Biomarker_NLP

J. Lin, Y. He, S. Madhavan, C. Kim, S. M. Boca

Sep 04, 2021

CONTENTS:

CHAPTER

ONE

BIOMARKER_NLP

1.1 biomarker_extraction module

biomarker_extraction.**disease_content**(*dailyMedURL*, *disease*, *header=False*)

Extract subsection for a particular disease from a drug's DailyMed 'INDICATIONS AND USAGE' section.

Parse the URL link using the lxml library. Locate the subsection that discusses the disease in the 'INDICATIONS AND USAGE' section from the HTML's tree structure. If the header argument is set to False, only the text content will be extracted. If the header argument is set to True, the whole subsection including the subheading and its text content will be extracted.

Parameters

- **dailyMedURL** (*str*) – An URL link to a drug's DailyMed information page and quoted ("") as a string.
- **disease** (*str*) – The name of a disease whose text information will be extracted from the 'INDICATIONS AND USAGE' section.
- **header** (*bool, optional*) – Extract the subheading, by default False. If True, the subheading will be extracted. If False, only the text content will be extracted.

Returns Return the subsection text with or without the subheading. If no associated subsection of the disease is found, return None.

Return type None or str

Examples

Import the module

```
>>> from biomarker_nlp import biomarker_extraction
```

Example (without subheading)

```
>>> url = "https://dailymed.nlm.nih.gov/dailymed/drugInfo.cfm?setid=939b5d1f-9fb2-
↪4499-80ef-0607aa6b114e"
>>> disease = "Cervical Cancer"
>>> biomarker_extraction.disease_content(dailyMedURL = url, disease = disease,␣
↪header = False)
'\nAvastin, in combination with paclitaxel and cisplatin or paclitaxel and␣
↪topotecan, is indicated for the treatment of patients with persistent, recurrent,␣
↪or metastatic cervical cancer.'
```

Example (with subheading)

```
>>> url = "https://dailymed.nlm.nih.gov/dailymed/drugInfo.cfm?setid=939b5d1f-9fb2-
↪4499-80ef-0607aa6b114e"
>>> disease = "Cervical Cancer"
>>> biomarker_extraction.disease_content(dailyMedURL = url, disease = disease,␣
↪header = True)
'1.5    Persistent, Recurrent, or Metastatic Cervical Cancer\nAvastin, in␣
↪combination with paclitaxel and cisplatin or paclitaxel and topotecan, is␣
↪indicated for the treatment of patients with persistent, recurrent, or metastatic␣
↪cervical cancer.'
```

`biomarker_extraction.`**`drug_brand_label`**`(`*dailyMedURL*`)`

Extract drug label at the drug dailyMed label information page.

Parse the URL link using the lxml library. Locate and extract the drug label from the HTML's tree structure.

Parameters **`dailyMedURL`** (*`str`*) – An URL link to a drug DailyMed information page and quoted ("") as a string.

Returns Return drug label. If no drug label is found, return a zero-length string ("").

Return type str

Examples

Import the module

```
>>> from biomarker_nlp import biomarker_extraction
```

Example

```
>>> url = "https://dailymed.nlm.nih.gov/dailymed/drugInfo.cfm?setid=43a4d7f8-48ae-
↪4a63-9108-2fa8e3ea9d9c&audience=consumer"
>>> biomarker_extraction.drug_brand_label(dailyMedURL = url)
'SUTENT- sunitinib malate capsule'
```

See also:

`ndc_code`

`biomarker_extraction.`**`drug_search_url`**`(`*url*`)`

Extract the URL link about the search outcome of drug label information. DailyMed URL link (drug information vs multiple research results).

Extract the URL link of "FDA label information for this drug is available at DailyMed." on the therapy's NCI page. The URL link could be either a DailyMed link to drug label information or a DailyMed link showing multiple search results of the drug.

Parameters **`url`** (*`str`*) – An URL link to a targeted therapy's NCI page.

Returns Return a DailyMed URL link in a list.

