

WoSeDon – Instrukcja

Paweł Kędzia, Michał Moczulski

1 lutego 2016

Spis treści

1. Wstęp	2
2. Przygotowania	2
2.1. Instalacja boosta	2
2.2. Instalacja graph-toola	2
2.3. Instalacja WoSeDona	3
3. Uruchamianie	3
4. Testy jednostkowe	4
5. Konfiguracja	4
5.1. wosedon	5
5.2. wosedon:resources	7
5.3. wosedon:build_options	8
5.4. wosedon:merge_options	9
5.5. wosedon:rerank_options	9
5.6. wosedon:wsd_alg	9
5.7. wosedon:lesk_filter	10
6. Przykłady	10
6.1. Najprostszy przypadek	11
6.2. Bez angielskiego, z sumo	11
6.3. Krawędzie i wagi	12
6.4. Lesk	13
7. Inne narzędzia w repozytorium	13
7.1. CclExtractor	13
7.2. PLWNGraphBuilder	14
7.3. Resolver	14
7.4. average_path_length.py	15
7.5. corpus_statistic.py	15
7.6. make_xml_file_from_gloss.py	15
7.7. evaluation-kpwr.py, evaluation-sz.py	15
7.8. prec_general.py	15
8. Zawartość resources	16

1. Wstęp

WoSeDon to narzędzie przeznaczone do ujednoznaczniania znaczeń słów (ang. *Word Sense Disambiguation*). Zaprojektowane zostało do działania dla języka polskiego, jednak po odpowiednim opakowaniu tagsetu, możliwe jest również działanie na innych językach. Głównym stosowanym algorytmem, który został zaimplementowany do rozstrzygania niejednoznaczności jest algorytm PageRank. Narzędzie zawiera różne modyfikacje owego algorytmu oraz umożliwia manipulację wieloma parametrami mającymi wpływ na jakość ujednoznaczniania znaczeń leksykalnych. Opracowane algorytmy opisane zostały w artykułach [4, 3, 5].

2. Przygotowania

Uwaga! Poniższe przykłady podane są dla systemu Ubuntu Linux. W przypadku korzystania z innej dystrybucji, należy zastąpić je odpowiednimi dla danej dystrybucji. Zanim przystąpimy do instalacji samego WoSeDonu musimy upewnić się iż mamy zainstalowaną bibliotekę „Graph Tool” (<https://graph-tool.skewed.de/>) z bindingami do pythona. Znajduje się ona w repozytorium najnowszych wersji Ubuntu (testowane było na 14.4 oraz 15.4), jednak nie zawsze mamy możliwość z niej skorzystania.

Jeśli rzeczywiście jej nie ma musimy zacząć od zdobycia wystarczająco nowego boosta. Graph-tool wymaga co najmniej wersji 1.54, która prawie na pewno nie znajduje się w repozytorium w którym nie ma graph-toola.

2.1. Instalacja boosta

Jeśli boosta we w miarę nowej wersji nie ma w naszym repozytorium musimy ręcznie go pobrać i skompilować. Na szczęście nie jest to proces skomplikowany, aczkolwiek może trwać dość długo w zależności od parametrów maszyny, na której jest on kompilowany. Najprawdopodobniej wystarczy następujący ciąg poleceń:

```
$ wget http://sourceforge.net/projects/boost/files/boost/1.58.0/
  ↪ boost_1_58_0.zip/download -O boost.zip
$ unzip boost.zip
$ cd boost_1_58_0/
$ bash ./bootstrap.sh
$ sudo ./b2 install
```

2.2. Instalacja graph-toola

Pierwszym pomysłem jest wykonanie komendy:

```
$ sudo pip install graph-tool
```

Jeśli to zadziała to dobrze. Zazwyczaj jednak to nie wystarczy i musimy sami tę bibliotekę zbudować. *Być może* wystarczy następujące polecenia:

```
$ sudo aptitude install clang-3.4 libexpat1-dev python-scipy python-numpy  
  ↳ libcgal-dev libsparsehash-dev python-cairo python-matplotlib  
  ↳ libgraphviz-dev  
$ wget https://downloads.skewed.de/graph-tool/graph-tool-2.11.tar.bz2  
$ tar xvjf graph-tool-2.11.tar.bz2  
$ cd graph-tool-2.11/  
$ CXX='clang++' ./configure --with-sparsehash-prefix=/usr/include/google/  
$ make  
$ sudo make install
```

Niestety po drodze będziemy często pytani o liczne zależności (niektóre ujawnia się dopiero przy próbie uruchomienia WoSeDona). Należy cierpliwie doinstalowywać brakujące pakiety. Może pojawić się konieczność zainstalowania kompilatora C++14. W przypadku Ubuntu 12 jedyną opcją jest clang-3.4. Biblioteka sparsehash w Ubuntu jest instalowana w niestandardowej lokalizacji o czym musimy skrypt configure *explicite* poinformować.

