Spis treści

Rozdział 1

Wstęp

1.1. Opis problemu

Ujednoznacznianie znaczeń słów (ang. word sense disambiguation) jest jednym z kluczowych problemów analizy semantycznej. Polega na wyznaczeniu znaczenia danego słowa w kontekście, tak aby cały kontekst był spójny. Znaczenia słów opisane są przez tak zwane synsety (zbiory synonimów, ang. synonyms sets). W tym celu stosuje się algorytmy nadzorowane i nienadzorowane. W algorytmach opartych o uczenie nadzorowane stosowane są metody uczenia maszynowego np. sieci neuronowe lub wektory maszyn wspierających. W uczeniu nienadzorowanym stosowane są zewnętrzne źródła wiedzy np. słowniki, słowosieci, grafy synsetów itp.

1.2. Idea metody

W niniejszej pracy zaprezentowano rozwiązanie oparte o algorytm nienadzorowanym. Wykorzystano słowosieć oraz graf synsetów. W tym celu zaimplementowano rozszerzenia do platformy WoSeDon, która realizuje problem ujednoznaczniania sensów słów za pośrednictwem algorytmów opartych o PageRank. Rozszerzenia skupiają się na wykorzystaniu innych miar centralności w grafach m.in. betweenness, closeness, eigenvector. Poniżej przedstawiono szczegółowy opis metody.

1.3. Miary centralności

1.3.1. PageRank

1.3.2. Betweenness

1.3.3. Closeness

1.3.4. Pozostałe

Rozdział 2

Platforma testowa

2.1. Komponenty

Do realizacji projektu, wykorzystano szerg gotowych rozwiązań, które znacznie przyspieszyły proces implementacji. Poniżej przedstawiono istotne biblioteki, komponenty oraz elementy użyte w projekcie:

- WoSeDon - program bazowy
- Wrocław CFR Tagger - tager
- corpus2 - biblioteka dostarczająca struktur danych i metod do obsługi anotowanych korpusów.
- Korpus PWr - anotowany korpus służący do testów
- docker - środowisko uruchmieniowe WoSeDona

Głównym elementem system jest WoSeDon, program służący do ujdnoznaczniania znaczeń słów za pomocą algorytmu PageRank. Niniejszy projekt sprowadził się do zaimplementowania rozszerzeń do WoSeDona wykorzystujących inne metody pomiaru centralności.

2.2. Docker

Ze względu na dosyć żmudny proces tworzenia środowiska testowego, zadecydowano o użyciu docekra, który pozwoli na szybkie tworzenie homogenicznego środowiska na wielu niezależnych od siebie maszynach. W ramach projektu powstały dwa typy środowisk opisanych przez poniższe dockerfile'e:

- Dockerfile-arch
- Dockerfile-ubuntu

Utworzenie dwóch odrębnych dockerfile'i było wymuszone, ze względu na długi proces budowy środowiska opartego o Linkusa Arch (brak skompilowanych komponentów, długotrwałe kompilacje). Aby ominąć problem czasowy podjęto decyzję o utworzeniu środowiska opartego o Linuksa Ubuntu.

Rozdział 3

Dane

3.1. Charakterystyka

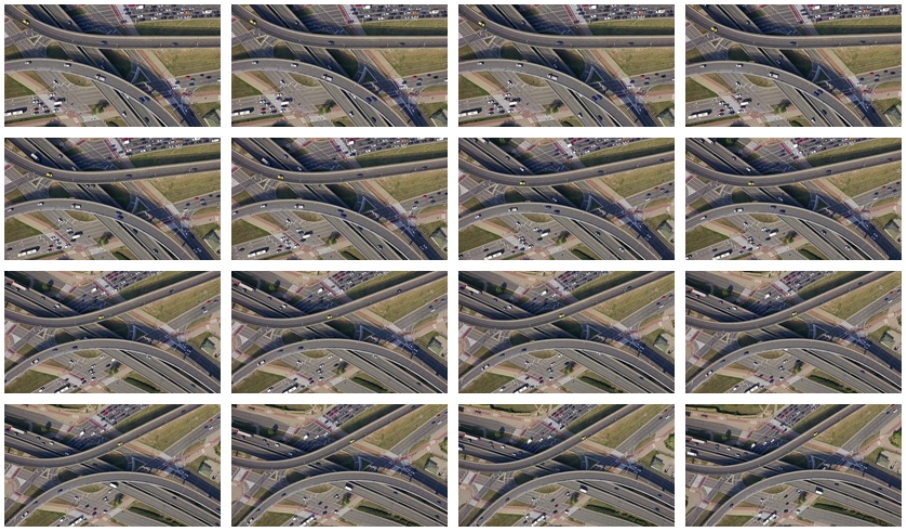
Dane pozyskane do eksperymentów pochodziły z ogólnodostępnych portali takich jak YouTube. Wybór sekwencji wideo dokonano tak, aby realizowany algorytm jak najlepiej spełniał swoje funkcje. W tym celu starano się spełnić następujące ograniczenia:

- Ujęcia pochodzące kamery, której pozycja nie zmienia się w dużym stopniu w czasie. Aczkolwiek możliwe są jej obroty o dowolny kąt.
- Ujęcia z dużą ilością elementów poruszających się tj. samochody, piesi, statki itp.
- Ujęcia ruchome o możliwie niskiej dystorsji soczewki (zaproponowane rozwiązanie całkowicie pomija to zagadnienie)

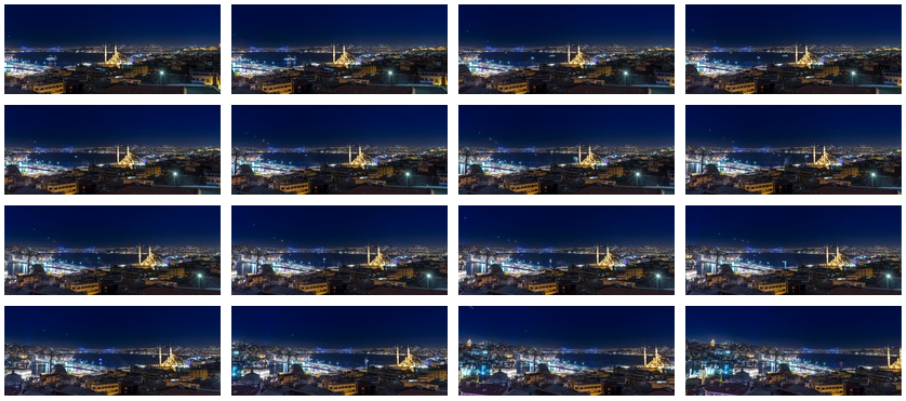
3.2. Przykłady

Rys. 3.1: Przykładowa sekwencja zdjęć 'ski'.

Rys. 3.2: Przykładowa sekwencja zdjęć ’roundabout’.



Rys. 3.3: Przykładowa sekwencja zdjęć ’warsaw’.



Rys. 3.4: Przykładowa sekwencja zdjęć ’istanbul’.

Rozdział 4

Eksperymenty

4.1. Metodologia

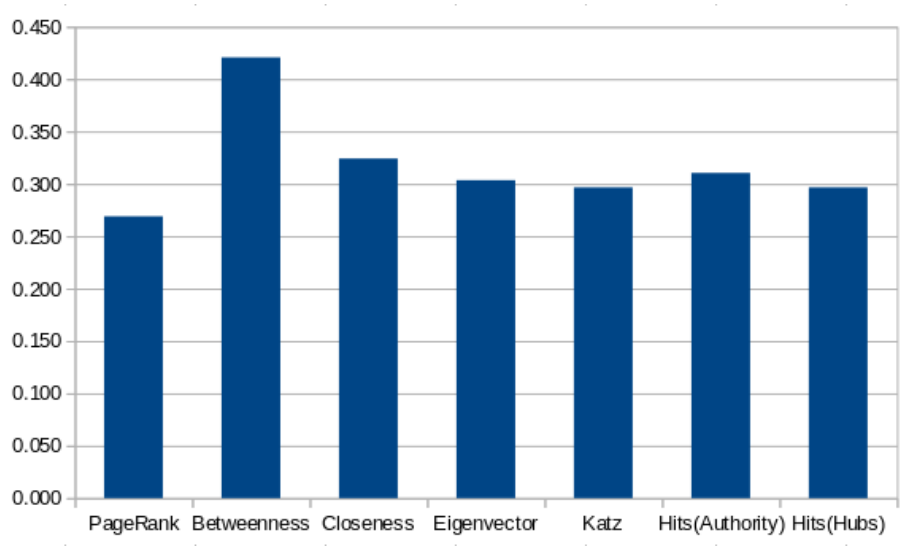
Testy przeprowadzono przy użyciu skryptu zawartego w repozytorium wosedona. Skrypt `evaluation-kpwr.py` wymaga do swojego działania połączenia z bazą danych Słowosieci. Na wejście skryptu podawany jest wcześniej przetworzony przez wosedon, anotowany tekst. Wymóg anotowanego tekstu jest spowodowany tym, że musi istnieć podstawa, na której zostanie określona poprawność ujednoznaczniania. Testy wykonano na wybranych wcześniej pięciu plikach, które zawierają najwięcej wieloznacznych wyrazów, tak aby pliki wynikowe miały jak najwięcej przetworzonych wyrazów. Każdy z siedmiu zaimplementowanych algorytmów został uruchomiony na tych pięciu plikach i już zbiorczo oznaczono ilość poprawnie ujednoznacznionych wyrazów. Do usprawnienia testowania implementacji posłużono się trzema skryptami: `runDocker` - uruchamiający kontener Dockera, `runWosedon` uruchamiający wosedon na kolejnych algorytmach na wszystkich wybranych plikach oraz `runTests`, który przetwarzał wyniki poprzedniego skryptu i oznaczał poprawnie ujednoznacznione wyrazy.

