Sumaryzacja (ekstraktywna); Dependency parsing; Tokenizacja do postaci subtokenów

mgr inż. Dawid Wiśniewski - Przetwarzanie języka naturalnego

18.05.2021

Sumaryzacja ekstraktywna i abstraktywna i

Bigos

Bigos – tradycyjna dla kuchni polskiej, litewskiej i białoruskiej potrawa z kapusty i mięsa. Pochodzenie słowa bigos, według Andrzeja Bańkowskiego, jest niejasne. Istnieje wiele sposobów przygotowywania bigosu i jego odmian. Wszystkie opierają się na tych samych, zasadniczych składnikach, różniąc się jedynie niektórymi dodatkami i kolejnością ich dodawania. Podstawowymi składnikami bigosu staropolskiego jest drobno szatkowana kapusta kiszona, kapusta świeża (niekiedy używa się tylko kiszonej), różne gatunki mięsa i wędlin, suszone grzyby, suszone śliwki (lub wędzone), cebula i przyprawy.

Sumaryzacja ekstraktywna i abstraktywna ii

Bigos

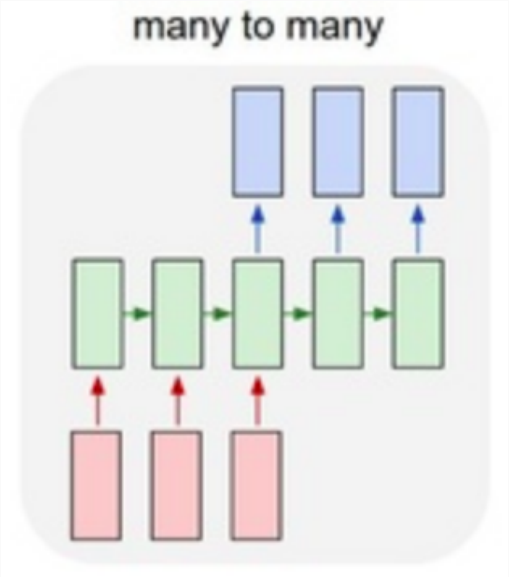
Bigos – tradycyjna dla kuchni polskiej, litewskiej i białoruskiej potrawa z kapusty i mięsa. Pochodzenie słowa bigos, według Andrzeja Bańkowskiego, jest niejasne. Istnieje wiele sposobów przygotowywania bigosu i jego odmian. Wszystkie opierają się na tych samych, zasadniczych składnikach, różniąc się jedynie niektórymi dodatkami i kolejnością ich dodawania. **Podstawowymi składnikami bigosu staropolskiego jest drobno szatkowana kapusta kiszona, kapusta świeża (niekiedy używa się tylko kiszonej), różne gatunki mięsa i wędlin, suszone grzyby, suszone śliwki (lub wędzone), cebula i przyprawy.**

Sumaryzacja ekstraktywna i abstraktywna iii

Sumaryzacja ekstraktywna -> tworzenie podsumowań poprzez wybór zdań, które niosą najwięcej informacji.

Sumaryzacja ekstraktywna i abstraktywna iv

Sumaryzacja abstraktywna -> tworzenie podsumowań, będących nowymi tekstami zawierającymi potencjalnie nowe (niewystępujące w oryginalnym tekście) słowa i zdania.



