Analiza danych tekstowych

dr inż. Aleksander Smywiński-Pohl

apohllo@agh.edu.pl

http://apohllo.pl/dydaktyka/nlp (http://apohllo.pl/dydaktyka/nlp)

Plan

- dane tekstowe
    - charakterystyka danych tekstowych
    - wyrażenia regularne
    - odległość edycyjna oraz inne miary odległości
    - normalizacja tekstu: tokenizacja, podział na zdania
    - wyrażenia wielosegmentowe
- podstawowe algorytmy NLP
    - stemming i lematyzacja
    - tagowanie
    - parsowanie
    - rozpoznawanie jednostek nazewniczych
    - modelowanie języka
- zastosowania

Dane tekstowe

- charakterystyka danych tekstowych
- wyrażenia regularne
- odległość edycyjna oraz inne miary odległości
- normalizacja tekstu: tokenizacja, podział na zdania
- wyrażenia wielosegmentowe

Ile razy w bazie danych w tabeli `words` w kolumnie `value` występuje słowo `wirus`?

```
select count(*) from words where value = 'wirus'
```

Ile razy na stronie `onet.pl` występuje słowo "wirus"?

- czy uwzględniamy pisownię wielką i małą literą?
- czy uwzględniamy fleksję? - wirus, wirsa, wirusowi, wirusem
- czy uwzględniamy słowotwórstwo i relacje morfologiczne? - wirsu/wirusowe, lekarz/leczenie
- jak szybko można to policzyć?

Ile razy w tekście występuje wyrażenie ”wirusowe zapalenie żołądka”?

- **czy wszystkie słowa są odmienne?**
- **czy pomiędzy słowami wyrażenia mogą występować inne słowa?**
- **czy spacja jest zawsze tym czym nam się wydaje, że jest?**

Wyrażenia regularne - regex(p)

- pozwalają na konstrukcję wzorców dopasowujących się do tekstu
- pozwalają na uwzględnienie wariantów w pisowni
- w ograniczonym stopniu pozwalają na uwzględnienie błędów w pisowni
- w ograniczonym stopniu pozwalają na uwzględnienie fleksji
- wiele realnie działających systemów przetwarzających tekst opiera się w dużej mierze na wyrażeniach regularnych

In [ ]:
```
import re

pattern = re.compile('łódź', re.I)
text = "Łódź to stolica województwa łódzkiego"
if(pattern.search(text)):
    print("Wyrażenie zostało dopasowane")
```

In [ ]:
```
tab = ["Poznań", "Łódź", "Andrychów"]
sorted(tab)
```

In [40]:
```
pattern = re.compile(r'\bwirus(\w*)',)
texts = ["a _wirusów", "wirusowe zapalenie żołądka", "wirusy są groźne", "korono
wirus"]
for text in texts:
    match_data = pattern.search(text)
    if(match_data):
        print(match_data[0])
```

```
wirusowe
wirusy
```

Wyrażenia regularne

- Podstawowe metaznaki:
    - * – (kwantyfikator) zero lub więcej wystąpień
    - ( ) – grupowanie wyrażeń dla kwantyfikatorów, alternatywy, dopasowań wstecznych
    - | – alternatywa – jedna opcja spośród wielu
- Kwantyfikatory (określają ile razy ma być dopasowane wyrażenie, które stoi przed nimi):
    - * – zero lub więcej wystąpień (to samo co wyżej)
    - + – jedno lub więcej wystąpień
    - ? – zero lub jedno wystąpienie

- Kotwice (dopasowują się do pozycji w łańcuchu, a nie konkretnych znaków):
    - `^` – początek łańcucha
    - `$` – koniec łańcucha
    - `\b` – granica słowa
    - `\<` – początek słowa
    - `\>` – koniec słowa
- Klasy znaków:
    - `[]` – jeden ze znaków znajdujących się wewnątrz nawiasów
    - `^` – pojawiając się na początku w kontekście klasy znaków powoduje jej zanegowanie
    - `a-z` – zakres znaków (tylko wew. nawiasów kwadratowych)
    - `\w` – znak będący literą, cyfrą lub podkreśleniem
    - `\s` – znak będący białą spacją (spacja, tabulator, koniec linii, etc.)
    - `\d` – cyfra
    - `.` – dowolny znak (zazwyczaj – niebędący końcem linii)

https://regex101.com/r/pTEWml/2 (https://regex101.com/r/pTEWml/2)

Zaawansowane własności wyrażeń regularny

- Wsparcie dla Unicode:
  - https://www.regular-expressions.info/unicode.html (https://www.regular-expressions.info/unicode.html)
  - `\p{L}` - litery z dowolnego alfabetu (np. a, ą, ć, ü, 力)
  - `\p{Ll}` - mała litera z dowolnego alfabetu
  - `\p{Lu}` - wielka litera z dowolnego alfabetu
  - `\X` - dowonly znak Unicode (jak `.` ale automatycznie uwzględnia znaki przejścia do nowej linii
- *positive lookahead*
  - wyrażenie `(\w+)(?= ma kota)` dopasuje się do łańcucha Ala ma kota, ale dopasowanie obejmie tylko słowo Ala.
- *negative lookbehind*
  - wyrażenie `(?!<starych )(zł)` dopasuje się do łańcucha 10 złotych ale nie do łańcucha 10 starych złotych.