Return type list

Examples

Import the module

```
>>> from biomarker_nlp import biomarker_extraction
```

Example

```
>>> url = "https://www.cancer.gov/about-cancer/treatment/drugs/atezolizumab"
>>> biomarker_extraction.drug_search_url(url = url)
['https://dailymed.nlm.nih.gov/dailymed/drugInfo.cfm?setid=6fa682c9-a312-4932-9831-
↪f286908660ee&audience=consumer']
>>> url = "https://www.cancer.gov/about-cancer/treatment/drugs/bevacizumab"
>>> biomarker_extraction.drug_search_url(url = url)
['https://dailymed.nlm.nih.gov/dailymed/search.cfm?labeltype=all&query=BEVACIZUMAB&
↪pagesize=20&page=1']
```

`biomarker_extraction.`**`gene_protein_chemical`**`(`*text*, *gene=1*, *protein=1*, *chemical=1*`)`

Extract gene, protein, and drug labels from a string.

The function uses three pre-trained NER models from scispacy. Please see https://allenai.github.io/scispacy/. We use en_ner_craft_md model to recognize genes. Entities labeled with "GGP" in this model are categorized as genes. We use the en_ner_jnlpba_md model to recognize proteins. Entities labeled with "PROTEIN" in this model are categorized as proteins. We use en_ner_bionlp13cg_md model to recognize drugs. Entities labeled with "SIMPLE_CHEMICAL" in this model are categorized as drugs.

Parameters

- **`text`** (*`str`*) – A single string.
- **`gene`** (*`int, optional`*) – Extract genes, by default 1. 0: do not extract genes. 1: extract genes.
- **`protein`** (*`int, optional`*) – Extract proteins, by default 1. 0: do not extract proteins. 1: extract proteins.
- **`chemical`** (*`int, optional`*) – Extract simple chemicals, by default 1. 0: do not extract simple chemicals. 1: extract simple chemicals.

Returns Return a dictionary in which "gene", "proteins", or/and "chemical" are the keys, lists of genes, proteins, or/and chemicals are the values.

Return type dic

Examples

Install the necessary packages and pre-trained models

```
>>> # If using Colab Notebook, use !pip
$ pip install scispacy
$ pip install -U spacy==2.3.1
$ pip install https://s3-us-west-2.amazonaws.com/ai2-s2-scispacy/releases/v0.3.0/en_
↪ner_craft_md-0.3.0.tar.gz
$ pip install https://s3-us-west-2.amazonaws.com/ai2-s2-scispacy/releases/v0.3.0/en_
↪ner_jnlpba_md-0.3.0.tar.gz
$ pip install https://s3-us-west-2.amazonaws.com/ai2-s2-scispacy/releases/v0.3.0/en_
↪ner_bionlp13cg_md-0.3.0.tar.gz
```

Import the module

```
>>> from biomarker_nlp import biomarker_extraction
```

Example (Recognize entities)

```
>>> txt = "Patients with EGFR or ALK genomic tumor aberrations should have disease␣
↪progression on FDA-approved therapy for NSCLC harboring these aberrations prior␣
↪to receiving TECENTRIQ."
>>> biomarker_extraction.gene_protein_chemical(text = txt, gene= 1, protein = 1,␣
↪chemical = 1)
{'gene': ['EGFR', 'ALK genomic'], 'protein': ['EGFR', 'TECENTRIQ'], 'chemical': []}
>>> genProChe = biomarker_extraction.gene_protein_chemical(text = txt, gene= 1,␣
↪protein = 1, chemical = 1)
# get genes
>>> genProChe.get("gene")
['EGFR', 'ALK genomic']
# get proteins
>>> genProChe.get("protein")
['EGFR', 'TECENTRIQ']
# Only detect genes
>>> biomarker_extraction.gene_protein_chemical(text = txt, gene= 1, protein = 0,␣
↪chemical = 0)
{'gene': ['EGFR', 'ALK genomic']}
```

`biomarker_extraction.`**`is_accelerated_approval`**`(`*text*`)`

Detect if the drug is accelerated approval.

