Przetwarzanie języka naturalnego
Pracownia 2
Zajęcia 4 i 5

(lista jest trochę dłuższa, zakłada że ewentualne zajęcia 12 listopada mają formę konsultacji i nie liczą się jako termin). Ponadto można wydłużyć termin wybranego zadania.

Zadanie 1. (**3+1+Xp**) Napisz program, który ustala autora zdania. Autorów jest trójka: Prus, Orzeszkowa, Sienkiewicz (POS), na SKOSie znajdziesz odpowiednie dane uczące. Powinieneś je podzielić na część uczącą i walidacyjną. Część X punktacji zależeć będzie od tego, jak Twój program wypadnie na tle innych programów dla danych testowych (które pojawią się później na SKOSie). Dwie podstawowe metody są następujące:

a) Naive Bayes (będzie na wykładzie 4, powinieneś użyć przynajmniej jednej cechy, która nie jest *słowem*)

b) Utworzenie trzech modeli językowych i wybór tego, który daje najepszy wynik na klasyfikowanym zdaniu.

Za sprawdzenie dwóch podejść jest punkt premiowy.

Zadanie 2. (**3+Xp**) W zadaniu tym powinieneś uporządkować (spermutować) ciąg słów, żeby utworzył zdanie. W stosunku do poprzedniej listy mamy następujące różnice:

a) powinieneś wykorzystać tagi słów (na przykład z pliku supertags.txt, link na SKOSie)

b) powinieneś je połączyć ze zwykłymi statystykami bigramowymi (lub, opcjonalnie, z sufiksami)

c) powinieneś wyodrębnić z danych uczących część walidacyjną i dobrać parametry łączenia modeli bazujących na słowach i bazujących na tagach.

Dane uczące (i walidacyjne) to korpus PolEval. Zadanie będzie oceniane na podzbiorze części testowej korpusu PolEval, w której znajdą się zdania o długości od 4 do 8 wyrazów, zawierające jedynie słowa z pliku supertags.txt (oraz interpunkcję). Miarą sukcesu dla pojedynczego zdania jest współczynnik Jackarda[1] dla bigramów słów (wliczając BOS i EOS), miarą sukcesu dla zbioru zdań jest uśredniona wartość tego współczynnika.

Zadanie 3. (**4p**) W zadaniu tym powinieneś losować zdania o słowach z identyczną charakterystyką gramatyczną jak zdanie wejściowe). Przykładowo dla zdania:

> Mały Piotruś spotkał w niewielkiej restauracyjce wczoraj poznaną koleżankę.

wynikiem mogłoby być

> Gruby Stefan przeczytał we wczorajszej gazecie starannie przygotowaną analizę.

Zgodność gramatyczną sprawdzamy za pomocą tagów z pliku supertags. Przyjmijmy, że słowo `s` niewystępujące w tym pliku ma opis gramatyczny `('^' + s)[-3:]`. Powinieneś korzystać ze statystyk unigramowych.

Zadanie 4. (**3p**) Dodaj statystyki bigramowe do powyższego zadania. Postaraj się, by jak najrzadziej zdażały się sytuacje, w których musisz losować posługując się unigramami. Zaznaczaj znakiem `"|"` każdą taką nieciągłość.

Zadanie 5. (**5+1p**) W zadaniu tym zajmiemy się kolokacjami słów, o których mamy informacje gramatyczną. Napisz program, który dla danego słowa znajduje k najbardziej z nim „spokrewnionych" słów. Rozważ następujące metody wyznaczania kolokacji:

a) PPMI (Positive Pointwise Mutual Information)

[1] Współczynnik Jackarda dla dwóch zbiorów to stosunek wielkości części wspólnej tych zbiorów, do sumy mnogościowej tych zbiorów.

b) jakiś inny, dowolnie wybrany, z używanych na kolokacje wzorów (więcej na wykładzie),

c) kolokacje gramatyczno-słowowe (tzn. żeby dwa słowa były uznane za kolokacje, warunek kolokacyjności (dowolnie wybrany) powinny spełniać zarówno tagi słów, jak i same słowa.

d) jakaś dowolna inna metoda, lub Twoja modyfikacja powyższych (za to dodatkowy punkt).