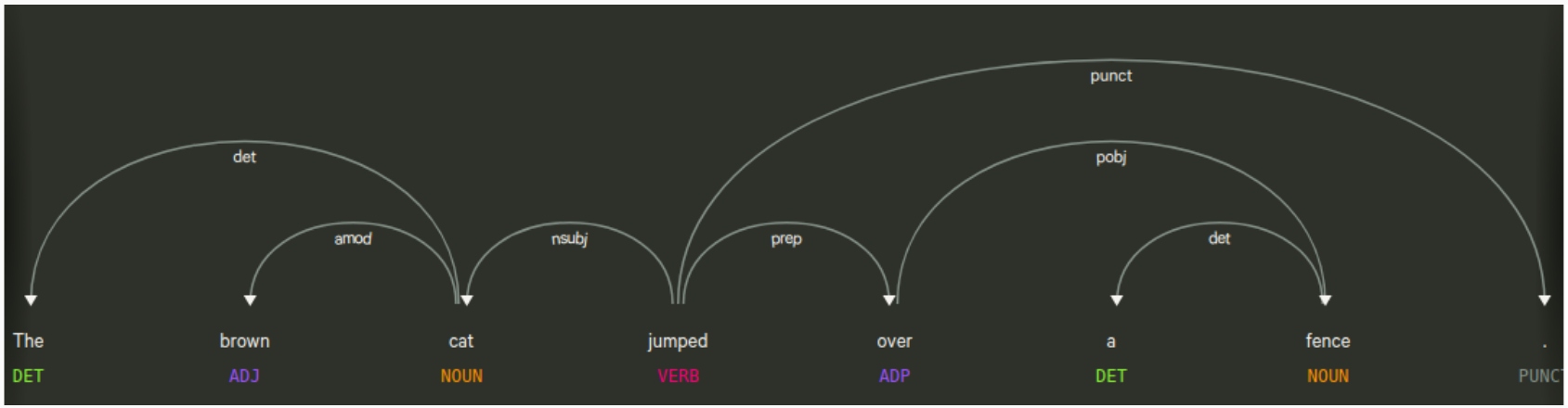
Dependency parsing/parsowanie zależnościowe i

Sekwencje POS-tagów (tagowanie częściami mowy) pozwalają nam w sposób częściowy zrozumieć teksty.

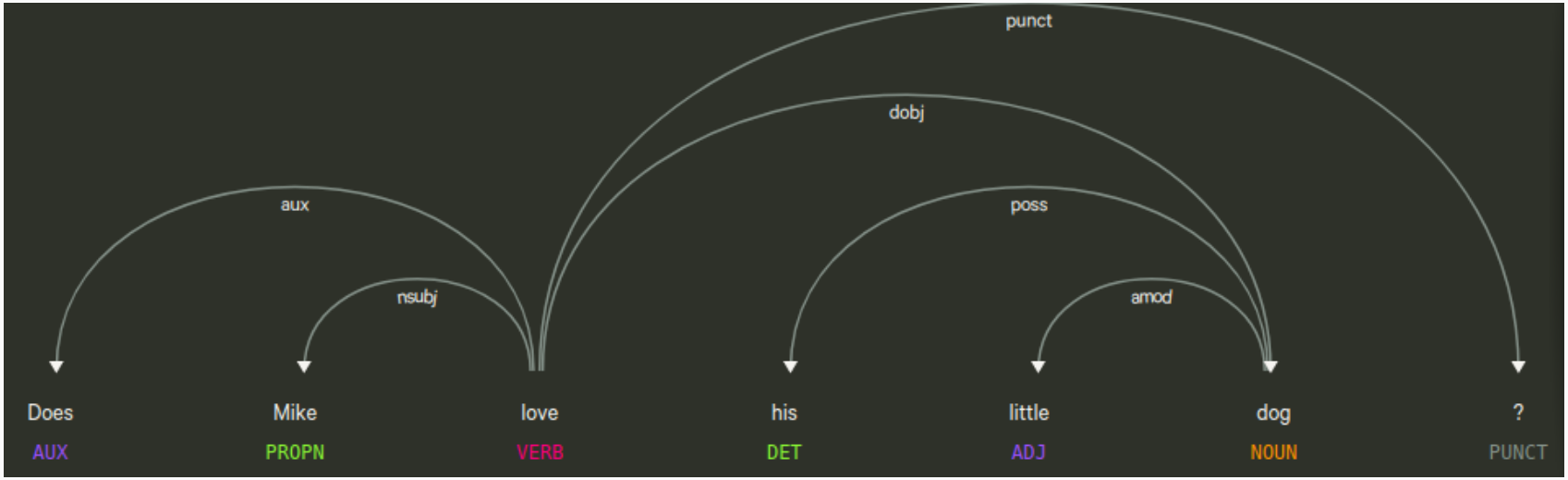
W przykładzie poniżej widzimy, że ”The brown fox”to byt (gdyż fox to rzeczownik).

