
sparekasseassistent
butiksassistent
bankas
banka
toneassistent
husassistent
```

4.9.2 Dasem

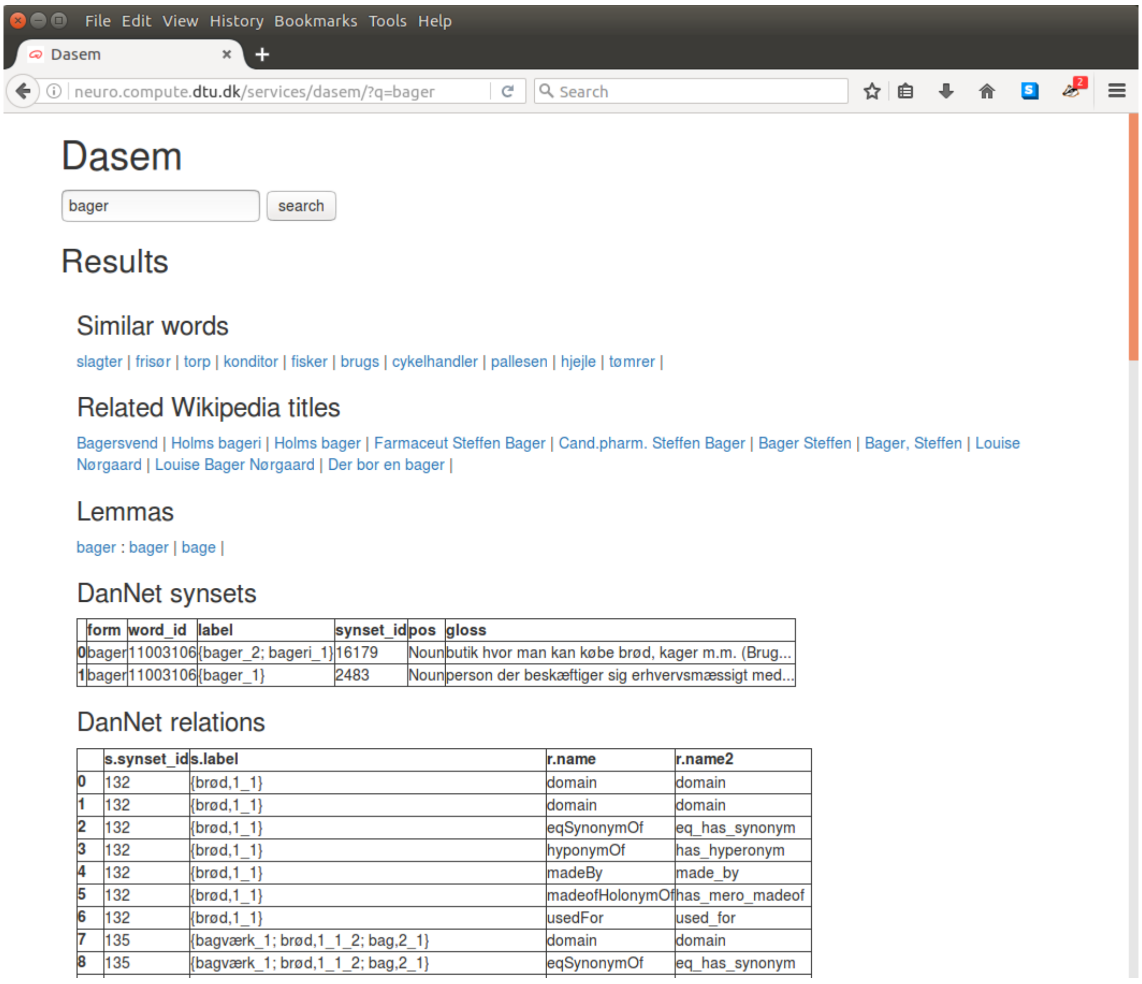
The `dasem` Python module attempts to assemble various methods for Danish semantic analysis. It is available from https://github.com/fnielsen/dasem. The current resources that form the basis for `dasem` are the Danish Wikipedia, Wiktionary, Danish part of Project Gutenberg, DanNet and ePAROLE. Semantic relatedness for Danish words and phrases are implemented and uses the Explicit Semantic Analysis (ESA) method [24] or the word2vec approach via the implementation in Gensim [25].

The Python code below shows an example of related word to the Danish word "bil" (car) with the Wikipedia-based Gensim word2vec approach:

```
>>> from pprint import pprint
>>> from dasem.wikipedia import Word2Vec
>>> w2v = Word2Vec()
>>> pprint(w2v.most_similar('bil')[:4])
[(u'lastbil', 0.7803581953048706),
 (u'motorcykel', 0.7234832048416138),
 (u'cykel', 0.7216866612434387),
 (u'vogn', 0.7213534116744995)]
```

A pretrained and stored model is read during the instantiation. The Danish Wikipedia XML dump needs to be downloaded in advance for the training to succeed.

```
>>> from dasem.semantic import Semantic
>>> from numpy import around
```


File Edit View History Bookmarks Tools Help
Dasem
neuro.compute.dtu.dk/services/dasem/?q=bager
Search

Dasem

bager search

Results

Similar words

slagter | frisør | torp | konditor | fisker | brugs | cykelhandler | pallesen | hjejle | tømrer |

Related Wikipedia titles

Bagersvend | Holms bageri | Holms bager | Farmaceut Steffen Bager | Cand.pharm. Steffen Bager | Bager Steffen | Bager, Steffen | Louise Nørgaard | Louise Bager Nørgaard | Der bor en bager |

Lemmas

bager : bager | bage |

DanNet synsets

	form	word_id	label	synset_id	pos	gloss
0	bager	11003106	{bager_2; bageri_1}	16179	Noun	butik hvor man kan købe brød, kager m.m. (Brug...
1	bager	11003106	{bager_1}	2483	Noun	person der beskæftiger sig erhvervsmæssigt med...

DanNet relations

	s.synset_id	s.label	r.name	r.name2
0	132	{brød,1_1}	domain	domain
1	132	{brød,1_1}	domain	domain
2	132	{brød,1_1}	eqSynonymOf	eq_has_synonym
3	132	{brød,1_1}	hyponymOf	has_hyperonym
4	132	{brød,1_1}	madeBy	made_by
5	132	{brød,1_1}	madeofHolonymOf	has_mero_madeof
6	132	{brød,1_1}	usedFor	used_for
7	135	{bagværk_1; brød,1_1_2; bag,2_1}	domain	domain
8	135	{bagværk_1; brød,1_1_2; bag,2_1}	eqSynonymOf	eq_has_synonym

Figure 1: Screenshot of dasem webservice with a query on the Danish word "bager" (baker/bake/bakes).

```
>>> semantic = Semantic()
>>> around(semantic.relatedness(['bil', 'lastbil', 'insekt']), 3)
array([[ 1.   ,  0.048,  0.   ],
       [ 0.048,  1.   ,  0.   ],
       [ 0.   ,  0.   ,  1.   ]])
```

Currently, two preliminary semantic evaluations of `dasem` has been performed. One evaluation is based on a translation of the English semantic relatedness data in the original evaluation of ESA [24]. The other evaluation uses a odd-one-out-of-four task, where a semantic outlier word should be distinguished among four words presented. The current implementation of ESA and word2vec reach accuracies of 78% and 64% respectively.[12]

Dasem also implements a simple lemmatizer based on ePAROLE's word form and lemmas.

A Dasem webservice currently runs from http://neuro.compute.dtu.dk/services/dasem/ where the results of ESA and word2vec analysis are displayed together with ePAROLE-based lemmatization and DanNet synsets relations.

[12]See the Jupyter Notebook.

4.10 Named-entity recognition

SpaCy's multilingual model (labeled 'xx') can to some degree extract named entities in Danish texts:

```
>>> import spacy
>>> text = ("Jeg␣tror␣ikke␣at␣Helle␣Thorning␣eller␣Odenses␣"
...         "H.C.␣Andersen␣kommer␣til␣Det␣Kongelige␣Teater␣i␣morgen.")
>>> nlp = spacy.load('xx')
>>> doc = nlp(text)
>>> doc.ents
(Helle Thorning, H.C. Andersen, Det Kongelige Teater)
```

With `polyglot download ner2.da`, Polyglot downloads its named-entity recognizer model. When this model is installed, Polyglot's named entity recognizer can detect the language of the text, find named entity chunks and annotate each of them.