Use keyword matching to detect if the keyword “accelerated approval” appears in the sentence.

Parameters **`text`** (*str*) – A string.

Returns Return True if “accelerated approval” is mentioned in the string, False otherwise.

Return type bool

Examples

Import the module

```
>>> from biomarker_nlp import biomarker_extraction
```

Example

```
>>> txt = "This indication is approved under accelerated approval based on␣
↪progression free survival."
>>> biomarker_extraction.is_accelerated_approval(text = txt)
True
```

`biomarker_extraction.`**`is_accelerated_approval_rate`**`(`*text*`)`

Detect if the drug is accelerated approval based on response rate.

Use keyword matching to detect if the keywords “accelerated approval” and “response rate” appear in the sentence.

Parameters **`text`** (*str*) – A string.

Returns Return True if "accelerated approval" and "response rate" are mentioned in the string, False otherwise.

Return type bool

Examples

Import the module

```
>>> from biomarker_nlp import biomarker_extraction
```

Example

```
>>> txt = "This indication is approved under accelerated approval based on tumor␣
↪response rate and durability of response."
>>> biomarker_extraction.is_accelerated_approval_rate(text = txt)
True
```

biomarker_extraction.**is_firstline**(*text*, *medicine*, *disease*)

Detect if first-line treatment is mentioned with a medicine in a sentence.

Use keyword matching to detect if the keywords "first-line treatment" or "first-or second-line treatment", medicine name, and disease name all appear in the sentence.

Parameters

- **text** (*str*) – A single sentence.
- **medicine** (*str*) – A medicine's name.

Returns Return True if the medicine and first-line treatment are mentioned in the sentence, False otherwise.

Return type bool

Examples

Import the module

```
>>> from biomarker_nlp import biomarker_extraction
```

Example

```
>>> txt = "TECENTRIQ, in combination with carboplatin and etoposide, is indicated␣
↪for the first-line treatment of adult patients with extensive-stage small cell␣
↪lung cancer (ES-SCLC)."
>>> medicine = "TECENTRIQ"
>>> disease = "small cell lung cancer"
>>> biomarker_extraction.is_firstline(text = txt, medicine = medicine, disease =␣
↪disease)
True
```

biomarker_extraction.**is_metastatic**(*text*, *disease*)

Detect if the metastatic disease is mentioned.

Use keyword matching to detect if the combination of keywords "metastatic" or "unresectable" with the disease's name appears in the sentence.

Parameters

- **text** (*str*) – A string or sentence
- **disease** (*str*) – A disease's name

Returns Return True if "metastatic" or "unresectable" and the disease are mentioned together, False otherwise.

Return type bool

Examples

Import the module

```
>>> from biomarker_nlp import biomarker_extraction
```

Example

```
>>> txt = "TECENTRIQ, in combination with bevacizumab, is indicated for the␣
↪treatment of patients with unresectable or metastatic hepatocellular carcinoma␣
↪(HCC) who have not received prior systemic therapy."
>>> disease = "hepatocellular carcinoma"
>>> biomarker_extraction.is_metastatic(text = txt, disease = disease)
True
```

`biomarker_extraction.`**`ndc_code`**`(`*dailyMedURL*`)`

Extract NDC code(s) from the drug dailyMed label information page.

Parse the URL link using the lxml library. Locate and extract the NDC codes from the HTML's tree structure. It returns all the codes found in a string. The codes are separated by commas.

Parameters **dailyMedURL** (*str*) – An URL link to a drug DailyMed information page and quoted ("") as a string.

Returns Return NDC codes(s) in a string. If more than one code is found, codes are separated by commas.

Return type str

Examples

Import the module

```
>>> from biomarker_nlp import biomarker_extraction
```

Example

```
>>> url = "https://dailymed.nlm.nih.gov/dailymed/drugInfo.cfm?setid=43a4d7f8-48ae-
↪4a63-9108-2fa8e3ea9d9c&audience=consumer"
>>> biomarker_extraction.ndc_code(dailyMedURL = url)
'0069-0550-38, 0069-0770-38, 0069-0830-38, 0069-0980-38'
```

`biomarker_extraction.`**`section_content`**`(`*dailyMedURL*, *section*`)`

Extract a whole section text content from the drug's DailyMed information page excluding the section heading.