Dependency parsing/parsowanie zależnościowe ii

Drzewo zależnościowe (dependency parse tree) pozwala nam odnaleźć zależności gramatyczne między słowami.

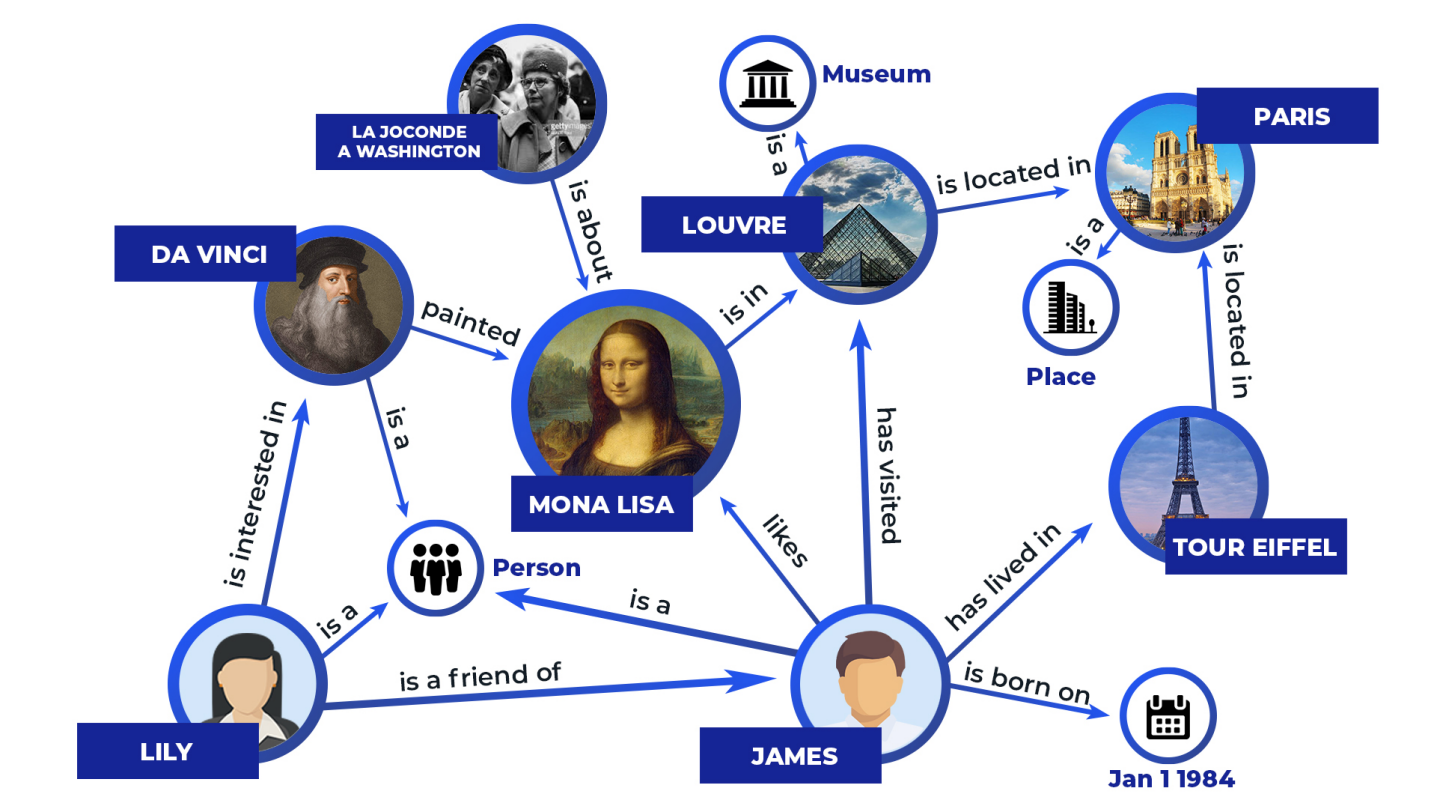


Dependency parsing/parsowanie zależnościowe iii



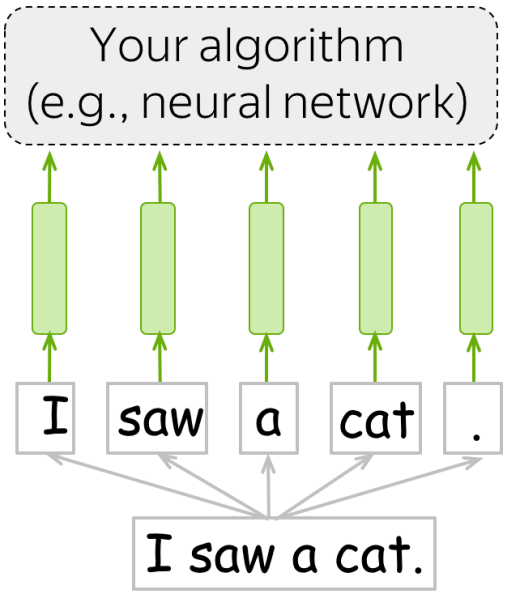
does love(Mike, his little dog)

Dependency parsing/parsowanie zależnościowe iv



(Obrazek pochodzi z: https://yashuseth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/)

Tokenizacja do postaci subtokenów i