In [45]:
```
text = "aaa"
re.findall("a(?=a)", text)
```

Out[45]:
```
['a', 'a']
```

Narzędzia

- Python
    - `re` - standardowa biblioteka - https://docs.python.org/3/library/re.html (https://docs.python.org/3/library/re.html)
    - `pip install regex` - biblioteka nie wbudowana w język - https://pypi.org/project/regex/ (https://pypi.org/project/regex/)
- Java
    - java.util.regex.Pattern - https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html (https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html)
- Ruby - https://ruby-doc.org/core-2.5.1/Regexp.html (https://ruby-doc.org/core-2.5.1/Regexp.html)
    - literał `/a.*/`
- Javascript - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions)
    - literał `/a.*/`

Odległość edycyjna (odległość Levenshteina)

Minimalna liczba operacji:

- usunięcia
- dodania
- zamiany

pojedynczego znaku, dzięki którym łańcych S1 może zostać przekształcony w łańcuch S2 .

		k	**i**	**t**	**t**	**e**	**n**
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

		S	**a**	**t**	**u**	**r**	**d**	**a**	**y**
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	3

źródło: https://en.wikipedia.org/wiki/Levenshtein_distance (https://en.wikipedia.org/wiki/Levenshtein_distance)

		k	**i**	**t**	**t**	**e**	**n**
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

		S	**a**	**t**	**u**	**r**	**d**	**a**	**y**
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	3

Implementacje

- algorytm dynamiczny - do porównywania pojedynczych łańcuchów znaków
- algorytm korekty tekstu Norviga - odwrócenie problemu https://norvig.com/spell-correct.html (https://norvig.com/spell-correct.html)
- Levenshtein automaton - znacznie szybszy niż algorytm Norviga, zaimplementowany np. w ElasticSearch

Zastosowania miary edycyjnej

- wyszukiwanie przybliżone (fuzzy search - did you mean?)
- automatyczna korekta tekstu
- interpretacja komend głosowych

Narzędzia

- Fuzzywuzzy - https://github.com/seatgeek/fuzzywuzzy (https://github.com/seatgeek/fuzzywuzzy)
- ElasticSearch - fuzzy match https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-fuzzy-query.html (https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-fuzzy-query.html)
- Algorytm Norviga - https://norvig.com/spell-correct.html (https://norvig.com/spell-correct.html)
- Language tool - https://www.languagetool.org/ (https://www.languagetool.org/)
- biblioteka regex - https://pypi.org/project/regex/ (https://pypi.org/project/regex/)

In [46]:

```
pattern = re.compile('wirus')
texts = ["wrius", "wirrus", "wirs"]
for text in texts:
    match_data = pattern.search(text)
    if(match_data):
        print(match_data[0])
```

In [48]:

```
import regex

pattern = regex.compile(r'(wirus\b){e<=1}')
texts = ["virus", "wirrus", "wirusa"]
for text in texts:
    match_data = pattern.search(text)
    if(match_data):
        print(match_data[0])
```

```
virus
wirrus
wirusa
```

Normalizacja tekstu - tokenizacja

https://regex101.com/r/pTEWml/3/ (https://regex101.com/r/pTEWml/3/)

Kwestie problematyczne:

- podział wyrażeń wielosegmentowych: `Bielsko-Biała`
- wyrażenia z apostrofem: `McDonald's`
- liczby: `100 000 000`, `20%`
- adresu url i e-mail: `http://www.agh.edu.pl`

Narzędzia:

- ElasticSearch - https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-tokenizers.html (https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-tokenizers.html)
- Solr - https://lucene.apache.org/solr/guide/6_6/tokenizers.html (https://lucene.apache.org/solr/guide/6_6/tokenizers.html)

Normalizacja tekstu - podział na zdania

https://regex101.com/r/pTEWml/5/ (https://regex101.com/r/pTEWml/5/)

Narzędzia:

- LanguageTool, reguły SRX - https://github.com/languagetool-org/languagetool (https://github.com/languagetool-org/languagetool)
- Alternatywna implementacja dla reguł SRX - https://github.com/loomchild/segment (https://github.com/loomchild/segment)
- SpaCy https://spacy.io/usage/linguistic-features#sbd (https://spacy.io/usage/linguistic-features#sbd)

In [51]:

```
import spacy
from spacy.lang.pl import Polish

nlp = Polish()
sentencizer = nlp.create_pipe("sentencizer")
nlp.add_pipe(sentencizer)
doc = nlp("Piątek był trzecim dniem w tym tygodniu na giełdzie w Warszawie, w kt
órym WIG20 spadł o więcej niż 4 proc. na zamknięciu sesji. W efekcie cały giełdo
wy tydzień zakończył się zniżką aż o ponad 15 proc., co jest trzecim najgorszym
 wynikiem w historii tego indeksu.")
for sent in doc.sents:
    print(sent.text)
```

```
Piątek był trzecim dniem w tym tygodniu na giełdzie w Warszawie, w którym WIG2
0 spadł o więcej niż 4 proc. na zamknięciu sesji.
W efekcie cały giełdowy tydzień zakończył się zniżką aż o ponad 15 proc., co j
est trzecim najgorszym wynikiem w historii tego indeksu.
```

Wyrażenia wielosegmentowe

- Bielsko-Biała
- Rzeczpospolita Polska
- *post scriptum*
- Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
- zastawka mitralna
- benzyna bezołowiowa

Algorytmy ekstrakcji wyrażeń

- point-wise mutual information (PMI):
  - $pmi(a;b) = log\frac{p(a,b)}{p(a)p(b)}$
- likelihood ratio-test (LLR test):
  - $llr(a;b) = 2 * (H(rows) + H(columns) - H(all))$
  - $H(cells) = -\sum_{i \in cells} (p_i * total * log(p_i))$
  - https://github.com/tdunning/python-llr/blob/master/llr.py (https://github.com/tdunning/python-llr/blob/master/llr.py)
- algorytmy ML oparte o Wikipedię, np. AutoPhrase
  - https://github.com/shangjingbo1226/AutoPhrase (https://github.com/shangjingbo1226/AutoPhrase)
- TermoPL - oparte o miarę C, uwzględniający ograniczenia morfosyntaktyczne
  - $C - value(p)$

$$= \begin{cases} l(p) * \left(freq(p) - \frac{1}{r(LP)} \sum_{lp \in LP} freq(lp)\right) & r(LP) > 0 \\ l(p) * freq(p) & r(LP) = 0 \end{cases}$$

Forma podstawowa	Forma odmieniona	Miara "C"
prezes urząd	[prezes Urzędu]	267.73
kara pieniężny	[kara pieniężna]	69.92
zbiorowy interes konsument	[zbiorowy interes konsumentów]	54.68
ochrona konkurencja	[ochrona konkurencji]	53.33
urząd	[urząd]	30.97
przedsiębiorca	[przedsiębiorca]	30.87
mowa	[mowa]	29.60
sprawa praktyka	[sprawa praktyk]	29.00
sąd ochrona konkurencja	[sąd ochrony konkurencji]	28.00
prezes	[prezes]	27.14
postępowanie antymonopolowy	[postępowanie antymonopolowe]	25.16
droga decyzja	[droga decyzji]	23.00

Wyniki TermoPL dla Ustawy o Ochronie Konkurencji i Konsumentów

Podstawowe algorytmy NLP

- stemming i lematyzacja
- tagowanie
- parsowanie
- rozpoznawanie jednostek nazewniczych
- modelowanie języka

Stemming i lematyzacja

Znajdowanie formy wspólnej dla różnych form powiązanych syntaktycznie.