Parse the URL link using the lxml library. Locate the section using the section's heading from the HTML's tree structure. Extract the text content of the section excluding the heading.

Parameters

- **dailyMedURL** (*str*) – An URL link to a drug DailyMed information page and quoted (“”) as a string.
- **section** (*str*) – The header of the section.

Returns Return section content. If no such a section is found, return None.

Return type None or str

Examples

Import the module

```
>>> from biomarker_nlp import biomarker_extraction
```

Example

```
>>> url = "https://dailymed.nlm.nih.gov/dailymed/drugInfo.cfm?setid=43a4d7f8-48ae-
↪4a63-9108-2fa8e3ea9d9c&audience=consumer"
>>> sectionHeader = "INDICATIONS AND USAGE"
>>> biomarker_extraction.section_content(dailyMedURL = url, section = sectionHeader)
'1.1  Gastrointestinal Stromal Tumor
SUTENT is indicated for the treatment of adult patients with gastrointestinal␣
↪stromal tumor (GIST) after disease progression on or intolerance to imatinib␣
↪mesylate.
1.2   Advanced Renal Cell Carcinoma
SUTENT is indicated for the treatment of adult patients with advanced renal cell␣
↪carcinoma (RCC).
1.3   Adjuvant Treatment of Renal Cell Carcinoma
SUTENT is indicated for the adjuvant treatment of adult patients at high risk of␣
↪recurrent RCC following nephrectomy.
1.4   Advanced Pancreatic Neuroendocrine Tumors
SUTENT is indicated for the treatment of progressive, well-differentiated␣
↪pancreatic neuroendocrine tumors (pNET) in adult patients with unresectable␣
↪locally advanced or metastatic disease.'
```

See also:

disease_content

biomarker_extraction.**sent_subtree**(*text*)

Extract the subtree of the patterns ‘in combination with’ and ‘used with’ based on dependency parsing.

The function uses pattern match to recognize two patterns (‘in combination with’ and ‘used with’) from a sentence. Once such a pattern is recognized, the sentence is parsed as a dependency tree by scispacy’s nlp_bionlp13cg model which is based on Stanford Dependency Converter. The “combination” or “used” is used as a headword to extract its subtree.

Parameters **text** (*str*) – A single sentence.

Returns Return a list of subtree clauses in string.

Return type list

Examples

Install the necessary packages and pre-trained models

```
>>> # If using Colab Notebook, use !pip
$ pip install scispacy
$ pip install -U spacy==2.3.1
$ pip install https://s3-us-west-2.amazonaws.com/ai2-s2-scispacy/releases/v0.3.0/en_
↪ner_bionlp13cg_md-0.3.0.tar.gz
```

Import the module

```
>>> from biomarker_nlp import biomarker_extraction
```

Example (extract the subtree)

```
>>> txt = "TECENTRIQ, in combination with cobimetinib and vemurafenib, is indicated␣
↪for the treatment of patients with BRAF V600 mutation-positive unresectable or␣
↪metastatic melanoma."
>>> biomarker_extraction.sent_subtree(text = txt)
['in combination with cobimetinib and vemurafenib']
```

Another example

```
>>> txt = "BAVENCIO in combination with axitinib is indicated for the first-line␣
↪treatment of patients with advanced renal cell carcinoma (RCC)."
>>> biomarker_extraction.sent_subtree(text = txt)
['in combination with axitinib']
```

`biomarker_extraction.`**`targeted_therapy_name`**`(`*url*`)`

Extract targeted therapy's name on its NCI page.

Use the lxml library to parse the URL link and xpath to find the therapy's name from the parsed tree structure.

Parameters **`url`** (*`str`*) – An URL link to a targeted therapy's NCI page.