Any algorithm for solving a task

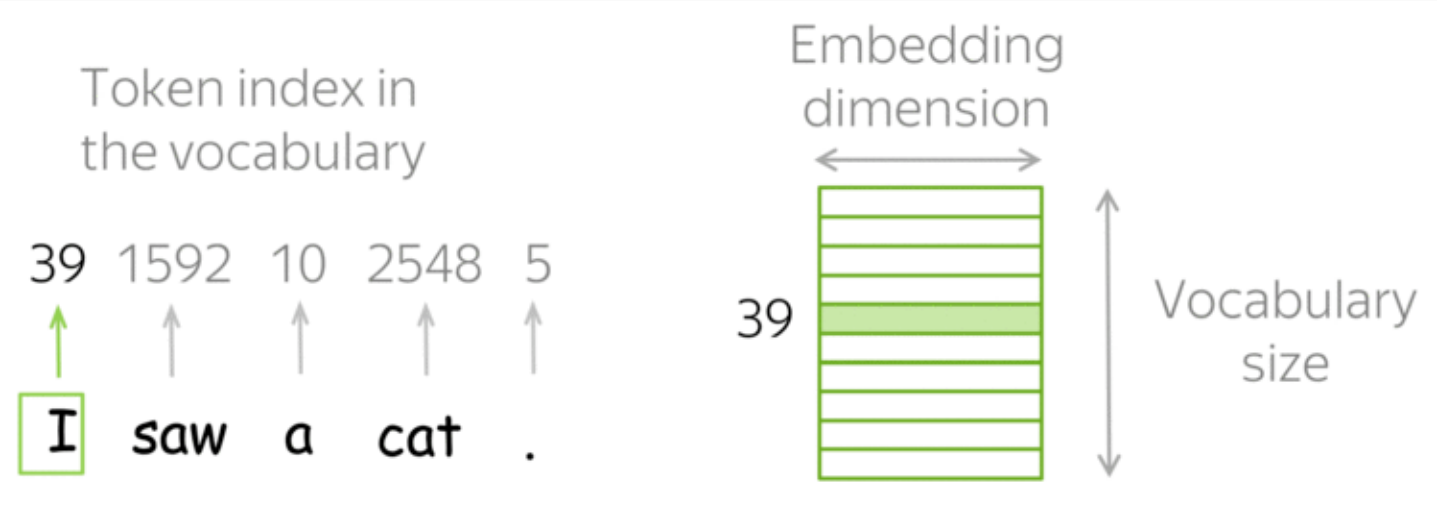
Word representation - vector
(input for your model/algorithm)

Sequence of tokens

Text (your input)

(https://lena-voita.github.io/nlp_course/word_embeddings.html)

Tokenizacja do postaci subtokenów ii



(https://lena-voita.github.io/nlp_course/word_embeddings.html)

Tokenizacja do postaci subtokenów iii

Każde słowo (indeks) zamieniane jest na reprezentację one-hot encoding (jedna jedynka na pozycji indeksu, reszta zer) w wektorze długości słownika.