Stemming (j. angielski)

```
This was not the map we found in Billy Bones's chest, but
an accurate copy, complete in all things-names and heights
and soundings-with the single exception of the red crosses
and the written notes.
```

Porter stemmer (proste przekształcenia usuwające końcówki słów):

```
Thi wa not the map we found in Billi Bone s chest but an
accur copi complet in all thing name and height and sound
with the singl except of the red cross and the written note
```

Lematyzacja (j. polski)

```
Partia polityczna jest dobrowolną organizacją, występującą pod określoną
nazwą, stawiającą sobie za cel udział w życiu publicznym poprzez wywieranie
metodami demokratycznymi wpływu na kształtowanie polityki państwa
lub sprawowanie władzy publicznej.
```

KRNNT (tagger i lemmatyzer)

```
partia polityczny być dobrowolny organizacja, występować pod określony
nazwa, stawiać siebie za cel udział w życie publiczny poprzez wywierać
metoda demokratyczny wpływ na kształtować polityka państwo
lub sprawować władza publiczny.
```

Narzędzia

- Stempel - http://www.getopt.org/stempel/ (http://www.getopt.org/stempel/)
- Morfologik - https://github.com/morfologik/morfologik-stemming (https://github.com/morfologik/morfologik-stemming)
- Morfeusz - http://sgjp.pl/morfeusz/ (http://sgjp.pl/morfeusz/)
- KRNNT - https://github.com/kwrobel-nlp/krnnt/ (https://github.com/kwrobel-nlp/krnnt/)
- Stanford NLP - https://stanfordnlp.github.io/stanfordnlp/installation_download.html (https://stanfordnlp.github.io/stanfordnlp/installation_download.html)
- COMBO - https://github.com/360er0/COMBO (https://github.com/360er0/COMBO)

Tagowanie morfosyntaktyczne

Partia	partia	subst:sg:nom:f
polityczna	polityczny	adj:sg:nom:f:pos
jest	być	fin:sg:ter:imperf
dobrowolną	dobrowolny	adj:sg:inst:f:pos
organizacją	organizacja	subst:sg:inst:f
,	,	interp
występującą	występować	pact:sg:inst:f:imperf:aff
pod	pod	prep:inst:nwok
określoną	określony	adj:sg:inst:f:pos
nazwą	nazwa	subst:sg:inst:f

Tagi morfosyntaktyczne 1/2

flexeme	abbreviation	base form	example
noun	*subst*	singular nominative	*profesor*
depreciative form	*depr*	singular nominative form of the corresponding noun	*profesor*
main numeral	*num*	inanimate masculine nominative form	*pięć, dwa*
collective numeral	*numcol*	inanimate masculine nominative form of the main numeral	*pięć, dwa*
adjective	*adj*	singular nominative masculine positive form	*polski*
ad-adjectival adjective	*adja*	singular nominative masculine positive form of the adjective	*polski*
post-prepositional adjective	*adjp*	singular nominative masculine positive form of the adjective	*polski*
predicative adjective	*adjc*	singular nominative masculine positive form of the adjective	*zdrowy, ciekawy*
adverb	*adv*	positive form	*dobrze, bardzo*
non-3rd person pronoun	*ppron12*	singular nominative	*ja*
3rd-person pronoun	*ppron3*	singular nominative	*on*
pronoun SIEBIE	*siebie*	accusative	*siebie*
non-past form	*fin*	infinitive	*czytać*
future BYĆ	*bedzie*	infinitive	*być*
agglutinate BYĆ	*aglt*	infinitive	*być*
l-participle	*praet*	infinitive	*czytać*
imperative	*impt*	infinitive	*czytać*
impersonal	*imps*	infinitive	*czytać*
infinitive	*inf*	infinitive	*czytać*

http://nkjp.pl/poliqarp/help/ense2.html (http://nkjp.pl/poliqarp/help/ense2.html)

Tagi morfosyntaktyczne 2/2

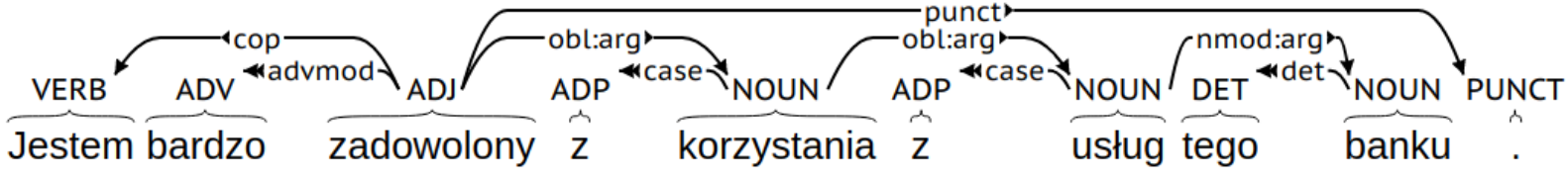
contemporary adv. participle	*pcon*	infinitive	*czytać*
anterior adv. participle	*pant*	infinitive	*czytać*
gerund	*ger*	infinitive	*czytać*
active adj. participle	*pact*	infinitive	*czytać*
passive adj. participle	*ppas*	infinitive	*czytać*
winien	*winien*	singular masculine form	*powinien, rad*
predicative	*pred*	the only form of that flexeme	*warto*
preposition	*prep*	the non-vocalic form of that flexeme	*na, przez, w*
coordinating conjunction	*conj*	the only form of that flexeme	*oraz*
subordinating conjunction	*comp*	the only form of that flexeme	*że*
particle-adverb	*qub*	the only form of that flexeme	*nie, -że, się*
abbreviation	*brev*	the full dictionary form	*rok, i tak dalej*
bound word	*burk*	the only form of that flexeme	*trochu, oścież*
interjection	*interj*	the only form of that flexeme	*ech, kurde*
punctuation	*interp*	the only form of that flexeme	*;, ., (,]*
alien	*xxx*	the only form of that flexeme	*cool , nihil*
unknown form	*ign*	the only form of that flexeme	

http://nkjp.pl/poliqarp/help/ense2.html (http://nkjp.pl/poliqarp/help/ense2.html)