```
>>> from polyglot.text import Text
>>> text = Text("Jeg␣tror␣ikke␣at␣Helle␣Thorning␣eller␣Odenses␣"
...             "H.C.␣Andersen␣kommer␣til␣Det␣Kongelige␣Teater␣"
...             "i␣morgen.")
>>> text.entities
[I-PER(['Helle', 'Thorning']), I-PER(['Andersen']),
 I-ORG(['Det', 'Kongelige', 'Teater'])]
```

In this case the initials of H.C. Andersen are lost, while two persons and the organization are correctly annotated as such. The approach is described in [26].

Leon Derczynski's `daner` will work from the command line with Java installed:

```
> git clone git@github.com:ITUnlp/daner.git
> cd daner
> echo "Jeg␣tror␣ikke␣at␣Helle␣Thorning␣eller␣Odenses␣"\
       "H.C.␣Andersen␣kommer␣til␣Det␣Kongelige␣Teater␣"\
       "i␣morgen." > test.txt
> ./daner test.txt > output.txt
> fold -w 60 -s output.txt
Jeg/O tror/O ikke/O at/O Helle/PER Thorning/PER eller/O
Odenses/PER H.C./PER Andersen/PER kommer/O til/O Det/LOC
Kongelige/LOC Teater/LOC i/O morgen/O ./O
```

The Alexandra Institute has trained models that are available through their `danlp` Python package:[13]

```
>>> from danlp.models.ner_taggers import load_ner_tagger_with_flair
>>> from flair.data import Sentence
>>> recognizer = load_ner_tagger_with_flair()
>>> sentence = Sentence(
...     "Jeg␣tror␣ikke␣at␣Helle␣Thorning␣eller␣Odenses␣"
...     "H.C.␣Andersen␣kommer␣til␣Det␣Kongelige␣Teater␣"
...     "i␣morgen.")
>>> recognizer.predict(sentence)
```

[13]See example as https://github.com/alexandrainst/danlp/blob/master/docs/models/ner.md.

```
>>> print(sentence.to_tagged_string())
Jeg tror ikke at Helle <B-PER> Thorning <I-PER> eller Odenses
H.C. <B-PER> Andersen <I-PER> kommer til Det <B-ORG> Kongelige
<I-ORG> Teater <I-ORG> i morgen.
```

Here the algorithm has identified "Thorning", "Andersen" and "Kongelige Teater", while missing "Helle", "H.C." and "Det".

The Alexandra Institute has evaluated the flair-based model against daner, Polyglot and their model with the multilingual BERT and found the best performance with the flair and their multilingual BERT model.

4.11 Entity linking

DBpedia Spotlight for entity linking also exists in a Danish version. A docker image is built that can be pulled from Docker Hub:

```
$ docker pull dbpedia/spotlight-danish
$ docker run -i -p 2240:80 dbpedia/spotlight-danish spotlight.sh
```

Running the docker image brings up a web service on http://localhost:2240/rest/ that is documented at https://www.dbpedia-spotlight.org/api. The REST API of the web service can be reached from within Python, e.g., by