2.3. Instalacja WoSeDona

Musimy jeszcze zainstalować libcorpus2. Instrukcja znajduje się w [1]. Sama instalacja WoSeDonu w porównaniu z instalacją graph-toola nie stanowi wyzwania:

```
$ git clone git@nlp.pwr.wroc.pl:wosedon  
$ cd wosedon/PLWNGraphBuilder  
$ sudo python setup.py install  
$ cd ../wosedon  
$ sudo python setup.py install
```

3. Uruchamianie

W rozdziale znajduje się przykład uruchamiania WoSeDona. Wraz z WoSeDonem dostarczone są grafy zbudowane dla Słowosieci, znajdują się w katalogu `wosedon/resources`. Jeżeli użytkownik chce zbudować graf z własnej wersji Słowosieci, musi użyć narzędzia PLWNGraphBuilder, który przyjmuje jako parametr ścieżkę do konfigu bazy danych.

Przykład uruchomienia PLWNGraphBuilder:

```
PLWNGraphBuilder -b baza_danych.ini -o Plwn_graph
```

Po zbudowaniu grafów synsetów i jednostek leksykalnych (w przypadku kiedy użytkownik chce wykorzystać swoje grafy zamiast dostarczonych z aplikacją) można przystąpić do ujednoznaczniania.

Przykłady uruchomienia narzędzia WoSeDon:

```
$ wosedon -c wosedon.ini -m ../model -V -f plik.xml -o wynik.xml
$ wosedon -c wosedon.ini -m ../model -VVV -f spis_plikow.txt -b
```

Spis wszystkich możliwych parametrów można poznać wydając polecenie `wosedon --help`. Tutaj nastąpi opis tylko najważniejszych z nich.

<code>-c wosedon.ini</code>	Ścieżka do pliku konfiguracyjnego (<i>v.</i> sekcja 5.).
<code>-m ../model</code>	Opcjonalne. Ścieżka do katalogu z modelem ¹ . Jeśli katalog jest pusty zostanie utworzony nowy model i zapisany w tym katalogu. Jeśli nie, duża część pliku konfiguracyjnego zostanie zignorowana a wczytany zostanie model z katalogu.
<code>-V</code>	Im więcej ,V' tym więcej WoSeDon będzie pisał. Domyslnie wypisuje tylko poważne błędy na stderr. Jedno ,V' spowoduje wypisywanie ostrzeżeń, również na stderr. Kolejne poziomy tego ustawienia wypisują zwykłe informacje na stdout.
<code>-f f.xml -o r.xml</code>	Ujednoznacznienie pliku „f.xml” zostanie zapisane jako „r.xml”.
<code>-f s.txt -b</code>	Alternatywnie wobec powyższego. Pliki wypisane linia po linii w pliku „s.txt” zostaną ujednoznacznione i zapisane w plikach „[nazwa pliku].wosedon.xml”.

4. Testy jednostkowe

W celu uruchomienia testów jednostkowych należy stojąc w katalogu `wosedon/wosedon` wydać polecenie:

```
$ python -m unittest discover tests/
```

5. Konfiguracja

Plik konfiguracyjny jest zwykłym plikiem tekstowym w formacie `.ini`. Opisuje on sposób budowy modelu oraz algorytm ujednoznaczniania. Jeśli opcja przyjmuje wiele argumentów podajemy je oddzielone odstępami. Jeśli wymagana jest lista par podajemy ją w postaci `klucz1:wartosc1 klucz2:wartosc2`.

¹Model to gotowy graf, przechowujący informacje o synsetach i relacjach między nimi, na którym będą puszczane algorytmy.

Poniżej przedstawimy opis wszystkich możliwych ustawień z podziałem na sekcje pliku. Przykłady konfiguracji zostaną podane w sekcji 6, a także można je znaleźć w repozytorium WoSeDonu w katalogu `wosedon/cfg`.