Uczone (albo gotowe, np. GloVe) embeddingi znajdują się w macierzy wag między warstwami. Za pomocą kodowania one-hot możemy wskazać wiersz reprezentujący embedding danego słowa.

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

Tokenizacja do postaci subtokenów iv

Fajne, ale:

- Słownik może być bardzo duży.
- Słownik stworzony na podstawie zbioru treningowego. Co jeśli podczas używania narzędzia (np. klasyfikatora) napotkamy na słowo, które nie pojawiło się podczas treningu?

Tokenizacja do postaci subtokenów v

Podejście naiwne, jeśli pojawia się coś co nie należy do słownika, zamieniane jest na specjalny token <UNK>.

Token ten ma swój embedding, ale każde słowo spoza słownika zostanie zmapowane na taką samą reprezentację.

Tokenizacja do postaci subtokenów vi

Gdyby tylko istniał sposób reprezentowania znacznia częstych podciągów w tekstach i składania tych znaczeń w znaczenie słów.

Tokenizacja do postaci subtokenów vii

przezajefajny = prze + zaje + fajny

astrology = astro + logy

xyzfswlogy = x + y + z + f + s + w + logy

Tokenizacja do postaci subtokenów viii

Popularne podejścia:

BPE (byte pair encoding) oraz WordPiece

WordPiece wykorzystywany jest przez architekturę BERT.

Tokenizacja do postaci subtokenów ix

Poczatkowy słownik: pojedyncze znaki

Tokenizacja do postaci subtokenów x

Korpus:

’astrology and geology and analogy’

Tokenizacja do postaci subtokenów xi

Korpus:

’astrology and geology and analogy’

a s t r o l o g y </w> a n d </w> g e o l o g y </w> a n d </w> a n a l o g y </w>

Tokenizacja do postaci subtokenów xii

Korpus:

'astrology and geology and analogy'

a s t r o l o g y </w> a n d </w> g e o l o g y </w> a n d </w> a n a l o g y </w>

count(a, s) = 2

count(s, t) = 2

...

count(y, </w>) = 3 <- max

...

Tokenizacja do postaci subtokenów xiii

Zbitka y</w> jest najczęstsza, dodajemy ją do słownika i powtarzamy operację.

Tokenizacja do postaci subtokenów xiv

Korpus:

'astrology and geology and analogy'

a s t r o l o g **y</w>** a n d </w> g e o l o g **y</w>** a n d </w> a n a l o g **y</w>**

count(a, s) = 2

count(s, t) = 2

...

count(g, y</w>) = 3 <- max

...

Tokenizacja do postaci subtokenów xv

Częste zbitki iteracyjnie dodawane są do słownika tak długo, aż nie osiągniemy założonych przez nas rozmiarów słownika

Jeśli słownik będzie duży, częste słowa będą w całości reprezentowane w słowniku

Jeśli słowa w całości nie będzie w słowniku, będziemy mogli złożyć je z elementów słownika (z pojedynczych liter w worst case scenario, gdyż one tworzą wstępny słownik).

Tokenizacja do postaci subtokenów xvi

tokenize('astrology and geology and analogy') = [astro, logy, and, geo, logy, and, ana, logy]

Tokenizacja do postaci subtokenów xvii

WordPiece:

tokenize('astrology and geology and analogy') = [astro, ##logy, and, geo, ##logy, and, ana, ##logy]

Tokenizacja do postaci subtokenów xviii

Specjalne sekwencje (##) wskazują jak odtworzyć tokeny z subtokenów:

(WordPiece) token zaczyna się na ## - wcześniejszy subtoken będzie częścią tokenu

(WordPiece) detokenize(astro, ##logy) = astrology

detokenize(and this) = and this

Tokenizacja do postaci subtokenów xix

https://huggingface.co/transformers/tokenizer_summary.html

https://colab.research.google.com/drive/1fCKIBJ6fgWQ-f6UKs7wDTpNTL9N-Cq9X

https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/22-tokenization

https://youtu.be/zJW57aCBCTk