Przetwarzanie języka naturalnego
Pracownia 1
Zajęcia 1 i 2

W zadaniach z tej listy powinieneś korzystać z korpusu ze strony: `http://poleval.pl/tasks#task3` (wygodna jest wersja ztokenizowana). Korpus ten został skopiowany do kartoteki `/pio/data/data/poleval_polish_corpora/`, tam też są policzone statystyki 2-gramów i 3-gramów (będą też 4 i 5 gramy w przyszłości).

Zadanie 1. (**3+1p**) W zadaniu tym powinieneś sprawdzić, jak skuteczny jest algorytm MaxMatch[1] do tokenizacji tekstu polskiego, w którym usunęliśmy spacje i zamieniliśmy wszystkie litery na małe. W tym celu:

a) zaproponuj miarę podobieństwa otrzymanej tokenizacji z tokenizacją początkową (daną w tekście, bowiem tokeny oddzielone są spacjami)

b) sprawdź, jaką efektywność w tej mierze osiąga algorytm MaxMatch (korpus jest duży, wydaje się że wzięcie nawet niewielkiej, reprezentatywnej jego części, pozwoli wiarygodnie oszacować skuteczność algorytmu),

c) zaproponuj dowolny inny algorytm tokenizacji, porównaj jego skuteczność z MaxMatch (+1 punkt)

Wynikowa tokenizacja powinna składać się z legalnych słów. Przyjmiemy, że każde słowo występujące w korpusie jest legalnym słowem.

Do zadanie tego będziemy jeszcze wracać, rozważając inne metody, będzie też punktowana jakość rozwiązań – ale to na kolejnych listach.

Zadanie 2. (**3p**) W zadaniu tym powinieneś skorzystać z bigramów (oraz trigramów) z korpusu NKJP (czyli pliku, który zawiera informacje, ile razy w korpusie znajduje się dana para lub trójka wyrazów). Powinieneś napisać program, który generuje ciągi niby-zdań, korzystając z ngramów. Algorytm opiszemy w wersji bigramowej:

> Należy utworzyć strukturę, która dla każdego słowa pamięta listę jego możliwych następników; generacja przebiega tak, że zaczynamy od wybranego losowo słowa (albo od słowa `<BOS>`, losujemy dla niego następnik, dla niego kolejny następnik etc. Gdy dojdziemy do słowa bez następnika, to rozpoczynamy losowanie od początku. Oczywiście kolejne słowa należy wypisywać.

Napisz program generujący teksty dla bigramówi i trigramów (jeżeli będziesz miał kłopoty ze zmieszczeniem się w pamięci operacyjnej, wówczas ogranicz się do N-gramów występujących co najmniej K razy, dla odpowiednio dobranego $K > 1$. Przeanalizuj otrzymane teksty.

Zadanie 3. (**4p**) Powtórz poprzednie zadanie, ale tym razem powinieneś losować kolejne słowa nie z jednorodnym rozkładem, ale w ten sposób, by prawdopodobieństwo wylosowania następnika było proporcjonalne do liczby wystąpień odpowiedniego N-gramu w korpusie. Czy otrzymujesz w ten sposób bardziej naturalne teksty?

Zadanie 4. (**3p**) Rozważmy trzy zdania:

> Judyta dała wczoraj Stefanowi czekoladki.
> Babuleńka miała dwa rogate koziołki.
> Wczoraj wieczorem spotkałem pewną piękną kobietę

Zwróć uwagę, że w języku polskim, dla każdego z tych zdań bardzo wiele z wszystkich możliwych permutacji wyrazów to zdania mniej lub bardziej poprawne, choć czasem mocno nacechowane stylistycznie, jak przykładowo:

> Dwa miała rogate koziołki babuleńka

[1] Był na wykładzie, slajd N

Dodaj kilka zdań do powyższej trójki. Twoim zadaniem jest napisanie programu, który dla danego (multi)zbioru wyrazów (na tyle małego, że można przejrzeć szybko wszystkie permutacje) wypisuje kilka przykładowych uszeregowań tych wyrazów, w kolejności od najbardziej naturalnego. Naturalność powinieneś oceniać patrząc na statystyki występowania N-gramów. W tym zadaniu Twój program może założyć, że ma działać na *predefiniowanych zestawach* – co oznacza, że możesz wczytać z N-gramów tylko informacje o niewielkiej liczbie wyrazów (i na pewno nie będziesz miał kłopotów ze zmieszczeniem się w pamięci).

Zadanie 5. (**4p**) W tym zadaniu rozwiązujemy analogiczny problem jak w powyższym (czyli porządkujemy wyrazy), ale tym razem:

a) nie zakładamy niczego o słowach do uporządkowania

b) chcemy poradzić sobie z „dziurawością" danych, czyli tym, że o pewnych „dobrych" połączeniach w korpusie nie ma żadnych informacji (na przykład nie ma nic o „rogatych koziołkach", ale są za to „jajowate dołki", czy „garbate aniołki").

Na SKOS-ie będzie opisany dodatkowy test, który powinno przejść nasze rozwiązanie.