5.1. wosedon

Ta sekcja zawiera najbardziej podstawowe opcje, determinuje co będzie ustawiane w pozostałych sekcjach.

context WoSeDon poszukuje znaczeń słów na podstawie kontekstu, czyli słów sąsiednich. Ta opcja mówi które słowa mają zostać uznane za sąsiednie. Możliwe są dwie wartości:

- Document – cały dokument
- Sentence – tylko obecne zdanie

gbuilders Jakie grafy podać algorytmom. Wszystkie algorytmy wymagają by pierwszym był graf synsetów. Do niego można dołączyć (*v. mergers*, 5.1) dowolny inny, ale na razie tylko jeden. Na chwilę obecną znane są następujące typy grafów:

- SynsetGraphBuilder – podstawowy graf synsetów, musi zostać użyty. Wierzchołkami jego są synsety, krawędziami zaś relacje międzysynsetowe ze Słowosieci, o nazwach w stylu „s11” albo „s23”. Są to identyfikatory relacji z bazy poprzedzone literą „s”.
- LexicalUnitGraphBuilder – również graf na podstawie Słowosieci, jego wierzchołkami są jednostki leksykalne a krawędziami relacje między nimi, o nazwach w stylu „l11” albo „l23” – identyfikatory z bazy danych poprzedzone znakiem „l”.
- MSRGraphBuilder – graf zbudowany na podstawie plików **k-best**. Wierzchołki to jednostki leksykalne zaś krawędzie to „podobieństwo”. Jako że pliki te traktują każdą krawędź indywidualnie, jako nazwę posiadają krawędzie słowo puste.
- BidirectionalMSRGraphBuilder – to samo co **MSRGraphBuilder** ale bierze pod uwagę wyłącznie symetryczne relacje.
- WNDGraphBuilder – graf WordNet Domains. Zawiera wyłącznie krawędzie „isa”.
- SUMOGraphBuilder – graf opisujący relacje zawierania się znaczeń. Wierzchołkami są anglojęzyczne koncepty SUMO, relacją zaś może być: „instance”, „subclass”, „subrelation” albo „subAttribute”.
- CCLGraphBuilder – graf informacji wyciągniętych z korpusu „Słownik znaczeń”. Wierzchołkami są synsety. Relacje mówią jak blisko występowały razem: „in_sentence”, „in_paragraph” oraz „in_document”.

mergers Opcja wskazuje jak połączyć grafy zbudowane przy pomocy **gbuilders**. Obecnie sensownym jest podanie jednego tylko *merger* i to wyłącznie w przypadku gdy używane są dwa *gbuilders*.

Istnieją dwa podejścia do łączenia grafów. Można grafy *łączyć*, tzn. utworzyć nowe krawędzie między wierzchołkami obu grafów. Drugą możliwością jest *rzutować* jeden graf na drugi, tj. krawędziami z jednego grafu połączyć wierzchołki drugiego, po czym o całym pierwszym grafie zapomnieć.

WoSeDon korzysta z obu tych podejść, co zawyża liczbę *mergers*. Te rzutujące są istotnie wolniejsze od dołączających. Możliwe wartości to:

- SynsetsLUMerger – dołącza graf LU do grafu synsetów. Dodaje krawędzie „syn-lu”.
- SynsetsLUMerger2 – rzutuje graf LU na graf synsetów
- SynsetsSUMOMerger – dołącza graf SUMO do grafu synsetów. Wymaga odpowiedniego pliku z rzutowaniem. Dodaje krawędzie „syn-sumo”.
- SynsetsSUMOMerger2 – rzutuje graf SUMO na graf synsetów. Wymaga odpowiedniego pliku z rzutowaniem.
- SynsetsWNDMerger – dołącza graf WND do grafu synsetów. Wymaga odpowiedniego pliku z rzutowaniem. Dodaje krawędzie „syn-wnd”.
- SynsetsWNDMerger2 – rzutuje graf WND na graf synsetów. Wymaga odpowiedniego pliku z rzutowaniem.
- SynsetsMSRMerger – dołącza graf MSR do grafu synsetów. Dodaje krawędzie „syn-msr”.
- SynsetsCCLMerger – rzutuje graf CCL na graf synsetów

wsdalgorithm Wybór algorytmu do ujednoznaczniania. Większość z nich to różne warianty PageRanka (o czym poczytać można w [2]). Do wyboru są:

- GTPersonalizedPR
- GTPersonalizedPRNorm
- GTPersonalizedPRNorm2
- GTPersonalizedPRNormIt
- GTPersPRNormModV
- GTPersPRNormItModV
- GTPersPRNormItModVRankNorm
- GTPersonalizedPRNormReduction
- GTPersonalizedPRNormTwoStep²
- GTPersonalizedW2WPR

²Ważne jest aby w pliku konfiguracyjnym ustawić parametr `ini_nodes = sumo`.