4.2. Wyniki

Poniżej przedstawiono wyniki przeprowadzonych eksperymentów.

Tab. 4.1: My caption

Nazwa	Ilość poprawnych	Ilość niejednoznacznych	Dokładność
PageRank	39	145	0.269
Betweenness	61	145	0.421
Closeness	47	145	0.324
Eigenvector	44	145	0.303
Katz	43	145	0.297
Hits(Authority)	45	145	0.310
Hits(Hubs)	43	145	0.297



Rys. 4.1: Wyniki poszczególnych metod.

Rozdział 5

Podsumowanie

5.1. Wnioski

Zaproponowana metoda daje relatywnie dobre wyniki. Jakość obrazu wynikowego istotnie zależy od wstępnego dopasowania poszczególnych obrazów. Najlepsze efekty uzyskano dla serii obrazów o stałej ekspozycji albo bardzo dużych serii. Warto również zwrócić uwagę na problemy związane z dystorsją, które również mają wpływ na efekt końcowy. Dużym atutem zaproponowanego algorytmu jest prostota i szybkość implementacji.

5.2. Krytyka

Głównym mankamentem wykonanego projektu jest jego czas działania. Przetworzenie kilkuset zdjęć o średniej rozdzielczości zajmuje czas mierzony w minutach na średniej jakości sprzęcie. Głównym powodem powolnego przetwarzania jest wykonywanie obliczeń związanych z grafiką na procesorze, a nie na karcie graficznej.

Inną wadą jest ubogi zestaw zaproponowanych funkcji. Ich działanie opiera się na prostej analizie statystycznej. Dodatkowo warto zauważyć, że zaproponowany algorytm zależy od wielu czynników (przestrzeń barw, kolejność funkcji, poszczególne parametry) co powoduje z jednej strony problemy w dostosowaniu do odpowiedniego obrazu, z drugiej jednak strony daje swego rodzaju swobodę w dostosowaniu algorytmu do danych wejściowych.

5.3. Dalszy rozwój

W dalszym rozwoju projektu warto zaimplementować więcej funkcji redukujących ilość pikseli z serii. Jest to bowiem kluczowy element zaproponowanego podejścia. Przykładowo mogą to być metody oparte o analizę histogramu albo analizę różnicową. Dopracowanie tego elementu znacznie może wpłynąć na jakość uzyskiwanych obrazów.

Innym elementem wartym wspomnienia jest problem czasu wykonywania. Ze względu na dużą ilość danych, dla którego należy wykonać podobną sekwencję instrukcji idealnym rozwiązaniem wydaje się tu być wykorzystanie procesora graficznego, który został stworzony do tego typu zadań. Zrównoleglenie algorytmu na GPU znacząco przełoży się na czas wykonywania, możliwość wykorzystania bardziej skomplikowanych funkcji oraz może przyczynić się do efektywniejszego znajdowania optymalnych konfiguracji parametrów dla danej sekwencji obrazów.