Przetwarzanie Języka Naturalnego – konspekt (30 godzin)
Dr inż. Krzysztof Rzecki

Literatura:

- W. Lubaszewski, „Słowniki komputerowe i automatyczna ekstrakcja informacji z tekstu", AGH Kraków 2009.
- Kłopotek M., „Inteligentne wyszukiwarki internetowe", Akademicka Oficyna Wydawnicza Exit, Warszawa 2001.
- Saloni Z., Świdziński M., „Składnia współczesnego języka polskiego", Wyd. V., Warszawa 2001.
- Daciuk J., zbiór informacji na http://www.eti.pg.gda.pl/~jandac/.
- Gusfield D., „Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology", Cambridge University Press, 1997.
- Jurafsky D., Martin J.H., „Speech and language processing, An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition", Prentice-Hall, Inc., Upper Saddle River, New Jersey 07458, 2000.

W trakcie zajęć wszystkie zadania do zrealizowania są wyjaśniane, analizowane na przykładach, a także pokazywane są modelowe rozwiązania w celu wyrobienia intuicji, jaki jest oczekiwany wynik danego ćwiczenia.

1. Wprowadzenie do języka Perl, wyrażenia regularne.
Cele ćwiczenia:
- wprowadzenie do programowania w języki Perl podstawowych programów,
- zapoznanie się z podstawowymi strukturami danych w języku Perl,
- opanowanie metodologii układania wyrażeń regularnych.

Plan ćwiczenia:
- napisać skrypt w języku Perl realizujący podstawowe operacje na tekście (wprowadzanie, wycinanie, zliczanie, zmiana kolejności),
- napisać skrypt w języku Perl wyszukujący zadane ciągi znaków w tekście,
- napisać skrypt w języku Perl zamieniający w zadanym pliku tekstowym kolejność słów w każdej linii.

2. Zaawansowane typy danych i ich persystencja.
Cele ćwiczenia:
- opanowanie operowaniem zaawansowanymi strukturami danych w języku Perl,
- zapoznanie się z technikami realizacji persystencji struktur danych.

Plan ćwiczenia:
- zaprojektować strukturę danych do przechowywania hash'a tablic,
- zaprojektować i zaimplementować strukturę drzewa binarnego,
- sprawdzić działanie exists(), defined(),
- zaimplementować program wczytujący drzewo binarne do pamięci i zapisujący całą strukturę do pliku na dysku; następnie wczytać tę strukturę i sprawdzić poprawność.

3. Parsowanie dokumentów tekstowych i analiza leksykalna.
Cele ćwiczenia:
- zaprojektowanie wyrażeń regularnych do parsowania dokumentów tekstowych:
- implementacja analizatora leksykalnego,
- przechowywanie przetworzonych danych w pamięci RAM i na dysku twardym.

Plan ćwiczenia:

- napisać program implementujący wyrażenia regularne wydobywające słowa (napisy składające się z liter) z pliku tekstowego przez:
  - wyrażenia regularne do wyszukania znaczników,
  - wyrażenia regularne do zamiany znaczników,
  - znaczniki pojedyncze (np.
),
  - znaczniki obejmujące tekst (np. i),
  - znaczniki agregujące (np. table, tr, td),
- rozbudować program o zamienianie wszystkich liter w napisach na małe,
- rozbudować program o przetwarzanie wszystkich plików w zadanym katalogu i zapisywanie do nowych plików samych napisów małą literą.

4. Algorytmy pozyskiwania tekstów i budowy korpusu tekstowego.

Cele ćwiczenia:

- zapoznanie się z technikami Web-crawlingu i działania robaków internetowych,
- zapoznanie się z technikami wydobywania informacji ze stron WWW.

Plan ćwiczenia:

- zaprojektować i napisać program potrafiący ‘chodzić’ po stronach WWW poprzez zawarte na kolejnych stronach linki do innych stron internetowych,
- zaprojektować i napisać program parsujący strony HTML z portali internetowych wydobywający treściwe informacje (np. przez budowę drzewa semantycznego),
- zebrać korpus tekstów o wielkości min. 100 MB czystego tekstu (bez znaczników HTML).