Narzędzia - taggery dla j. polskiego

- Toygger - http://mozart.ipipan.waw.pl/~kkrasnowska/PolEval/src/ (http://mozart.ipipan.waw.pl/~kkrasnowska/PolEval/src/)
- KRNNT - https://github.com/kwrobel-nlp/krnnt (https://github.com/kwrobel-nlp/krnnt)
- COMBO - https://github.com/360er0/COMBO (https://github.com/360er0/COMBO)
- Stanford NLP - https://stanfordnlp.github.io/stanfordnlp/installation_download.html (https://stanfordnlp.github.io/stanfordnlp/installation_download.html)
- Concraft - https://github.com/kawu/concraft-pl (https://github.com/kawu/concraft-pl)

Parsing zależnościowy



Narzędzia

- Universal dependencies - https://universaldependencies.org/ (https://universaldependencies.org/)
- MALT Parser - http://www.maltparser.org/ (http://www.maltparser.org/)
- COMBO - https://github.com/360er0/COMBO (https://github.com/360er0/COMBO)
- Stanford NLP - https://stanfordnlp.github.io/stanfordnlp (https://stanfordnlp.github.io/stanfordnlp)

In [52]:
```
import stanfordnlp
#stanfordnlp.download('pl')
nlp = stanfordnlp.Pipeline(lang='pl', treebank='pl_lfg')
doc = nlp("Piątek był trzecim dniem w tym tygodniu na giełdzie w Warszawie, w kt
órym WIG20 spadł o więcej niż 4 proc. na zamknięciu sesji.")
```

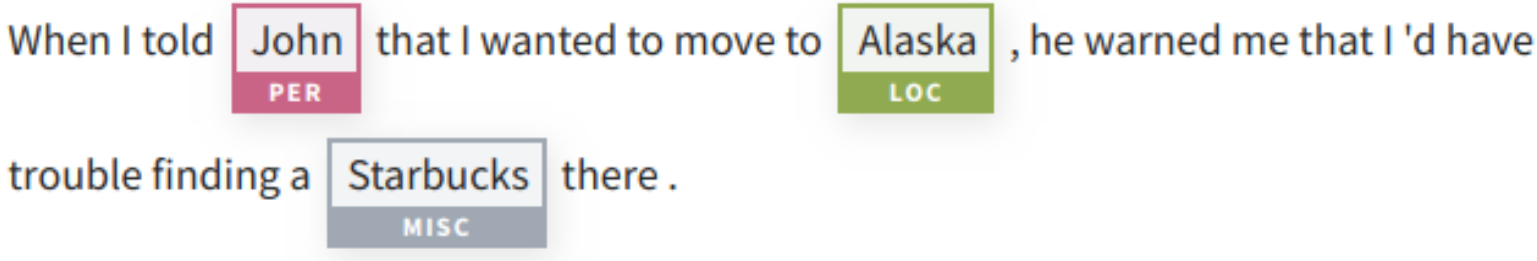
```
Use device: cpu
---
Loading: tokenize
With settings:
{'model_path': '/home/apohllo/stanfordnlp_resources/pl_lfg_models/pl_lfg_token
izer.pt', 'lang': 'pl', 'shorthand': 'pl_lfg', 'mode': 'predict'}
---
Loading: pos
With settings:
{'model_path': '/home/apohllo/stanfordnlp_resources/pl_lfg_models/pl_lfg_tagge
r.pt', 'pretrain_path': '/home/apohllo/stanfordnlp_resources/pl_lfg_models/pl_
lfg.pretrain.pt', 'lang': 'pl', 'shorthand': 'pl_lfg', 'mode': 'predict'}
---
Loading: lemma
With settings:
{'model_path': '/home/apohllo/stanfordnlp_resources/pl_lfg_models/pl_lfg_lemma
tizer.pt', 'lang': 'pl', 'shorthand': 'pl_lfg', 'mode': 'predict'}
Building an attentional Seq2Seq model...
Using a Bi-LSTM encoder
Using soft attention for LSTM.
Finetune all embeddings.
[Running seq2seq lemmatizer with edit classifier]
---
Loading: depparse
With settings:
{'model_path': '/home/apohllo/stanfordnlp_resources/pl_lfg_models/pl_lfg_parse
r.pt', 'pretrain_path': '/home/apohllo/stanfordnlp_resources/pl_lfg_models/pl_
lfg.pretrain.pt', 'lang': 'pl', 'shorthand': 'pl_lfg', 'mode': 'predict'}
Done loading processors!
---
```

```
/pytorch/aten/src/ATen/native/LegacyDefinitions.cpp:14: UserWarning: masked_fi
ll_ received a mask with dtype torch.uint8, this behavior is now deprecated,pl
ease use a mask with dtype torch.bool instead.
```