- `GTPersonalizedW2WPRNorm`
- `GTStaticPR`
- `LeskAlg` – algorytm porównujący słowa z kontekstu ze słowami z definicji synsetu przy pomocy wybranej funkcji (v. 5.6). Wykorzystuje plik `gloss_file`.
- `PaintballWSD` – każdy węzeł otrzymuje początkowe pobudzenie i nie tracąc go przekazuje je dalej. Wykorzystuje tabelę impedancji. Więcej informacji można znaleźć w pracy [6].
- `GTSUDOKURun2`

rerankers Algorytmy tworzą dla każdego słowa z kontekstu ranking synsetów dla niego. Synset o najlepszej pozycji w rankingu zostanie wybrany jako znaczenie słowa. Zanim to jednak nastąpi można użyć rerankera, który przebuduje ranking po swojemu, być może poprawiając wyniki. Obecnie dostępne są następujące rerankery (można podać wiele na raz, poodzielane spacją):

- `LemmaRankingNormalizer` – normalizuje ranking dla każdego lematu do przedziału $\langle 0, 1 \rangle$
- `NodeDegreeRanker` – wartość każdego synsetu mnoży przez stopień jego węzła w grafie
- `LemmaRankingFirstSelector` – jeśli drugi wg rankingu synset jest mniej niż `percentage_diff` (v. 5.5) procent mniejszy od pierwszego, wybierany jest ten z nich, który posiada najmniejszy wariant.
- `LemmaRankingSelector` – jw., ale z pierwszym porównywane są wszystkie pozostałe które różnią się od niego o mniej niż `percentage_diff`
- `AContentReranker`

use_weights `True` lub `False`. Jeśli `False` to każda krawędź otrzyma wagę 1. (włącznie z krawędziami MSR, które dostały już indywidualne wagi. W przeciwnym wypadku użyta zostanie opcja „weights” by nadać krawędziom wagi, a każdy brak skutkuje stosownym ostrzeżeniem.

weights Lista par oddzielonych spacjami, każda para jest w postaci „a:b”. Pierwszy element pary oznacza nazwę krawędzi której dotyczy (np. „s23” albo „syn-lu”). Drugim elementem jest waga zapisana jako ułamek dziesiętny rozdzielony kropką. Każdy algorytm może swobodnie interpretować tę właściwość, jednak zazwyczaj preferują wagi mniejsze od 1, im wyższa waga tym ważniejsza krawędź. Każda krawędź nie uwzględniona w tym spisie otrzyma wagę 0.

5.2. wosedon:resources

W tej sekcji przede wszystkim znajdują się ścieżki do zasobów używanych przy budowie modeli albo w algorytmach. Większość zasobów powinna znajdować się w repozytorium WoSeDonu, w katalogu `wosedon/resources`, o którym więcej w sekcji 8.

Opcja	Opis
<code>sumo_graph_file</code>	ścieżka do pliku z grafem SUMO
<code>mapping_sumo_file</code>	ścieżka do pliku z mapowaniem Słownosieci na SUMO
<code>ccl_graph_file</code>	ścieżka do pliku z grafem CCL
<code>wnd_graph_file</code>	ścieżka do pliku z grafem WND
<code>mapping_wnd_file</code>	ścieżka do pliku z mapowaniem Słownosieci na WND
<code>msr_file</code>	ścieżka do pliku z grafem MSR
<code>plwn_graph_file</code>	ścieżka do pliku z grafem Słownosieci
<code>gloss_file</code>	ścieżka do pliku ze słownymi definicjami synsetów
<code>gloss_rel_file</code>	nieużywany
<code>impedance_table</code>	ścieżka do tabeli impedancji dla algorytmu Paintball
<code>tagset</code>	tagset korpusu który będziemy przetwarzać

5.3. wosedon:build_options

Są to wspólne ustawienia dla wszystkich budowniczych grafów, choć część wymienionych tu opcji jest rozpoznawana tylko przez budowniczych grafów ze Słownosieci, tzn. `SynsetGraphBuilder` oraz `LexicalUnitGraphBuilder`.