```
>>> import lxml.etree, requests
>>> text = "Mette␣Frederiksen␣er␣på␣Christiansborg."
>>> url = "http://localhost:2240/rest/candidates"
>>> response = requests.get(url, params={"text": text})
>>> tree = lxml.etree.fromstring(response.content)
>>> elements = tree.xpath("//resource[@uri]")
>>> [element.attrib['uri'] for element in elements]
['Mette_Frederiksen', 'Christiansborg']
```

Here the first element of the listing corresponds to the DBpedia Linked Open Data URI http://da.dbpedia.org/resource/Mette_Frederiksen.

5 Audio

5.1 Datasets

A few of the Danish Wikipedia articles have been read out aloud, recorded and released as free audio files. These files are listed on a page on Wikimedia Commons: https://commons.wikimedia.org/wiki/Category:Spoken_Wikipedia_-_Danish. A related page lists free audio files for pronounciation of over hundred Danish words: https://commons.wikimedia.org/wiki/Category:Danish_pronunciation.

From https://librivox.org, LibriVox distributes free crowdsourced audio recordings of readings of public domain works. The project features multiple languages, including 18 completed Danish works (as of November 2017), e.g., "Takt og Tone" of Emma Gad and "Fem Uger i Ballon" of Jules Verne.

5.2 Text-to-speech

Commercial systems with Amazon Polly (https://aws.amazon.com/polly/) and ResponsiveVoice enable Danish cloud-based text-to-speech (TTS) synthesis. ResponsiveVoice.JS (https://responsivevoice.org/) is—as the name implies—a JavaScript library. It is free for non-commercial use. Amazon Polly works from a variety of languages and platforms.

With ResponsiveVoice.JS a short Hello World HTML file for Danish text-to-speech can read:

```
<html>
  <head>
    <script
      src="http://code.responsivevoice.org/responsivevoice.js">
    </script>
  </head>
  <body>
    <button
      type="button"
      onclick='responsiveVoice.speak("hej␣verden", "Danish␣Female");'>
        PLAY
    </button>
  </body>
</html>
```

When viewed in a browser, the HTML page displays a button and each time the button is pressed a female voice sounds with the short Danish greeting. ResponsiveVoice has parameters for varying pitch, speed and volume of the generated voice.

6 Geo-data and services

Various services exist for querying Danish geodata. Danmarks Adressers Web API (DAWA) available from https://dawa.aws.dk/ presents an API for searching for Danish addresses in a variety of ways.

OpenStreetMap (OSM) is an open data world map. OpenStreetMap has extensive updated maps of Denmark with interactive route finding. It is available from https://www.openstreetmap.org. There are various ways to interact with OSM, e.g., by the Overpass API.

6.1 Wikidata

As of November 2016, Wikidata has 15'928 items where the item has been associated with Denmark and has a geo-coordinate. These items can be queried from the Wikidata Query Service with the following SPARQL:

```
select * where {
  ?place wdt:P17 wd:Q35 .
  ?place wdt:P625 ?geo .
  }
```

The data comes with labels and variations (aliases). There are various ways to use this data.

An application is named entity extraction of geo-referenceable names in natural language texts. A prototype for this application is implemented in the `stednavn` Python module available from https://github.com/fnielsen/stednavn. An application of this module on the Danish sentence “Anker Engelunds Vej er i Kongens Lyngby ikke i København eller Sandbjerg.” from within Python3 looks like this:

```
>>> from stednavn import Stednavn
>>> stednavn = Stednavn()
>>> s = ('Anker␣Engelunds␣Vej␣er␣i␣Kongens␣Lyngby␣ikke␣'
...      'i␣København␣eller␣Sandbjerg.')
>>> stednavn.extract_placenames_from_string(s)
['Kongens␣Lyngby', 'København', 'Sandbjerg']
```

Here the module extracts the three different named entities to a list of strings. The module may also be used as a script. The following lines download a historical novel, *Bent Bille*, from Runeberg as a text file and then extract geo-locatable named entities:

```
curl "http://runeberg.org/download.pl?mode=ocrtext&work=bentbille" \
    > bentbille.txt
python -m stednavn bentbille.txt
```

The last command extracts currently 333 words or phrases on the command-line from the 57’898 words document in a matter of a few seconds. The first few lines of the result are listed here:

```
Sjælland
Kloster
Paris
Radsted
Lolland
Søholm
Paris
Borup
København
...
```

There are various problems with the this simple approach.

1. Different entities may have similar names, e.g., “Lyngby” may be one of several separate places in Denmark. Currently there are no way of automatically selecting between the various versions.

2. Some named entities (proper nouns) resemble common nouns, e.g., “Bispen” is the Danish noun for “The bishop”, but is also the name of the cultural institution in Haderslev. “Kloster” in the above example with *Bent Bille* is likely also a similar error. The `stednavn` Python module maintains a stopword list (stopwords-da.txt) of currently 72 words for partially handling these cases. As the case with “Kloster” shows, this list is not complete.

3. The number of places are limited, e.g., only a minority of Danish street names is in Wikidata.

On the positive side, Wikidata records not only inheritly geographical items (towns, streets, etc.) but also items such as companies, events, sculptures and a range of other types of items that can be associated with a geo-coordinate.

The Python module `fromtodk` can return coordinates from Wikidata items and compute the distance between two Wikidata items via geopy's vincenty function. `fromtodk` is available from https://github.com/fnielsen/fromtodk and a web application is running as a prototype from https://fromtodk.herokuapp.com/. It can compute the distance between, e.g., the university department *DTU Compute* and the sculpture *Storkespringvandet*. With the Heroku-based fromtodk webservice the URL is:

https://fromtodk.herokuapp.com/?f=DTU+Compute&t=Storkespringvandet

It presently reports 12.3 kilometers. The command-line version would look like this:

```
$ python -m fromtodk "DTU␣Compute" "Storkespringvandet"
12.3085761921
```

7 Public sector data

7.1 Company information

The Danish Business Authority (Erhvervsstyrelsen, ERST) makes several datasets available. http://datahub.virk.dk/data/ points to currently 197 different business-relevant datasets or tools from ERST and other Danish agencies.

An interactive search interface for The Central Business Register (Det Centrale Virksomhedsregister) is available from https://datacvr.virk.dk. This particular database contains the "CVR number" (the company identifier that is usually a number, — old companies may have contain letter), addresses and information about board members, top-level directors (CEOs), owners, company state (e.g., whether it is bankrupt), number of employers and other relevant data. There is a API available at http://distribution.virk.dk/cvr-permanent/_mapping. It is password protected to guard against ad spamming.

ERST publishes digital company filings that includes annual financial statements. These are for almost all companies available in PDF and in the XML dialect XBRL. ERST makes a sample on 1'000 company filings available for download at:

http://datahub.virk.dk/dataset/regnskabsdata-fra-selskaber-sample.

An API with ElasticSearch returning JSON to pointers for the complete data is available from http://distribution.virk.dk/offentliggoerelser. It returns URLs to the XBRL and PDF files. An example of searching and returning information from within an Python program is available in the `cvrminer` Python package at

https://github.com/fnielsen/cvrminer/

within the `xbrler` submodule. `cvrminer` and its submodule also works as a script. For instance, searching for filings of the restaurant Noma can be done with the following command:

```
python -m cvrminer.xbrler search --cvr=27394698
```

It returns JSON lines on standard output.

A webservice which aggregate information from ERST about XBRL data and Wikidata are available from https://tools.wmflabs.org/cvrminer/.

Acknowledgement

Thanks to Lene Offersgaard, Asger Møberg, Michael Riis Andersen for pointing to resources and functionalities. This work was supported by the Danish Innovation Foundation (Innovationsfonden) through the projects Danish Center for Big Data Analytics and Innovation (DABAI) and ATEL.

References

[1] Mohamad Mehdi, Chitu Okoli, Mostafa Mesgari, Finn Årup Nielsen, and Arto Lanamäki. Excavating the mother lode of human-generated text: A systematic review of research that uses the Wikipedia corpus. *Information Processing & Management*, October 2016. In press.

[2] Imene Bensalem, Salim Chikhi, and Paolo Rosso. Building Arabic corpora from Wikisource. In *2013 ACS International Conference on Computer Systems and Applications (AICCSA)*. IEEE, 2013.

> Annotation: Describes a system for collecting texts from the Arabic Wikisource to form a corpus that can be used in a text mining application for plagiarism detection.

[3] Nikals Isenius, Sumithra Velupillai, and Maria Kvist. Initial results in the development of SCAN: a Swedish clinical abbreviation normalizer. In Hanna Suominen, editor, *CLEFeHealth 2012: The CLEF 2012 Workshop on Cross-Language Evaluation of Methods, Applications, and Resources for eHealth Document Analysis*, 2012.

[4] Steven Bird, Ewan Klein, and Edward Loper. Natural language processing with Python. O'Reilly, Sebastopol, California, June 2009.

> Annotation: The canonical book for the NLTK package for natural language processing in the Python programming language. Corpora, part-of-speech tagging and machine learning classification are among the topics covered.

[5] Uwe Quasthoff, Matthias Richter, and Christian Biemann. Corpus portal for search in monolingual corpora. In *Proceedings of the fifth international conference on Language Resources and Evaluation*, pages 1799–1802, 2006.

[6] Matthias T. Kromann and Stine K. Lynge. Danish Dependency Treebank v. 1.0. Department of Computational Linguistics, Copenhagen Business School., 2004.

[7] Matthias T. Kromann. The Danish Dependency Treebank: Linguistic principles and semi-automatic tagging tools. In *Swedish Treebank Symposium*, August 2002.

[8] Ryan McDonald and Fernando Pereira. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, 2006.

[9] Sabine Buchholz and Erwin Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164. Association for Computational Linguistics, June 2006.

[10] Jørg Asmussen. Korpus 2000. Et overblik over projektets baggrund, fremgangsmåder og perspektiver. *Studies in Modern Danish*, pages 27–38, 2002.

[11] Lene Offersgaard, Bart Jongejan, Mitchell Seaton, and Dorte Haltrup Hansen. CLARIN-DK – status and challenges. *Proceedings of the workshop on Nordic language research infrastructure at NODALIDA 2013*, pages 21–32, 2013.

[12] Sigrid Klerke and Anders Søgaard. DSim, a Danish Parallel Corpus for Text Simplification. *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 4015–4018, 2012.

[13] Bolette Sandford Pedersen, Sanni Nimb, Jørg Asmussen, Nicolai Hartvig Sørensen, Lars Trap-Jensen, and Henrik Lorentzen. DanNet: the challenge of compiling a wordnet for Danish by reusing a monolingual dictionary. *Language Resources and Evaluation*, 43:269–299, August 2009.

[14] Francis Bond and Kyonghee Paik. A survey of WordNets and their licenses. In *Proceedings of the 6th Global WordNet Conference (GWC 2012)*, pages 64–71, 2012.

[15] Francis Bond and Ryan Foster. Linking and extending an open multilingual wordnet. In *51st Annual Meeting of the Association for Computational Linguistics: ACL-2013*, pages 1352–1362, 2013.

[16] Finn Årup Nielsen. Ordia: A Web application for Wikidata lexemes. May 2019.

[17] Steven Bird. NLTK: the natural language toolkit. *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72, 2006.

[18] Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Polyglot: Distributed Word Representations for Multilingual NLP. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, June 2014.

[19] Leon Derczynski, Camilla Vilhelmsen Field, and Kenneth S. Bøgh. DKIE: Open Source Information Extraction for Danish. *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 61–64, April 2014.

[20] Finn Årup Nielsen. A new ANEW: evaluation of a word list for sentiment analysis

in microblogs. In Matthew Rowe, Milan Stankovic, Aba-Sah Dadzie, and Mariann Hardey, editors, *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages*, volume 718 of *CEUR Workshop Proceedings*, pages 93–98, May 2011.

Annotation: Initial description and evaluation of the AFINN word list for sentiment analysis.

[21] Yanqing Chen and Steven Skiena. Building Sentiment Lexicons for All Major Languages. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 383–389, June 2014.

[22] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomáš Mikolov. Enriching Word Vectors with Subword Information. July 2016.

[23] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomáš Mikolov. Bag of Tricks for Efficient Text Classification. August 2016.

[24] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using Wikipedia-based explicit sematic analysis. In *Proceedings of The Twentieth International Joint Conference for Artificial Intelligence*, pages 1606–1611, 2007.

[25] Radim Řehůřek and Petr Sojka. Software framework for topic modelling with large corpora. In *Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks*, 2010.

[26] Rami Al-Rfou, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. POLYGLOT-NER: Massive Multilingual Named Entity Recognition. *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 586–594, October 2014.

Index