5. Analiza statystyczna korpusu tekstowego.

Cele ćwiczenia:

- zbadanie własności statystycznych tekstów w języku polskim.

Plan ćwiczenia:

- przeprowadzić analizę statystyczną tekstów w języku polskim i sporządzić wykresy:
  - kołowy częstości (słowo, częstość, %),
  - przyrostu słownictwa (przyrostu słów bez powtórzeń),
  - pokrycia tekstu (czyli OX: od najczęściej występujących, OY: ile % tekstu pokrywają słowa od 0 do danej wartości).

6. Indekser dokumentów tekstowych, wyszukiwarka dokumentów.

Cele ćwiczenia:

- zapoznanie się z technikami indeksowania dokumentów tekstowych,
- konstrukcja prostej wyszukiwarki dokumentów tekstowych.

Plan ćwiczenia:

- zbudować indekser dokumentów z wagami (przez TF-IDF), tj. obliczyć wagi dla wszystkich termów przez TF-IDF,
- zaimplementować wyszukiwarkę do zbudowanego indeksera, która działa z linii komend i przyjmuje jako argument frazę do wyszukania; wyniki wyszukiwania posortować od najlepszego wyniku.

7. Automaty słownikowe.

Cele ćwiczenia:

- konstrukcja różnych algorytmów słownikowych,
- zbadanie własności struktur i algorytmów słownikowych.

Plan ćwiczenia:

- zaprojektować trzy rodzaje struktur do przechowywania słowników:
  - struktura tablicowa (jedna komórka, to jedno słowo),
  - struktura drzewiasta (kolejna pozycja litery w słowie to kolejny poziom drzewa),
  - struktura zminimalizowanego automatu skończonego.
- zaimplementować słowniki w strukturach:
  - tablicowej,
  - drzewiastej
  - w bazie danych (z i bez indeksów),
- zbadać i zilustrować:
  - wydajność czasową poszukiwania słowa w słowniku,
  - zajętość pamięciową,
  - porównać z możliwościami algorytmu Folda (zostanie podane rozwiązanie).

8. Własności stop-listy, algorytm budowy.

Cele ćwiczenia:

- zapoznanie się z własnościami stop-list,
- opracowanie stop-listy na bazie własnego korpusu.

Plan ćwiczenia:

- opracować listy standardowe:
  - słowa zdefiniowane (przysłówki, zaimki, spójniki, itp. – pełna lista zostanie podana na ćwiczeniach),
  - słowa najczęściej występujące.
- wykazać, że wśród słów z podanych list znajdują się słowa będące dobrymi kandydatami na słowa kluczowe,
- przygotować stop-listę w oparciu o współczynnik zmienności:
  - wybrać jako listę słowa o współczynniku mniejszym od 1,
  - do konstrukcji listy połączyć cały korpus w jeden plik,
  - tak połączony plik dzielić podziałami na N=10, 100, 1000, 10000,
  - nakreślić wykres, OX: średnia wystąpień, OY: odchylenie std,
  - sprawdzenie pisowni: TAK, lematyzacja: NIE.

9. Analiza morfologiczna tekstu, lematyzacja bezkontekstowa.

Cele ćwiczenia:

- poznanie własności morfologicznych słów języka polskiego,
- poznanie własności statystycznych analizy morfologicznej.

Plan ćwiczenia:

- korzystając z zadanej bazy danych SQL napisać program realizujący funkcjonalność lematyzatora słów z języka polskiego,
- wykonać lematyzację korpusu tekstów,
- zbadać procentową niejednoznaczność lematyzacji,
- zbadać i zilustrować na wykresie wydajność lematyzatora,
- zilustrować przyrost słownictwa zlematyzowanego w tekstach i porównać z wykresem przyrostu słownictwa niezlematyzowanego,
- rozbudować indekser i wyszukiwarkę dokumentów o unifikację poprzez lematyzację.

10. Binaryzacja i wektoryzacja tekstu, wskaźniki dopasowania.

Cele ćwiczenia:

- poznanie techniki wektoryzacji tekstu,
- poznanie własności statystycznych wskaźników podobieństwa.