In [53]:
```
doc.sentences[0].print_dependencies()
```

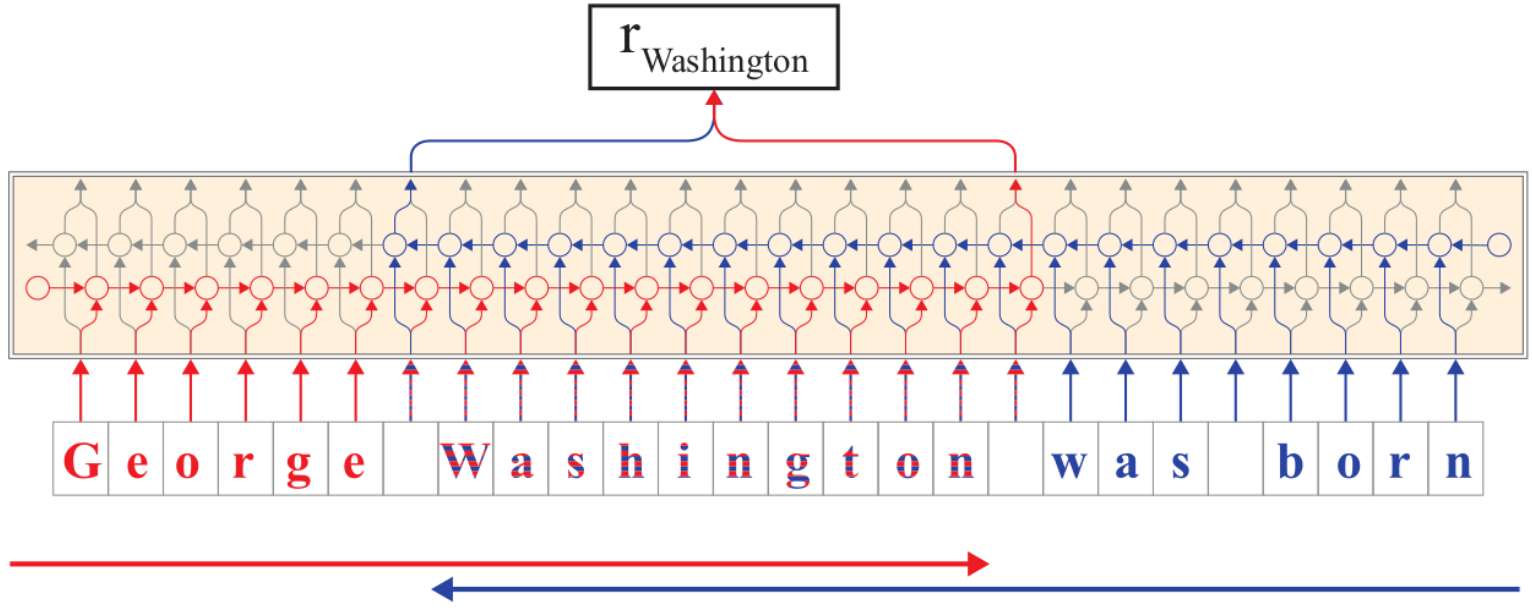
```
('Piątek', '4', 'nsubj')
('był', '4', 'cop')
('trzecim', '4', 'amod')
('dniem', '0', 'root')
('w', '7', 'case')
('tym', '7', 'det')
('tygodniu', '4', 'obl')
('na', '9', 'case')
('giełdzie', '4', 'nmod')
('w', '11', 'case')
('Warszawie', '4', 'nmod')
(',', '16', 'punct')
('w', '14', 'case')
('którym', '16', 'obl')
('WIG20', '16', 'nsubj')
('spadł', '11', 'acl:relcl')
('o', '18', 'case')
('więcej', '16', 'obl')
('niż', '21', 'case')
('4', '21', 'nummod')
('proc', '18', 'nmod')
('.', '11', 'punct')
```

Rozpoznawanie jednostek nazewniczych (NER)



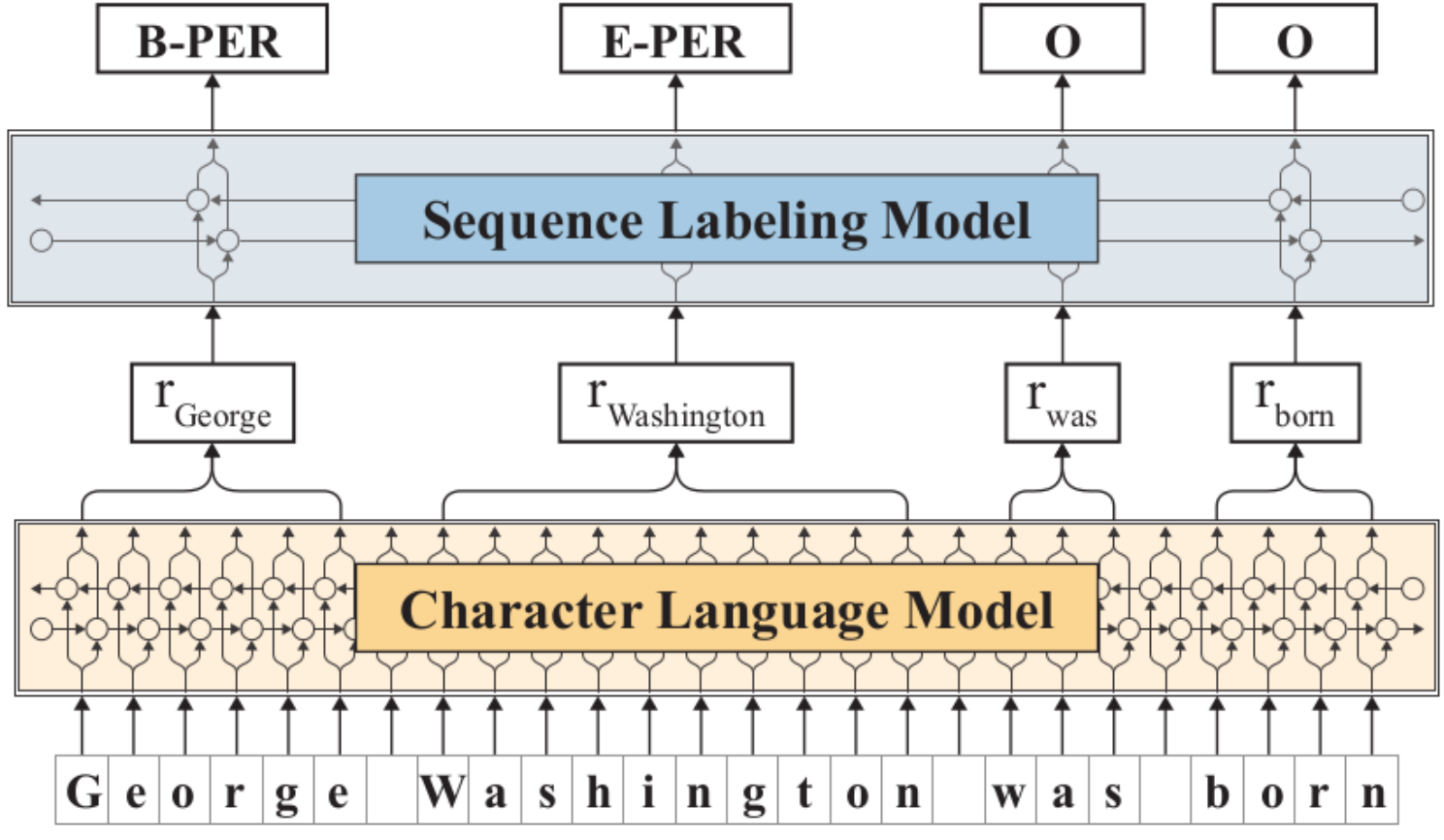
Źródło: https://demo.allennlp.org/named-entity-recognition (https://demo.allennlp.org/named-entity-recognition)