Uwaga: te wszystkie opcje służą do budowy modelu. Jeśli wczytywany jest gotowy model wszystkie zostaną zignorowane.

unique_edges Jeśli opcja ta przyjmuje wartość `True` z grafu zostaną usunięte powtarzające się krawędzie, tj. jeśli dwa wierzchołki są połączone wieloma krawędziami (nawet różnych typów) to pozostawiona zostanie tylko jedna. Nie ma możliwości ustalenia która.

directed_graphs Gdy zostanie ustawiona na `False` każda krawędź grafu będzie uważana za nieskierowaną.

syn_rel_ids Opcja używana tylko przez `SynsetGraphBuilder`. Lista numerów relacji międzysynsetowych (poddzielanych odstępami) które mają zostać *zachowane*. Wszystkie pozostałe relacje zostaną usunięte z grafu. W przypadku gdy lista jest pusta zachowane zostaną wszystkie krawędzie.

lu_rel_ids To samo co powyższe, ale dotyczy grafu jednostek leksykalnych i jest używane tylko przez `LexicalUnitGraphBuilder`.

accept_pos Dotyczy wyłącznie grafów Słownosieci. Podobnie jak powyższe opcje specyfikuje które części mowy (w postaci liczby) powinny zostać zachowane. Wierzchołki o niewypisanych częściach mowy zostaną usunięte. Wybierać można spośród następujących części:

Część mowy	Opis
------------	------

1	czasownik
2	rzeczownik
3	przysłówek
4	przymiotnik
5	angielski czasownik
6	angielski rzeczownik
7	angielski przysłówek
8	angielski przymiotnik

add_reversed_edges Opcja rozpoznawana tylko przez dwóch budowniczych grafów Słowosieci. Pozwala dodać podczas budowy przeciwne krawędzie dla pewnych relacji. Przyjmuje listę (rozdzieloną odstępami) par **a:b**, gdzie „a” to identyfikator już istniejącej relacji Słowosieci zaś „b” to identyfikator nowej relacji (koniecznie liczba całkowita), powstałej przez odwrócenie wszystkich krawędzi starej.

Tego identyfikatora można i należy używać w pozostałych opcjach, w szczególności w **syn_rel_ids**. Nowa relacja może otrzymać identyfikator już będący w użyciu. W szczególności używając tego samego identyfikatora co relacja pierwotna można wybraną relację uczynić nieskierowaną.

accept_lexicon Dopuszczalne wartości to: Słowosieć_2.2, Princeton_3.0 oraz AContent_1.0.

5.4. wosedon:merge_options

Będą to wspólne ustawienia dla wszystkich mergerów. Sekcja ta na razie jest pusta. Uwaga: te wszystkie opcje będą służyły do budowy modelu. Jeśli wczytywany będzie gotowy model wszystkie zostaną zignorowane.

5.5. wosedon:rerank_options

Ta sekcja zawiera ustawienia rerankerów. Na razie istnieje tylko jedna opcja: **percentage_diff**. Mówi ona o ile najwyżej procent dany synset może mieć niższy ranking od najlepszego, aby dalej rozważać jego kandydaturę.

5.6. wosedon:wsd_alg

Tu znaleźć można ustawienia wszystkich algorytmów. Oczywiście interpretacja poniższych wartości całkowicie zależy od wybranego algorytmu.

damping_factor Najczęściej jest to liczba przez którą przemnażane są wszystkie wartości gdy przechodzą po dowolnej krawędzi. Czasem synonimem zmieniania tej wartości jest zmiana wszystkich wag tyle samo krotnie. Najczęściej używa się wartości 0,85.

max_iter Po ilu iteracjach algorytm powinien się zatrzymać. Z nieznanym przyczyn liczba ta bywa później mnożona przez 2.

ini_nodes Lista typów węzłów do zainicjowania, rozdzielana odstępami. Możliwe wartości to: synset, sumo, wnd oraz msr.

lesk_function Funkcja której ma użyć algorytm Lesk. Możliwe opcje:

ExampleFunction Przykładowa funkcja, nie warto używać.

Intersection Liczy ile słów z definicji znajduje się w kontekście.

Cosine Cosinus wektorowy między wektorem kontekstu a wektorem wierzchołka. Wektor kontekstu to dla każdego słowa liczba jego wystąpień w kontekście.

Wektor wierzchołka powstaje podobnie. Dla wierzchołka tworzymy worek słów, tj. każdemu słowu występującemu w definicji przypisujemy liczbę 1. Następnie dodajemy do niego worki słów sąsiadów, przemnożone przez **damping_factor** oraz wagę krawędzi wiodącej do tego sąsiada. Worki sąsiadów powstały podobnie, chodzimy po grafie BFS-em na odległość **max_iter**.