Plan ćwiczenia:

- przygotować bazę wektorów dla tekstów z posiadanego korpusu:
  - do modelu boolowskiego,
  - do modelu wektorowego,
- baza może być przechowywana w pliku txt, jako hash lub w bazie danych,
- skonstruować wyszukiwarkę w oparciu o:
  - model boolowski, wskaźnik 'cosine',
  - model wektorowy, wskaźnik 'taxi',
- użytkowanie wyszukiwarki: jak w przypadku zadania z indekserem,
- porównywanie wyników wyszukiwania:
  - przygotować losowe frazy 1, 2, 3, ... 100 wyrazowe,
  - wykonać badanie:
    - przy pomocy wyszukiwarki opartej o TF-IDF przygotować listę wyszukań (listę dokumentów),
    - przy pomocy modelu bool i wektorowego,
    - zweryfikować kolejność na liście wyników (badanie na kolejność),
  - wykonać badanie:
    - przy pomocy każdej z trzech wyszukiwarek (TF-IDF, model bool, model wektorowy),
    - przygotować listy wyników,
    - porównać listy ze sobą (badanie na jednolitość wyników).

11. Programowanie dynamiczne i wskaźniki podobieństwa.

Cele ćwiczenia:

- poznanie technik programowania dynamicznego,
- poznanie własności programowania dynamicznego algorytmów,
- obliczanie wskaźników podobieństwa uwzględniających kolejność słów.

Plan ćwiczenia:

- zaimplementować wspólną bibliotekę algorytmów porównywania dokumentów:
  - Levenshtein,
  - LCS-substring,
  - LCS-subsequence,
- za wartości w komórkach porównywanych przyjąć słowa,
- zbadać własności algorytmów na krótkich tekstach.

12. Badanie kontekstu informacji, konstrukcja n–gramów.

- przygotować bazę bi-gramów (plik, hash lub baza danych) taką, że:
  - pierwsze słowo to słowo zlematyzowane jednoznacznie,
  - drugie słowo to słowo zlematyzowane jednoznacznie, ale posiadające formę lematyzowalną niejednoznacznie,
  - podać dla każdego bi-gramu parawdopodobieństwo wystąpienia słowa drugiego pod warunkiem wystąpienia słowa pierwszego,
- zaimplementować lematyzator kontekstowy korzystający z w/w bazy,
- jeśli podane do lematyzatora słowo ma więcej niż jeden lemat, to:
  - pobierane jest słowo występujące przed tym słowem,

    - szukane są w bazie wszystkie bi-gramy zlematyzowane pasujące do poszukiwanego bi-gramu,
- jako wynik lematyzacji wybieramy ten bi-gram, który ma największe prawdopodobieństwo wystąpienia,
- zbadać jakość lematyzacji (można podać wykres: OX: % pewności, OY: liczność przypadków).

13. Reprezentacja wiedzy, generowanie tekstu, odpowiedzi

Cel ćwiczenia:

- poznanie technik reprezentacji wiedzy,
- poznanie metod generowania tekstu,
- techniki udzielania odpowiedzi.

Plan ćwiczenia:

- zbadać własności n-gramów dla n>2,
- zaimplementować program przechowujący bazę ekspercką (szczegóły zostaną wyjaśnione w trakcie zajęć),
- opracować i zaimplementować bezkontekstowy generator tekstu.

14. Projekt i implementacja systemu przetwarzania tekstu.

Cel ćwiczenia:

- podsumowanie zadań implementacyjnych z całego semestru,
- przygotowanie pod kompletny system przetwarzania tekstu o szerokim przeznaczeniu.

Plan ćwiczenia:

- w usystematyzowany sposób uporządkować wszystkie programy wykonane w trakcie semestru, wybrać najlepsze (zdaniem studenta) rozwiązania,
- zaproponować przeznaczenie systemu przetwarzania tekstu i zintegrować programy w celu uzyskania oczekiwanego systemu.

15. Testowanie systemu przetwarzania tekstu.

Cel ćwiczenia:

- testowanie skonstruowanego systemu.

Plan ćwiczenia:

- zaproponować zestaw testów (danych treningowych) dla skonstruowanego systemu,
- przeprowadzić testy na zestawach,
- opracować wyniki i zilustrować je w sprawozdaniu.