Flair 1/2



Źródło: Akbik, A., Blythe, D. and Vollgraf, R., 2018. Contextual string embeddings for sequence labeling. In Proceedings of the 27th International Conference on Computational Linguistics (pp. 1638-1649).

Flair 2/2



Źródło: Akbik, A., Blythe, D. and Vollgraf, R., 2018. Contextual string embeddings for sequence labeling. In Proceedings of the 27th International Conference on Computational Linguistics (pp. 1638-1649).

Narzędzia

J. polski

- Flair - https://github.com/zalandoresearch/flair (https://github.com/zalandoresearch/flair)
- Stanford NLP - https://stanfordnlp.github.io/stanfordnlp/installation_download.html (https://stanfordnlp.github.io/stanfordnlp/installation_download.html)
- PolDeepNer - https://github.com/CLARIN-PL/PolDeepNer (https://github.com/CLARIN-PL/PolDeepNer)
- ApplicaAI (na bazie Flair) - https://github.com/applicaai/poleval-2018 (https://github.com/applicaai/poleval-2018)
- Liner2 - https://github.com/CLARIN-PL/Liner2 (https://github.com/CLARIN-PL/Liner2)

Narzędzia cd.

J. angielski

Modelowanie języka

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Figure 3.1 Bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Zero counts are in gray.

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Figure 3.2 Bigram probabilities for eight words in the Berkeley Restaurant Project corpus of 9332 sentences. Zero probabilities are in gray.

Generowanie tekstu

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

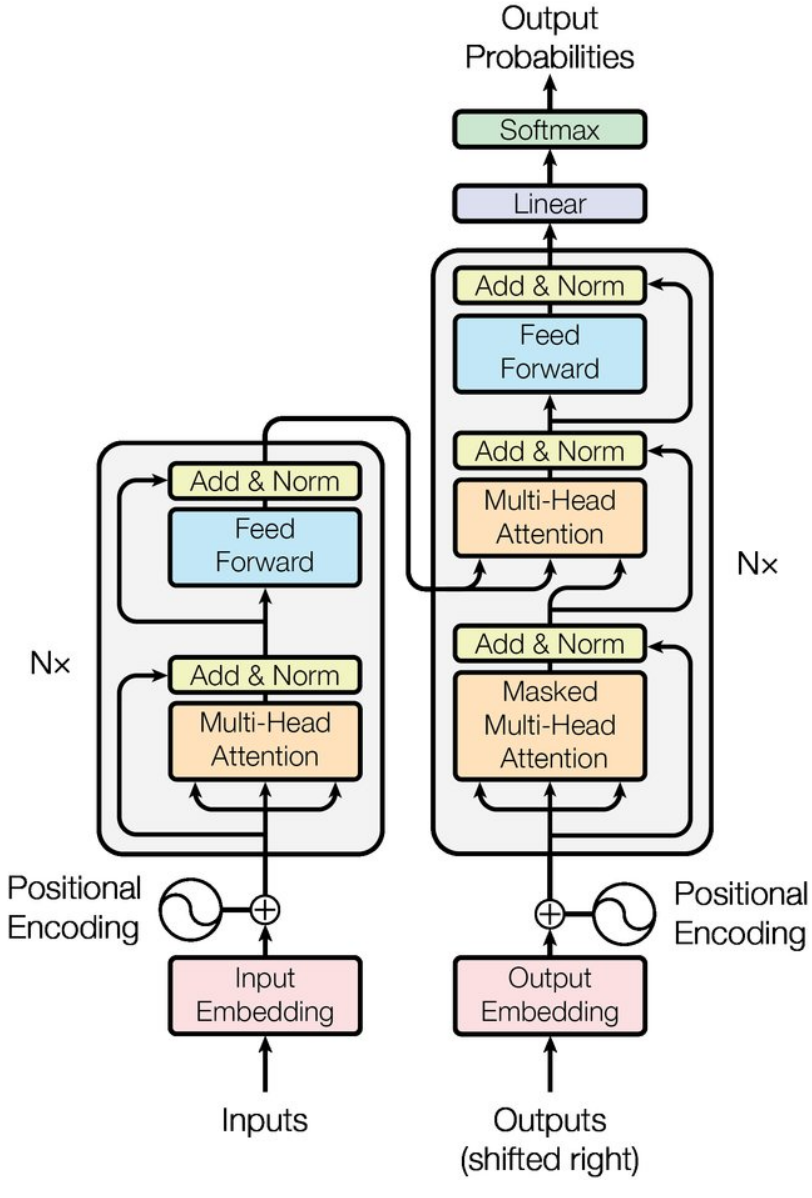
Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez.

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.

While examining these bizarre creatures the scientists discovered that the creatures also spoke some fairly regular English. Pérez stated, "We can see, for example, that they have a common 'language,' something like a dialect or dialectic."