Taki worek tworzy nam nowy wektor wierzchołka.

lesk_filter Filtr słów do leska. Lesk bierze pod uwagę tylko te słowa z definicji które zostaną przepuszczone przez filtr. Możliwe wartości:

- No – zabrania wszystkiego, prawdopodobnie bezużyteczny
- Yes – domyślna wartość, pozwala na wszystko
- Words – zabrania słów wypisanych w pliku
- POS – zabrania części mowy (w sensie tagsetu, nie liczb ze Słowosieci) wypisanych w pliku
- WordsPOS – zabrania słów o określonej części mowy wypisanych w pliku. Linia pliku powinna być w formacie „słowo; część mowy”. Oba człony mogą mieć wartość „*”, która pasuje do dowolnej wartości.

5.7. wosedon:lesk_filter

Ustawienia filtra dla Leska. Na razie istnieją tylko dwie opcje:

- list_file – ścieżka do pliku z lista rzeczy do odfiltrowania
- allow_only – odwraca filtr. Jeśli ustawione na **True**, tylko słowa znajdujące się w pliku będą dozwolone. W przeciwnym wypadku plik będzie zawierał definicje rzeczy zakazanych.

6. Przykłady

Oto kolejne przykłady najczęstszych zadań.

6.1. Najprostszy przypadek

Najprostsze i najbardziej standardowe ustawienia. Należy tu zwrócić szczególną uwagę na fakt, że nie odfiltrowujemy angielskich słów z bazy Słowosieci, co oznacza że WoSeDon ich użyje.

```
[wosedon]
context = Document      ; zdanie jest zbyt małym kontekstem
gbuilders = SynsetGraphBuilder ; musimy użyć grafu synsetów
wsdalgorithm = GTPersonalizedPRNorm2 ; najlepszy algorytm działający w
    ↪ sensownym czasie

[wosedon:resources]
plwn_graph_file = PLWN_06-07-2015/PLWN_graph ; plik z grafem. Trzeba
    ↪ zwrócić uwagę na brak końcówki: '_syn.xml.gz' zostanie dodane
    ↪ przez program.
tagset = nkjp ; już chyba wszystko używa tego tagsetu

[wosedon:build_options]
directed_graphs = False ; graf nieskierowany, dzięki temu mamy więcej
    ↪ połączeń w grafie i wyniki są lepsze

[wosedon:merge_options]

[wosedon:wsd_alg]
damping_factor = 0.85 ; standardowe ustawienie
max_iter = 15 ; standardowe ustawienie
ini_nodes = synset
```

6.2. Bez angielskiego, z sumo

Tak jak wyżej, ale tym razem dołączymy również graf SUMO, a także usuniemy angielską część grafu Słowosieci.

```
[wosedon]
context = Document
gbuilders = SynsetGraphBuilder SUMOGraphBuilder ; dodajemy graf SUMO
mergers = SynsetsSUMOMerger ; dołączamy go stosownym mergerem
wsdalgorithm = GTPersonalizedPRNorm2

[wosedon:resources]
sumo_graph_file = sumo_graph ; plik z grafem sumo
mapping_sumo_file = PLWN_06-07-2015/plwn2sumo_automap_rules_06-07-2015
    ↪ _resolved-rreduced-bulbuled-oknaked-corec-serdel-rubin.csv ;
    ↪ mapowanie grafu sumo na Słowosieć
plwn_graph_file = PLWN_06-07-2015/PLWN_graph
tagset = nkjp

[wosedon:build_options]
directed_graphs = False
```

```

accept_pos = 1 2 3 4 ; zostawiamy w grafie tylko polskie synsety, czyli
    ↪ te o polskich częściach mowy

[wosedon:merge_options]

[wosedon:wsd_alg]
damping_factor = 0.85
max_iter = 30 ; na wszelki wypadek
ini_nodes = synset ; nie inicjujemy wierzchołków sumo, potrafi to
    ↪ pogorszyć wyniki

```

6.3. Krawędzie i wagi

Niektóre relacje we Słowosieci z różnych przyczyn są tylko w jedną stronę, ale nie w każdym przypadku jest to dobrze. Jeśli używamy grafu skierowanego możemy chcieć dodać relacje przeciwne do niektórych. Poza tym przypiszemy krawędziom wagi.