Możesz się ograniczyć do słów, które są stosunkowo częste (więcej niż n wystąpień w korpusie) i występują co najmniej raz w jakimś trigramie (lub k razy w jakimś bigramie). Wybierz niewielki zbiór słów (powiedzmy koło 10). Przygotuj raport, w którym dla każdego z tych słów jest 10 najbardziej spokrewnionych słow (czyli takich, o największym współczynniku kolokacji), dla różnych metod wyznaczania kolokacji.

Zadanie 6. (**7p**) W zadaniu tym będziemy tworzyć pierwszą wersję programu tworzącego poezję (przypominającą Pana Tadeusza, oznaczanego dalej PT)). Przypomnijmy najważniejsze fakty odnoszące się do tego utworu (i ogólnie Poezji[2]):

F1. Wiersz składa się z wersetów, z których każdy ma ustaloną liczbę sylab (w PT to 13)

F2. Ostatnie słowo w wersecie n rymuje się z ostatnim słowem w wersecie $n+1$ (dla n parzystego, numeracja od 0)[3].

F3. Rym to zgodność ostatniej sylaby i części od samogłoski sylaby przedostatniej (zdr**owie**-d**owie**). W zasadzie rymy to zjawisko fonetyczne, ale dla języka polskiego (w pierwszej wersji) można sobie to trochę uprościć i powiedzieć, że dotyczą one liter.

F4. Akcenty słów i podziały słów muszą się jakoś sensownie układać. Nam wystarczy przyjąć, że godzimy się jedynie na takie podziały wersu na słowa k-sylabowe[4], których użył Adam Mickiewicz, na przykład:

> Litwo, Ojczyzno moja, Ty jesteś jak zdrowie, ile cię trzeba cenić, ten tylko się dowie

ma schemat: `[2,3,2,1,2,1,2] -- [2,1,2,2,1,2,1,2]`

F5. Podział na sylaby nie jest trywialny (dlaczego?). Ale na nasze szczęście *policzenie* sylab jest łatwe. Słowo ma tyle sylab, ile ma samogłosek (przy czym połączenia ie, iu, ię, itd traktujemy jako jedną samogłoskę). Część rymowana wyrazu to część wyrazu od przedostatniej samogłoski do końca.

Powinieneś stworzyć program, generujący dwuwersowe fragmenty wierszy w stylu PT, czyli powinieneś przypilnować:

a) żeby wersy były poprawne rytmicznie i się rymowały,

b) żeby dwuwers miał sens gramatyczny (czyli by tagi słów pasowały do jakiegoś zdania lub fragmentu zdania)

c) żeby były jakoś wykorzystane statystyki N-gramowe (plan minimum to statystyki 1-gramowe, dodatkowe +1 za wykorzystanie bigramów).

Zadanie 7. (**5p**) Zmodyfikuj algorytm z poprzedniego zadania w ten sposób, by starał się on maksymalizować wzajemną „kolokacyjność" słów z dwuwersu (tak, by jak najwięcej słów było ze sobą powiązanych, na przykład przez wysokie PPMI). Akceptowalna jest dowolna procedura (local search, jakieś błądzenie losowe, metody ewolucyjne, ...), która daje wartość liczby par słów będących kolokacjami istotnie większą niż losowanie z poprzedniego zadania.

[2] Bardzo proszę nie pokazywać tego żadnemu Poloniście ani Polonistce

[3] Jest to najprostszy schemat rymów aabb. Oczywiście są też inne, np. abab, ale na razie nimi się nie zajmujemy

[4] Korzystamy tu istotnie z tego, że z dobrym przybliżeniem, w języku polskim akcent wyrazowy jest funkcją liczby sylab