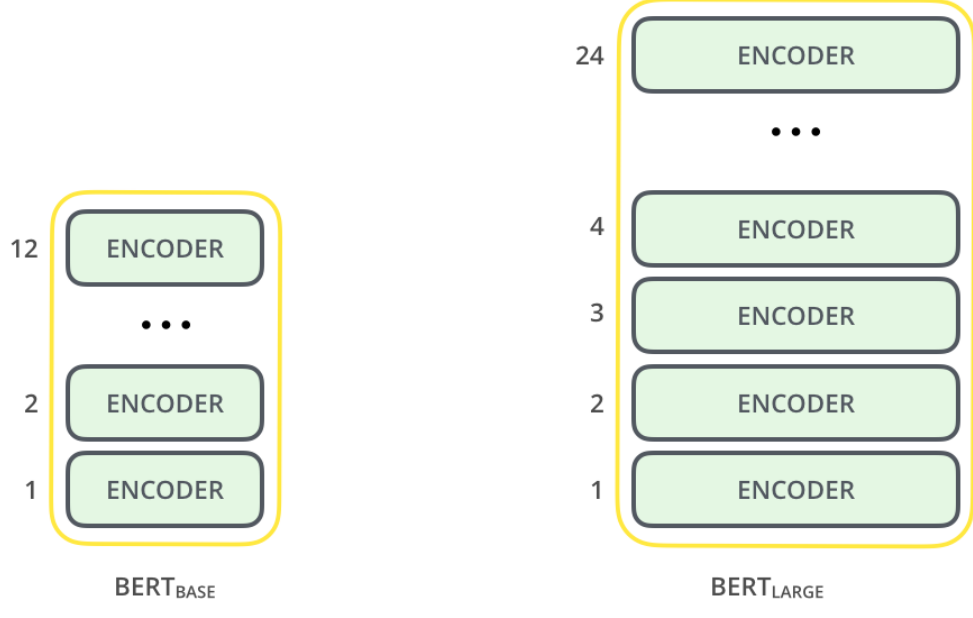
Źródło: "Language Models are Unsupervised Multitask Learners" A. Radford, J. Wu, R. Child, D. Luan,D. Amodei, I. Sutskever

Architektura transformer



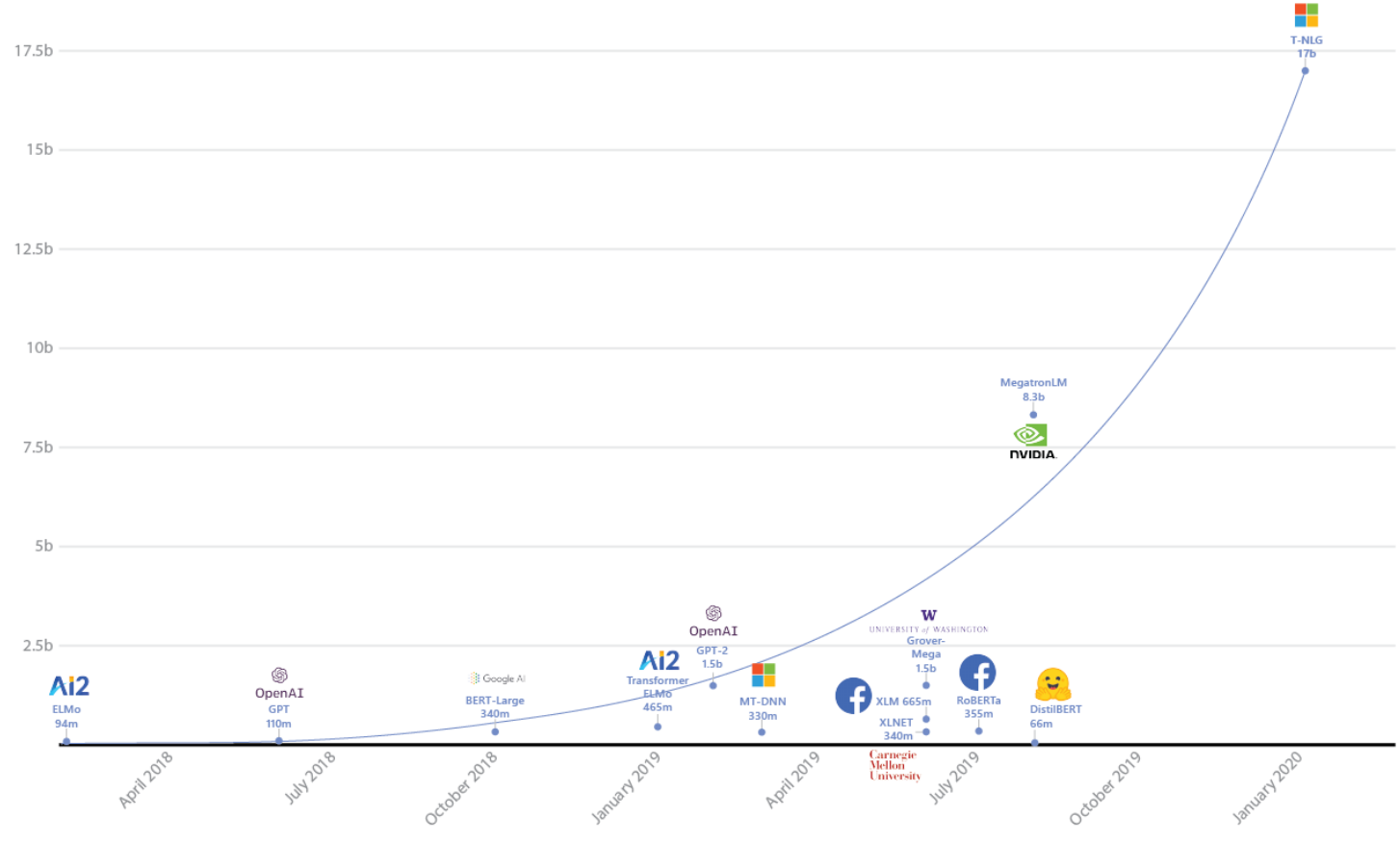
Źródło
https://www.researchgate.net/publication/323904682_Tensor2Tensor_for_Neural_Machi
lo=1
(https://www.researchgate.net/publication/323904682_Tensor2Tensor_for_Neural_Mach
lo=1)

Architektura BERTa



Źródło: [http://jalammar.github.io/illustrated-bert/ (http://jalammar.github.io/illustrated-bert/)](http://jalammar.github.io/illustrated-bert/)