W przykładzie tym należy zwrócić uwagę na fakt że opcje budowy dotyczą wewnętrznych spraw grafu, a zatem grafy Synsetów oraz LU używają na tym etapie swoich wewnętrznych identyfikatorów, czyli nazw relacji ze Słowosieci. Na końcu do wszystkich identyfikatorów (włącznie z tymi utworzonymi przez `add_reversed_edges`) dodają stosowny prefix.

```

[wosedon]
context = Document
gbbuilders = SynsetGraphBuilder
wsdalgorithm = GTPersonalizedPRNorm2

use_weights = true ; będziemy używać wag
weights = s11:0.7 s12:0.1 s30:0.5 s10000030:0.23 ; nadajemy wagi
    ↪ relacjom

[wosedon:resources]
plwn_graph_file = PLWN_06-07-2015/PLWN_graph
tagset = nkjp

[wosedon:build_options]
unique_edges = True ; tylko unikalne krawędzie
directed_graphs = True ; graf skierowany
add_reversed_edges = 30:10000030 ; dodajemy relacje przeciwną do
    ↪ relacji '30'
syn_rel_ids = 11 12 30 10000030 ; tylko te relacje nas interesują

[wosedon:merge_options]

[wosedon:wsd_alg]
damping_factor = 0.85
max_iter = 15
ini_nodes = synset

```

6.4. Lesk

Użyjemy algorytmu Lesk.

```
[wosedon]
context = Document
gbbuilders = SynsetGraphBuilder
wsdalgorithm = LeskAlg ; wybieramy Leska

[wosedon:resources]
plwn_graph_file = PLWN_06-07-2015/PLWN_graph
gloss_file = gloss_12-03-2015_Iobber_Defender_Npsemrel_Wsd_Sumo_Malt.xml
    ↪ ; potrzebny jest plik z definicjami dla synsetów
tagset = nkjp

[wosedon:build_options]
[wosedon:merge_options]

[wosedon:wsd_alg]
damping_factor = 0.85
max_iter = 3 ; nie potrzeba dużo
ini_nodes = synset
lesk_function = Cosine ; jedyna prawdziwa funkcja
lesk_filter = Words ; najprostszy filtr

[wosedon:lesk_filter]
list_file = poses ; plik z listą słów do filtrowania
allow_only = True ; tylko słowa na liście są brane pod uwagę
```

7. Inne narzędzia w repozytorium

7.1. CclExtractor

Jest to narzędzie do tworzenia grafu CCL na podstawie korpusu w tym formacie.

Instalacja:

```
$ cd CclExtractor
$ sudo python setup.py install
```

Opcje:

- `-i` – lista plików do przetworzenia
- `-o` – plik wynikowy
- `-d` – jeśli ustawione generowane będą krawędzie „in_document”, co jest powolne i raczej bezużyteczne
- `-p` – jeśli ustawione generowane będą krawędzie „in_paragraph”

- **-s** – jeśli ustawione generowane będą krawędzie „in_sentence”
- **-b** – jeśli ustawione, dla każdej pary wierzchołków utworzona zostanie co najwyżej jedna krawędź, ta najlepszego znalezionej typu

Przykłady:

```
$ CclExtractor -i a.ccl -o g.xml -d
$ CclExtractor -i a.ccl b.ccl c.ccl -o g.xml -b -p -s
```

7.2. PLWNGraphBuilder

Łączy się z bazą Słownosieci i tworzy na jej podstawie dwa grafy: synsetów oraz jednostek leksykalnych. Na chwilę obecną nie wydaje się działać.

7.3. Resolver

Skrypt przyjmuje jeden korpus ujednoznaczniiony na dwa różne sposoby i łączy wyniki ujednoznacznienia. W pierwszej kolejności patrzy na pierwszą metodę. Jeśli wg niej któreś słowo jest niepewne, tzn. znaczenie o najwyższym rankingu nie różni się wiele od znaczeń o niższych rankingach, wszyscy potencjalni kandydaci są oceniani na nowo wynikami drugiej metody.

Instalacja:

```
$ cd Resolver
$ sudo python setup.py install
```

Opcje:

- **-f, --file-list** – ścieżka do pliku zawierającego ścieżki względne (wobec obu folderów wejściowych) do plików mających być przetwarzane
- **-i, --input-dir** – pierwszy katalog wejściowy, z korpusem ujednoznacznionym główną metodą
- **-j, --input-dir2** – drugi katalog wejściowy, z korpusem ujednoznacznionym pomocniczą metodą
- **-o, --output-dir** – katalog w którym mają być składowane wyniki
- **-d, --diff** – różnica względem najlepszego wyniku dla kandydatów wybieranych względem pomocniczego rankingu, jako ułamek dziesiętny rozdzielany kropką

Przykłady:

```
$ Resolver -f lista.txt -i kpwr-GTPersonalizedPRNorm2/ -j kpwr-LeskAlg/ -
  ↪ o wyniki/ -d 0.3
```

7.4. average_path_length.py

Skrypt z katalogu `tools/`, służy do sprawdzania średniej długości ścieżki w grafie oraz jego spójności. Robi to poprzez serię prób przejścia między dwoma losowo wybranymi (z powtórzeniami) wierzchołkami. Próby trwają tak długo aż zostaną przerwane przez użytkownika.