Returns Return the name of the targeted therapy appearing on its NCI page in a list.

Return type list

Examples

Import the module

```
>>> from biomarker_nlp import biomarker_extraction
```

Example

```
>>> url = "https://www.cancer.gov/about-cancer/treatment/drugs/atezolizumab"
>>> biomarker_extraction.targeted_therapy_name(url = url)
['Atezolizumab']
```

`biomarker_extraction.`**`targeted_therapy_url`**`(`*url*`)`

Extract the targeted therapies' link from NCI page.

Use the lxml library to parse the URL link and xpath to find the therapy's link in the "what-targeted-therapies-have-been-approved-for-specific-types-of-cancer" section.

Parameters **url** (*str*) – An URL link to a targeted therapies fact sheet.

Returns Return a list of targeted therapies' URL links.

Return type list

Examples

Import the module

```
>>> from biomarker_nlp import biomarker_extraction
```

Example

```
>>> url_nci = "https://www.cancer.gov/about-cancer/treatment/types/targeted-
↪therapies/targeted-therapies-fact-sheet"
>>> biomarker_extraction.targeted_therapy_url(url = url_nci)[0:5]
['/about-cancer/treatment/drugs/atezolizumab', '/about-cancer/treatment/drugs/
↪nivolumab', '/about-cancer/treatment/drugs/avelumab', '/about-cancer/treatment/
↪drugs/pembrolizumab', '/about-cancer/treatment/drugs/erdafitinib']
```

`biomarker_extraction.`**`therapy_disease`**`(`*url*`)`

Extract the associated diseases (in bold text) on the therapy's NCI page.

Extract the associated diseases in bold text on the therapy's NCI page through parsing the URL link. Remove punctuations around the text if there are any.

Parameters **url** (*str*) – An URL link to a targeted therapy's NCI page.

Returns Return a list of disease's name.

Return type list

Examples

Import the module

```
>>> from biomarker_nlp import biomarker_extraction
```

Example

```
>>> url = "https://www.cancer.gov/about-cancer/treatment/drugs/atezolizumab"
>>> biomarker_extraction.therapy_disease(url = url)
['Breast cancer', 'Urothelial carcinoma', 'Non-small cell lung cancer',
↪'Hepatocellular carcinoma', 'Melanoma', 'Small cell lung cancer']
```

1.2 negation_cue_scope module

`negation_cue_scope.negation_detect`(*text*, *modelCue*)

Detect if a sentence contains any negation cues.

This function predicts if a sentence contains any negation words by using a pre-trained negation detection model that was pre-trained through Aditya and Suraj's (2020) NegBERT transfer learning program. Please see the reference link: https://github.com/adityak6798/Transformers-For-Negation-and-Speculation. The model was trained using 'bioscope_abstracts' and 'bioscope_full_papers' corpora. It was upload to a cloud repository and was freely available.

Parameters

- **text** (*str*) – a single sentence.
- **modelCue** (*torch model*) – pre-trained negation cue detection model

Returns True if any negation cues are detected. False if no negation cues are detected.

Return type bool

See also:

negation_scope

Notes

NVIDIA GPU will be required. Make sure your machine brings NVIDIA GPU or set GPU as Hardware accelerator if using Colab notebook.

Examples

Install the necessary packages, and make sure to turn the GPU Hardware accelerator on.

```
>>> If using Colab Notebook, use !pip instead pip.
$ pip install biomarker_nlp
$ pip install transformers
$ pip install knockknock==0.1.7
$ pip install sentencepiece
```

Load the necessary packages and pre-trained model

```
>>> from biomarker_nlp import negation_cue_scope
>>> from biomarker_nlp.negation_negbert import * # This code MUST be run before␣
↪loading the pre-trained negation models
>>> modelCue = torch.load('/path/to/negation/cue/detection/model') # path to the␣
↪location where the model file is placed
```