Turing NLG

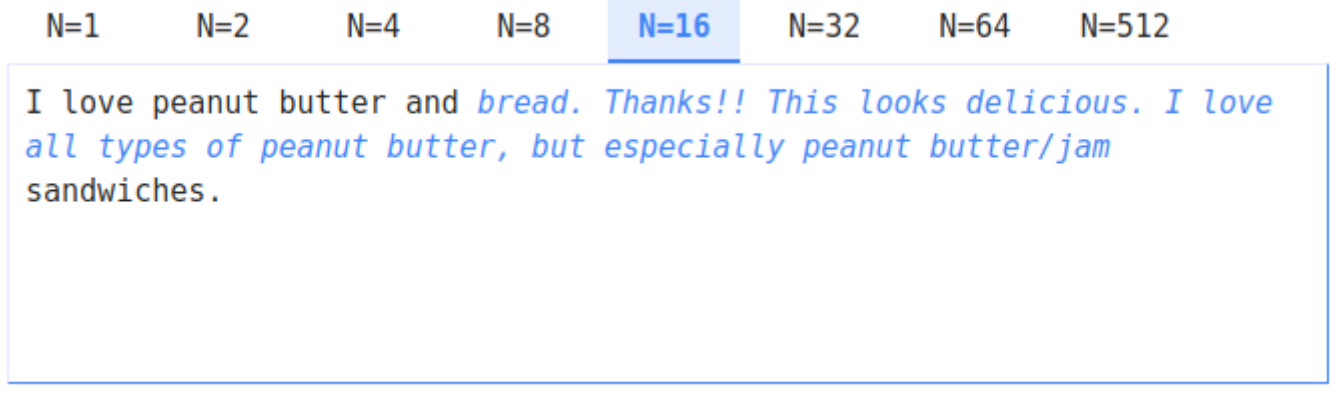


Źródło: [https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/ (https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/)](https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/)

Narzędzia:

- SRI LM - http://www.speech.sri.com/projects/srilm/ (http://www.speech.sri.com/projects/srilm/)
- Flair - https://github.com/zalandoresearch/flair (https://github.com/zalandoresearch/flair)
- ULMFiT - http://nlp.fast.ai/ (http://nlp.fast.ai/)
- BERT - https://github.com/google-research/bert (https://github.com/google-research/bert)
- ELMo - https://allennlp.org/elmo (https://allennlp.org/elmo)

Model T5



https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html (https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html)

Zastosowania

Praca z danymi tekstowymi

1. zebranie danych tekstowych
2. normalizacja danych tekstowych
3. anotacja danych
4. budowa modelu
5. testy
6. wdrożenie

Papers with code

Natural Language Processing

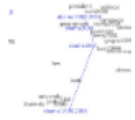
429 leaderboards • 232 tasks • 100 datasets • 3560 papers with code

Representation Learning

Representation Learning

3 leaderboards

439 papers with code



Word Embeddings

426 papers with code

Graph Embedding

90 papers with code

See all 17 tasks

Machine Translation

Machine Translation

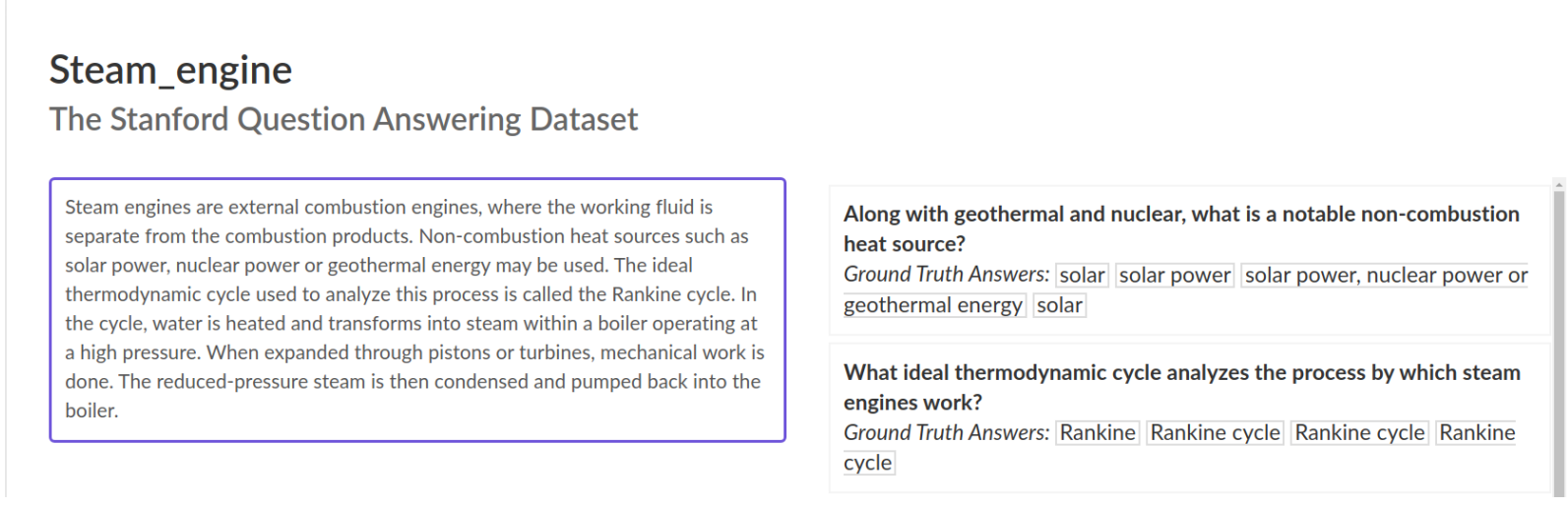
44 leaderboards

Transliteration

Unsupervised Machine Translation

9 leaderboards

SQuAD



źródło: [https://rajpurkar.github.io/SQuAD-explorer/explore/v2.0/dev/Steam_engine.html (https://rajpurkar.github.io/SQuAD-explorer/explore/v2.0/dev/Steam_engine.html)](https://rajpurkar.github.io/SQuAD-explorer/explore/v2.0/dev/Steam_engine.html)

Zasoby