Dotychczasowy wynik jest wypisywany na bieżąco: „trial” oznacza numer obecnej próby, „average lenght” średnią długość ścieżki, zaś „unconnected” procent niepołączonych wierzchołków.

Pierwszym parametrem jest ścieżka do pliku z grafem w formacie graph-toola, czyli np. model WoSeDonu. Drugi parametr jest opcjonalny, jeśli jest obecny potrafi zmienić skierowanie grafu.

Przykłady:

```
$ python average\_path\_length.py graph.xml
$ python average\_path\_length.py graph.xml directed
$ python average\_path\_length.py graph.xml undirected
```

7.5. corpus_statistic.py

Narzędzie do zbierania statystyk na temat korpusu.

7.6. make_xml_file_from_gloss.py

Generuje plik XML dla gloss z PLWN.

7.7. evaluation-kpwr.py, evaluation-sz.py

Skrypty (z katalogu `wosedon/wosedon`) do oceniania wyników działania WoSeDonu na korpusach odpowiednio „kpwr” oraz „składnica_znaczen”.

Przykład (dla składnicy wystarczy zmienić „kpwr” na „sz”):

```
evaluation-kpwr.py \
-i index_wsd.txt.wosedon \ # lista plikow wejsciowych
-r results/result.csv \    # trzy pliki wynikowe
-p results/precision.csv \
-rc results/recall.csv \
-d db_06-07-2015 \        # polaczenie z baza danych
-t nkjp                   # tagset
```

7.8. prec_general.py

Podsumowuje plik precyzji (ten którego ścieżka jest wartością parametru `-p` skryptów `evaluation-*`) i wypisuje (na `stdout`) ogólną informację o precyzji dla rzeczowników, czasowników oraz średnią z obu. Przyjmuje tylko jedną opcję, `-p`, czyli plik precyzji. Przykład:

```
prec_general.py -p results/precision.csv
```

8. Zawartość resources

Katalog ten zawiera zasoby do wykorzystania w pliku konfiguracyjnym WoSeDoNu. Znajduje się tam wiele plików plików, tu wypiszemy tylko skąd brać te najlepsze. Wszystkie ścieżki są względne wobec katalogu `wosedon/resources`.

- `sumo_graph_file = sumo_graph`
- `mapping_sumo_file = sumo_mapping-26.05.2015-Serdel.csv`
- `wnd_graph_file = ???`
- `mapping_wnd_file = ???`
- `msr_file = ???`
- `plwn_graph_file = PLWN_06-07-2015/PLWN_graph`
- `impedance_table = inhibit_chain_syn0.csv`
- `gloss_file`, `gloss_rel_file` – trzeba je rozpakować z archiwum `PLWN_12-03-2015/gloss_12-03-2015.zip`, używamy plików `gloss_12-03-2015_Iobber_Defender_Npsemrel_Wsd_Sumo_Malt`.
- połączenie z bazą danych – `PLWN_06-07-2015/db_06-07-2015`, przy czym trzeba uważać, gdyż „serwer” ma zmienne IP i może wystąpić konieczność ręcznej zmiany pliku

Literatura

- [1] O libcorpus2. <http://nlp.pwr.wroc.pl/redmine/projects/corpus2/wiki>.
- [2] Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. Random walks for knowledge-based word sense disambiguation. 2014.
- [3] Paweł Kędzia, Maciej Piasecki, Jan Kocoń, and Agnieszka Indyka-Piasecka. Distributionally extended network-based word sense disambiguation in semantic clustering of polish texts. *IERI Procedia*, 10(Complete):38–44, 2014.
- [4] Paweł Kędzia, Maciej Piasecki, and Marlena Orlińska. Word sense disambiguation based on large scale polish clarin heterogeneous lexical resources. pages 269–292, 2015.
- [5] Maciej Piasecki, Paweł Kędzia, and Marlena Orlińska. plwordnet inword sense disambiguation. 2016.
- [6] Maciej Piasecki, Radosław Ramocki, and Michał Kalinski. Information spreading in expanding wordnet hypernymy structure. In *Recent Advances in Natural Language Processing, RANLP 2013, 9-11 September, 2013, Hissar, Bulgaria*, pages 553–561, 2013.