Examples (predict negation)

```
>>> txt = "TECENTRIQ is not indicated for use in combination with paclitaxel for␣
↪the treatment of adult patients with unresectable locally advanced or metastatic␣
↪TNBC."
>>> negation_cue_scope.negation_detect(text = txt, modelCue = modelCue)
True
>>> txt = "KEYTRUDA is not recommended for treatment of patients with PMBCL who␣
↪require urgent cytoreductive therapy."
```

(continues on next page)

(continued from previous page)

```
>>> negation_cue_scope.negation_detect(text = txt, modelCue = modelCue)
True
>>> txt = "KEYTRUDA is indicated for the treatment of adult patients with relapsed␣
↪or refractory classical Hodgkin lymphoma (cHL)."
>>> negation_cue_scope.negation_detect(text = txt, modelCue = modelCue)
False
```

`negation_cue_scope.`**`negation_scope`**(*text*, *modelCue*, *modelScope*)

Extract the scope of negation in a sentence.

This function predicts the negation cues and their scope in a sentence by using two pre-trained negation models that were pre-trained through Aditya and Suraj's (2020) NegBERT transfer learning program. Please see the reference link: https://github.com/adityak6798/Transformers-For-Negation-and-Speculation. The models were trained using 'bioscope_abstracts' and 'bioscope_full_papers' corpora. One of them is used to performed negation cue detection and negation scope resolution. Another one is used to performed negation scope resolution. They were upload to a cloud repository and were freely available. This function predicts negation cues first through the negation cue detection model. If any negation cues were predicted, it predicts the scope of each negation cue, if there are more than one, through the negation scope resolution model.

Parameters

- **`text`** (*`str`*) – a single sentence.
- **`modelCue`** (*`torch model`*) – pre-trained negation cue prediction model
- **`modelScopre`** (*`torch model`*) – pre-trained negation scope prediction model

Returns a list of negated clauses. Some sentences would contain more than one negation cue, in this case, all of their negated clauses will be extracted. If no negation is found, return an empty list.

Return type list

See also:

`negation_detect`

Notes

The negation cue will not be extracted. NVIDIA GPU will be required. Make sure your machine brings NVIDIA GPU or set GPU as Hardware accelerator if using Colab notebook.

Examples

Install the necessary packages, and make sure to turn the GPU Hardware accelerator on.

```
>>> If using Colab Notebook, use !pip instead pip.
$ pip install biomarker_nlp
$ pip install transformers
$ pip install knockknock==0.1.7
$ pip install sentencepiece
```

Load the necessary packages and pre-trained model

```
>>> from biomarker_nlp import negation_cue_scope
>>> from biomarker_nlp.negation_negbert import * # This code MUST be run before␣
↪loading the pre-trained negation models
```

(continues on next page)

(continued from previous page)

```
>>> modelCue = torch.load('/path/to/negation/cue/detection/model') # path to the␣
↪location where the model file is placed
>>> modelScope = torch.load('/path/to/negation/scope/detection/model') # path to␣
↪the location where the model file is placed
```

Examples (predict negation scope)

```
>>> txt = "TECENTRIQ is not indicated for use in combination with paclitaxel for␣
↪the treatment of adult patients with unresectable locally advanced or metastatic␣
↪TNBC."
>>> negation_cue_scope.negation_scope(text = txt, modelCue = modelCue, modelScope =␣
↪modelScope)
['TECENTRIQ is', 'indicated for use in combination with paclitaxel for']
>>> txt = "KEYTRUDA is not recommended for treatment of patients with PMBCL who␣
↪require urgent cytoreductive therapy."
>>> negation_cue_scope.negation_scope(text = txt, modelCue = modelCue, modelScope =␣
↪modelScope)
['KEYTRUDA is', 'recommended for treatment of patients with PMBCL who']
```

1.3 negation_negbert module

This is part of Aditya Khandelwal & Suraj Sawant's (2020) NegBERT program, it is necessary to run the negation cue detection (negCue) and negation scope resolution (negScope) models. For more information about the NegBERT program, please see https://github.com/adityak6798/Transformers-For-Negation-and-Speculation.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX