ZESZYTY NAUKOWE POLITECHNIKI ŁÓDZKIEJ
Nr 1078 ELEKTRYKA, z. 121 2010

SZYMON GRABOWSKI, GRZEGORZ NOWAK
MARCIN RANISZEWSKI, CEZARY DRAUS
Wydział Elektrotechniki, Elektroniki, Informatyki i Automatyki
Politechniki Łódzkiej

PRZEGLĄD ZAGADNIEŃ ALGORYTMICZNO-INŻYNIERSKICH W APLIKACJI WSPOMAGAJĄCEJ TŁUMACZENIE PRZY TWORZENIU WIELOJĘZYCZNYCH WERSJI DOKUMENTÓW DTP

Recenzent: **doc. dr hab. Adam Jóźwik**

Maszynopis dostarczono 1. 10. 2010

Praca przedstawia szereg zagadnień związanych z automatycznym tłumaczeniem katalogów i broszur reklamowych przy użyciu systemu klasy CAT (Computer-Aided Translation) i dokumentuje nasze prace związane z otrzymaniem efektywnych rozwiązań algorytmicznych. Programy CAT zwykle działają na poziomie małych segmentów tekstu (fraz), zorganizowanych w postaci słowników (ang. Translation Memory). Programy CAT umożliwiają m.in. swobodną nawigację po dokumencie, automatyczne tłumaczenie rozpoznanych fraz i sugestie tłumaczenia dla fraz podobnych do już istniejących w systemie, wygodne wyszukiwanie i edycję słowników. Ogólnie biorąc rozważane przez nas zagadnienia można podzielić na: dotyczące interfejsu użytkownika oraz dotyczące algorytmów tekstowych. W szczególności rozwiązaliśmy zagadnienia detekcji symboli (tj. sekwencji znaków nie wymagających tłumaczenia dla większości par językowych takich jak liczby, jednostki fizyczne, kody, numery fabryczne i referencyjne, zastrzeżone znaki towarowe itp.), edycji słowników, etykietowania wybranych elementów dokumentu, tłumaczenia z dziurami (ang. gaps), pasowania rozmytego (ang. fuzzy matching). Funkcjonalności te przyśpieszają pracę tłumacza,

minimalizując szansę zaistnienia pewnych klas błędów w procesie tłumaczenia oraz ułatwiają zarządzanie dokumentem oraz bazą słowników. Tym samym, skrócony jest cykl produkcyjny dokumentu, co szczególnie jest ważne przy dokumentach DTP, które wymagają równoległego tłumaczenia na wiele języków (katalogi, broszury reklamowe).

1. WPROWADZENIE

W ostatnich latach można zaobserwować znaczący rozwój i upowszechnienie się aplikacji wspomagających pracę tłumaczy, określanych ogólnie mianem CAT (ang. *Computer-Aided Translation*, *Computer-Assisted Translation*) [1, 2]. Programy tego typu nie służą do automatycznego tłumaczenia (jak np. Yahoo! Babel Fish, Google Translate), lecz dostarczają narzędzi pozwalających na szybką i wygodną pracę człowieka tłumaczącego tekst z jednego języka na inny.

Programy CAT działają na poziomie małych fragmentów tekstu (zwroty, zdania, pojedyncze słowa), które przechowywane są w słownikach czy glosariuszach, często ogólnie określanych mianem *Translation Memory* (TM). Taka reprezentacja sprzyja ponownemu wykorzystaniu już znanego tłumaczenia, poprzez (być może kontekstową) podpowiedź, jak należy przełożyć frazę bieżącą. Często automatyczne tłumaczenie danej frazy nie jest dostępne, ale aplikacja potrafi znaleźć frazy podobne do bieżącej, których tłumaczenia mogą stać się wskazówką dla człowieka.

Do użyteczniejszych aplikacji CAT należą te, które wspierają nie tylko czysty tekst, ale i bardziej złożone formaty, np. prezentacje i dokumenty webowe, a nawet dokumenty DTP. W tym ostatnim przypadku przekład dokumentu wymaga idealnego oddania oryginalnej struktury i przestrzennego rozmieszczenia (tzw. layoutu) elementów tekstowych (i graficznych) na stronie. Elementy te są zwykle reprezentowane w postaci (często hierarchicznie zagnieżdżonych) ramek tekstowych.

Wg badań ankietowych przeprowadzonych w 2006 r. [3], w których wzięło udział 874 tłumaczy z 54 krajów, ponad 80% ankietowanych używa narzędzi CAT, ale w ograniczonym zakresie (tj. nie do wszystkich prac translatorskich). Wskazane przyczyny takiego stanu rzeczy to m.in.: brak wsparcia danego formatu, niewygodna obsługa (w szczególności zniechęcająca do użycia takiego narzędzia, gdy dokument do przetłumaczenia jest krótki), dostępność dokumentu źródłowego jedynie w postaci papierowej, niska skuteczność mechanizmu podpowiedzi. W świetle tych opinii wydaje się jasne, iż upowszechnienie się narzędzi CAT musi wynikać z powiększającego się repertuaru ich funkcjonalności (np. obsługi wielu formatów plików), bardziej przyjaznych

interfejsów użytkownika i ulepszeń algorytmicznych mechanizmów sugestii i automatycznych tłumaczeń. To ostatnie wiąże się ściśle z utrzymywaniem możliwie dużych, podzielonych tematycznie słowników.

Ważną kategorią dokumentów, które wymagają zwykle tłumaczenia z języka źródłowego na wiele języków docelowych, są katalogi i broszury reklamowe. Dokumenty te, nierzadko liczące kilkadziesiąt lub ponad sto stron, wymagają skoordynowanej w czasie pracy tłumaczy, gdyż efekt końcowy w postaci zlokalizowanych wersji dokumentu, powinien dotrzeć do odbiorców w wielu krajach mniej więcej w tym samym czasie. Oprócz więc wyzwań naszkicowanych powyżej, dochodzą wymagania logistyczne związane z zarządzaniem wielojęzycznym dokumentem.

Tłumaczenie dokumentów typu katalog reklamowy (czy ogólniej: dokumentów DTP) wymaga respektowania graficznego układu stron, użytych czcionek i innych atrybutów tekstu, zachowania układu tabel, rysunków, towarzyszących im opisów etc. W tekstach o charakterze technicznym (np. katalog produktów danej firmy) istnieje zwykle wiele symboli, skrótów i liczb, reprezentujących informacje takie jak np. nazwy, kody i numery fabryczne lub referencyjne poszczególnych modeli oraz jednostki fizyczne (opisujące np. wymiary, pobory mocy czy zakresy temperatur pracy danego urządzenia); „słowa" te nie powinny być tłumaczone. Wiele dokumentów DTP jest modyfikowanych cyklicznie (np. coroczne katalogi produktów danej firmy), lecz skala zmian między kolejnymi wersjami nie jest duża. Możliwe jest zatem, że 80–90% starej zawartości powinno być zwyczajnie skopiowane do wersji nowej. Choć wydaje się to zagadnieniem trywialnym, niekoniecznie musi takim być z uwagi na możliwą zmianę kolejności rozdziałów czy sekcji, dodawanie czy usuwanie poszczególnych produktów w obrębie danej kategorii etc. Wymienione aspekty sprawiają, iż tłumaczenie dokumentów DTP jest czymś jakościowo odmiennym od tłumaczenia ciągłego (linearnego) tekstu.

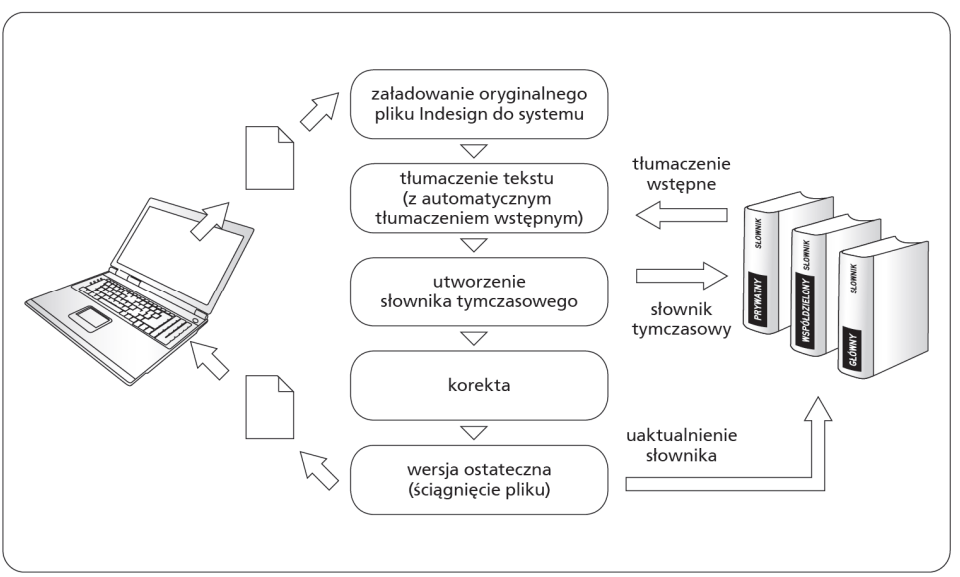
W praktyce, wiele firm i instytucji tworzących wielojęzyczne dokumenty DTP nie posiada dostępu do wygodnych narzędzi CAT, które byłyby odpowiednie do tego rodzaju dokumentów. W szczególności, istotne jest rozwiązanie kwestii poniższych:

- eksport / import fraz z / do dokumentu DTP;
- możliwość podglądu (przez tłumacza) tekstu źródłowego oraz tłumaczenia w postaci graficznej;
- gromadzenie i udostępnianie na potrzeby nowych projektów słowników (Translation Memory) zgromadzonych przy uprzednio zrealizowanych tłumaczeniach;
- utrzymywanie spójnej terminologii we wszystkich dokumentach tworzonych i tłumaczonych przez daną firmę / instytucję;
- możliwie rozbudowana automatyzacja procesu tłumaczenia.

2. GŁÓWNE FUNKCJONALNOŚCI I INFRASTRUKTURA OPRACOWYWANEGO SYSTEMU

2.1. Architektura funkcjonalna

Typowy proces tłumaczenia dokumentów DTP, zwany również procesem lokalizacji, składa się z następujących etapów: wyodrębnienie tekstu z jego graficznego układu, tłumaczenie tekstu, zastąpienie oryginalnego tekstu przez jego tłumaczenie, modyfikacja układu graficznego (długość tekstu różni się w poszczególnych językach), sprawdzenie i korekta. Jest to żmudny, ręczny i długotrwały proces. Naszym głównym celem jest jego skrócenie. Ponadto chcemy zaproponować wspólną internetową platformę dla wszystkich uczestników procesu lokalizacji, pozwalającą na wyeliminowanie wzajemnego przesyłania plików i ręcznego kopiowania tekstów, które zwykle stanowi potencjalne źródło wielu błędów. W tym celu zaproponowaliśmy funkcjonalną architekturę, spełniającą wymagania poszczególnych etapów procesu lokalizacji. Schemat tej architektury pokazany jest na rys. 1.



Rys. 1. Architektura funkcjonalna systemu

Najpierw dokument przesyłany jest na serwer, po czym wykonywane jest tłumaczenie dokumentu. Proces tłumaczenia wyodrębnionych z dokumentu fraz dokonywany jest w kontekście graficznym i przy wykorzystaniu automatycznego tłumaczenia wstępnego oraz mechanizmu sugestii, generowanych na podstawie wybranych słowników.

Następnie, frazy z dokumentu wraz z ich tłumaczeniami, są zapisywane w postaci tymczasowego słownika i są gotowe do wykorzystania w tłumaczeniu innych stron dokumentu. Po przetłumaczeniu, następuje proces korekty, a następnie generowana jest końcowa wersja dokumentu. Wraz z nią wykonywane jest uaktualnienie słownika, który w przyszłości może być użyty przy pracy z kolejnymi dokumentami.

Główne elementy systemu składające się na przestawioną architekturę funkcjonalną są następujące.

- interfejs webowy - ponieważ aplikacja powinna być dostępna dla klientów (firm) w dowolnym miejscu na świecie, w celu swobodnego tworzenia językowych wersji dokumentów korporacyjnych, zdecydowano się na model udostępniania aplikacji poprzez Internet. Intuicyjny interfejs zapewnia potrzebne funkcje: logowanie, przesyłanie dokumentów, tłumaczenie, generowanie przetłumaczonej wersji dokumentu, wybór słowników, zarządzanie dokumentami i słownikami;
- moduł tłumacza - pozwala na wykonanie tłumaczenia tekstów w ich oryginalnym kontekście graficznym. Umożliwia tłumaczenie przy wykorzystaniu jednego narzędzia, łączącego w sobie zalety aplikacji CAT (np. automatyczne odnajdywanie tłumaczenia znanych fraz, generowanie sugestii, propagacja tłumaczeń), z dodatkowymi funkcjami specyficznymi dla dokumentów DTP takich jak: automatyczne rozpoznawanie i uzupełnianie symboli nie podlegających tłumaczeniu;
- dedykowana baza danych - ponieważ większość zwrotów i wyrażeń jest używanych wielokrotnie w dokumentach korporacyjnych, ważne jest ich gromadzenie wraz z odpowiednimi tłumaczeniami. Taka baza danych znacząco wpływa na szybkość i jakość procesu tłumaczenia (80–90% wyrażeń może znajdować się w bazie i ich poprawne tłumaczenie może być wykonane automatycznie). Baza danych stanowi również źródło sugestii dla nowych fraz, które tłumaczone są po raz pierwszy. Wyrażenia z każdego nowego dokumentu mogą być dodane do istniejącej bazy danych jako nowy słownik i mogą być zastosowane w przyszłych tłumaczeniach. Po korekcie dokumentu odpowiadający mu słownik jest uaktualniany poprzez zastąpienie wszystkich zmodyfikowanych tłumaczeń ich wersjami końcowymi. Każdy słownik może być zapisany jako prywatny (korzystanie z niego zastrzeżone jest dla właściciela), współdzielony (zwykle dotyczy to specjalistycznych słowników współdzielonych z innymi użytkownikami) lub główny (rodzaj słownika globalnego). Konstrukcja bazy danych pozwala nie tylko na tradycyjne tłumaczenie dla par językowych, ale również na odnajdywanie relacji w ramach pojedynczego języka (np. synonimów). Używanie słowników znacząco zmniejsza nakład pracy potrzebny do

przetłumaczenia tekstów. Ma również dodatni wpływ na szybkość i jakość oraz spójność tłumaczenia;

- moduł wyszukiwania - ta funkcja jest bardzo użyteczna gdy wyrażenie nie zostaje znalezione w bazie danych jako tłumaczenie dokładne lub jako sugestia. Tradycyjne wyszukiwanie zastosowanego kiedyś tłumaczenia, w którym użyto konkretnego słowa, zwykle wymaga przeszukiwania wielu plików lub dokumentów drukowanych. Mechanizm wyszukiwania oparty jest na porównywaniu tekstów i pozwala na znalezienie dowolnego ciągu znaków w oryginale lub tłumaczeniu. Znalezione teksty (oryginał i tłumaczenie) wyświetlane są w pełnym kontekście z podświetlonym wyszukiwanym ciągiem znaków;
- moduł generowania przetłumaczonych stron lub całego dokumentu w oryginalnym układzie graficznym (ang. *layout*);
- moduł umożliwiający ściąganie (ang. *download*) dokumentu końcowego.

Kilka spośród wymienionych funkcji zasługuje na baczniejszą uwagę.

Pierwszą z nich jest rozpoznawanie słów nie podlegających tłumaczeniu. Wiele dokumentów korporacyjnych takich jak np. katalogi zawiera wiele wyrażeń, symboli, numerów referencyjnych, które nie powinny być tłumaczone. Automatyczne ich rozpoznawanie redukuje objętość i czas tłumaczenia.

Kolejną tego typu funkcją jest automatyczne tworzenie słowników w bazie danych. Spójne używanie terminologii przez pojedynczego tłumacza lub grupę tłumaczy jest jednym z wyzwań decydujących o jakości tłumaczenia. Ręczne tworzenie odpowiednich słowników pochłania wiele czasu. Nasz system analizuje dokumenty i automatycznie wyodrębnia wyrażenia i ich tłumaczenia. Wszystkie liczby, symbole i nieistotne językowo wyrażenia są eliminowane, a ważne dla tłumaczenia frazy są umieszczane w bazie danych. Utworzone w ten sposób słowniki mogą być łatwo edytowane i używane w przyszłych tłumaczeniach. Słowniki mogą być importowane lub eksportowane w postaci plików tekstowych (typu csv) lub w standardowym formacie wymiany słowników jakim jest TMX (*Translation Memory eXchange*).

Inna zaproponowana funkcja dotyczy zapewnienia jakości i spójności tłumaczenia. Dostarcza ona tłumaczowi narzędzia pozwalającego sprawdzić spójność używanej terminologii poprzez określenie czy specyficzne wyrażenia były przetłumaczone na jeden lub kilka sposobów. Użycie tej funkcji pozwala na większą kontrolę jakości tłumaczenia i pomaga wykryć błędy i możliwe nieścisłości.

Wszystkie funkcje są łatwe w użyciu. Nowoczesny interfejs webowy jest zaprojektowany jako intuicyjny, ergonomiczny i przyjazny użytkownikowi. Aby rozpocząć pracę z systemem nie jest konieczne żadne szkolenie.

2.2. Używane technologie

Działanie systemu opiera się na dwóch serwisach: WWW oraz DTP. Serwer WWW został zrealizowany w popularnej i szeroko rozpowszechnionej technologii LAMP: Linux jako system operacyjny, Apache jako serwer WWW, MySQL jako serwer bazy danych i PHP jako skryptowy język programowania dynamicznych stron webowych. Skalowalność i stabilność tworzonego w PHP oprogramowania uzyskano dzięki zastosowaniu architektury Zend Framework, obejmującej wzorzec projektowy Model–Widok–Kontroler (ang. MVC: *Model–View–Controller*) oraz system kontroli wersji SVN, pozwalający na koordynację pracy grupowej.

PHP jako popularny język programowania stron WWW pozwala na komunikację z bazą MySQL, która jest wykorzystywana do przechowywania danych dotyczących tłumaczonych dokumentów DTP oraz danych systemowych takich jak słowniki, dane użytkowników itp.

Najważniejszym wyzwaniem dla używanej bazy danych jest szybki dostęp do danych słownikowych, potrzebny zwłaszcza przy wyszukiwaniu on-line tłumaczeń i sugestii w zależności od wybranych słowników.

Język PHP pozwala na bezpośrednią komunikację z serwerem DTP. Przetwarzanie DTP jest realizowane w tle, w czasie rzeczywistym i jest przezroczyste dla użytkownika.

Serwerem DTP jest Adobe InDesign CS4 Server, który dla opracowania indywidualnych rozwiązań udostępnia środowisko programistyczne ze skryptowym językiem ExtendScript (rozszerzenie JavaScript opracowane przez Adobe).

Aby zapewnić niezależność i stabilność serwerów WWW i DTP, zastosowano technologię wirtualizacji, umieszczając obie wymienione części na dwóch niezależnych maszynach wirtualnych: Linux Fedora (serwer WWW: Apache, PHP, MySQL) oraz Windows Server 2008 (serwer DTP: InDesign Server CS4). Oba serwisy komunikują się przy użyciu protokołu http oraz systemu udostępniania plików Microsoft Windows (ang. *file sharing*).

2.3. Wymagania techniczne

Serwer DTP (InDesign Server CS4) wymaga instalacji na systemie operacyjnym Windows lub MAC OS. Serwer DTP jest odpowiedzialny za wszystkie przetwarzania DTP, w szczególności za ekstrakcję tekstów i generowanie podglądów stron oraz wersji końcowych tłumaczonych dokumentów. W prezentowanym systemie zastosowano 64-bitowy system operacyjny Windows Server 2008.

Minimalne wymagania dla InDesign Server CS4, pracującego na platformie Windows są następujące:

- procesor x86 lub x64, 2GHz lub szybszy;
- Microsoft® Windows Server® 2003, Service Pack 2 (wersja 32-bitowa lub 64-bitowa) albo Windows Server 2008 (wersja 32-bitowa lub 64-bitowa);
- .NET Framework 2.0;
- 2 GB RAM plus 256 MB dla każdej dodatkowej instancji InDesign Server;
- 1.8 GB dostępnego miejsca na twardym dysku; dodatkowa przestrzeń dyskowa wymagana jest podczas instalacji;
- wielkość pliku stronicowania w wirtualnej pamięci systemu Windows musi być zwiększona o 2 GB dla każdej instancji InDesign Server;
- rozdzielczość ekranu 1024 x 768.

Aby zapewnić niezawodność i elastyczność konfiguracji w fazie badawczo-rozwojowej, serwer WWW został zaimplementowany na 64-bitowym systemie operacyjnym Fedora, pozwalającym na łatwe i darmowe wykorzystanie potrzebnych technologii webowych.

Wymagania techniczne dla użytkownika końcowego (strona klienta) są raczej skromne:

- łącze internetowe o przepustowości 512 kb/s;
- przeglądarka internetowa kompatybilna z Internet Explorer 6+, Mozilla Firefox 2+;
- obsługa JavaScript.

3. OPIS ROZWIĄZAŃ WYBRANYCH FUNKCJONALNOŚCI SYSTEMU

Opracowywany przez nas system powinien spełniać następujące założenia:

- tłumacz, przypisany do danego projektu, może wybrać słownik(i) przydatne mu do pracy, będące rezerwuarem sugestii i możliwych automatycznych tłumaczeń;
- baza słowników zgromadzona w danej instytucji / korporacji powinna być zorganizowana w wygodny sposób, z podziałem językowym słowników (słowniki są dwujęzyczne: zestaw fraz w języku źródłowym i odpowiadających im tłumaczeń na język docelowy) oraz (jeśli to możliwe) tematycznym;
- powinny być dostępne wygodne mechanizmy wyszukiwania słów / fraz, z możliwością przeniesienia do strony dokumentu zawierającej dane dopasowanie, i w razie potrzeby, możliwość wniesienia poprawek do słownika;
- frazy, których tłumaczenia już istnieją (czy to w obrębie wcześniej przetłumaczonej części dokumentu, czy w słownikach istniejących

wcześniej) powinny być tłumaczone automatycznie, jednakże z możliwością odrzucenia takiego tłumaczenia / ręcznej edycji;
- w przypadku konfliktu dostępnych tłumaczeń (tj. danej frazie wejściowej odpowiada kilka różnych tłumaczeń w zgromadzonych słownikach), należy preferować sugestie: z bieżącego dokumentu, ze słowników z danej tematyki, z tych stron dokumentu referencyjnego (np. ubiegłorocznej wersji danego katalogu), które są najbardziej podobne do strony aktualnie tłumaczonej;
- frazy formalnie biorąc nowe, ale różniące się od istniejących tylko jednym lub większą liczbą słów nietłumaczonych (np. liczby, jednostki fizyczne), powinny być tłumaczone z automatyczną podmianą danych symboli (tłumaczenie „z dziurami");
- dla innych fraz powinny być na życzenie pokazywane frazy podobne (złożone w części z tych samych słów, być może z uwzględnieniem „zgrubnego" stemmingu), wraz z ich tłumaczeniami;
- do słów, których nie należy tłumaczyć, należy zaliczyć również znaki towarowe (ang. *trademarks*), sygnalizowane symbolem ™ lub ® na końcu takiej sekwencji; problem polega na automatycznej detekcji początku znaku towarowego;
- dostępność narzędzia typu lupa, powiększającego wskazany fragment strony dokumentu w postaci graficznej;
- dostępność narzędzia anotacji dokumentu, tj. nakładania na dokument w postaci graficznej dodatkowej warstwy, w której można przechowywać notki i proste elementy graficzne, np. strzałki. Mechanizm ten powinien ułatwić komunikację na linii tłumacz – korektor (lub menedżer projektu);
- dla tłumaczonych stron możliwość skorzystania z miary podobieństwa stron, tak by z większą precyzją wyszukiwać elementy dokumentu wcześniej już przetłumaczone.

W poniższych podrozdziałach opisujemy, w jaki sposób rozwiązaliśmy niektóre z wyliczonych wyżej problemów, w szczególności interesujących z algorytmicznego punktu widzenia.

3.1. Edycja słowników

Ponieważ jakość słowników i ich dopasowanie tematyczne do konkretnego dokumentu ma bardzo duże znaczenie dla tłumaczenia, dlatego staraliśmy się wyposażyć użytkowników naszego systemu w narzędzia pozwalające na tworzenie i utrzymanie słowników wysokiej jakości. Oprócz możliwości automatycznego tworzenia słownika z bieżącego dokumentu oraz jego aktualizacji po dokonaniu końcowej korekty, w systemie zaproponowano również funkcję ręcznej edycji słowników wraz z odpowiednimi procedurami wyszukiwania, zastępowania oraz czyszczenia słowników. Zaimplementowano

kilka sposobów wyszukiwania tekstów w wybranych słownikach: bez rozróżniania wielkości liter; z rozróżnianiem wielkości liter; wyszukiwanie oddzielnych wyrazów oraz wyszukiwanie oddzielnych słów z rozróżnianiem wielkości liter. Działanie tych procedur oparte jest na wykorzystaniu w MySQL klauzuli *like*, lub mechanizmu *match-againts* (w trybie boolean), co zapewnia odpowiednią do pracy on-line szybkość wyszukiwania. Przy zastosowaniu *match-against* liczba białych znaków pomiędzy wyrazami jest ignorowana, a słowa wpisane na listę „stop words" oraz krótsze niż czteroznakowe nie zostaną znalezione. Wyszukiwanie z rozróżnianiem wielkości liter jest natomiast oparte na wolniejszym operatorze *regexp*. Wyszukiwanie tekstu (w tekście źródłowym i w tłumaczeniu) odbywa się we wszystkich słownikach, wskazanych przez użytkownika lub określonych za pomocą wyboru konkretnej pary językowej.

Funkcja wyszukiwania może zostać wykorzystana do czyszczenia słowników poprzez znajdowanie i usuwanie błędnych tłumaczeń. Może również być użyta wraz z funkcją zastępowania: „znajdź i zamień" lub „zamień wszystkie wystąpienia", które działają podobnie jak ma to miejsce w większości edytorów tekstu.

Inny aspekt czyszczenia słowników polega na identyfikacji występujących w nim duplikatów i usuwaniu ich po akceptacji użytkownika. Duplikatem danego wiersza w słowniku są wiersze, w których kolumny zawierające tekst źródłowy i tłumaczenie składają się z tych samych wyrazów, występujących w tej samej kolejności co odpowiadające im kolumny wiersza wzorcowego. Duplikaty mogą ewentualnie zawierać białe znaki (spacje, tabulatory itp.), liczby lub symbole. Wielkość liter jest rozróżniana.

Przykładowo, dla frazy:

TEKST ŹRÓDŁOWY	**TŁUMACZENIE**
mała Ala ma kota	little Alice has a cat

następujące wiersze zostałyby wykryte jako duplikaty:

TXT_LAN1	**TXT_LAN2**
mała Ala ma kota	little Alice Has a cat
mała Ala ma kota	little Alice Has a cat
Mała Ala ma kota	Little ALICE has a cat
123 mała Ala ma kota123!@#	#$456 little Alice Has a cat 123

a następujące nie:

TXT_LAN1	**TXT_LAN2**
mała Ala ma czarnego kota	little Alice has a black cat
Bardzo mała Ala ma kota i psa	Very Little Alice has a cat and a dog
Ala ma kota	Alice has a cat

Wyszukiwanie duplikatów przebiega dwuetapowo. Wstępni kandydaci są znajdowani przez *match-against* w MySQL, a następnie w skrypcie PHP następuje ich tokenizacja i szczegółowe porównanie.

3.2. Sugestie dla fraz krótkich

Im większa zgromadzona baza słownikowa, tym większa i potencjalnie trafniejsza oferta sugestii przedkładanych tłumaczowi. Pewnym problemem są jednak frazy krótkie (np. skróty pewnych słów, które jednak wymagają tłumaczenia), gdzie potrzebna jest miara podobieństwa (z zerem oznaczającym identyczność fraz), której duże wartości będą wskazaniem do odfiltrowania mylnych sugestii. Nieraz dla tłumacza wystarczającą podpowiedzią jest znalezienie krótkiej frazy (np. pojedynczego skrótu) w obrębie frazy dłuższej. Również tego typu podobieństwa chcemy znajdować.

Zaproponowany przez nas algorytm oblicza najpierw kod Soundex [4] danej frazy, korzystając ze standardowej funkcji MySQL. Klasyczny algorytm Soundex, opracowany jeszcze przed rokiem 1920 do indeksowania nazwisk i imion, zwraca pierwszą literę napisu (najczęściej na wejściu jest pojedyncze słowo) oraz trzy cyfry, bazujące na kolejnych spółgłoskach napisu (jeśli spółgłosek jest więcej, to sufiks jest ignorowany). Dla przykładu, spółgłoski *f* i *v* mają tę samą przypisaną cyfrę i dzięki temu np. nazwiska *Smirnov* i *Smirnoff*, jako spokrewnione, mogą być w pewnym sensie utożsamione (otrzymują identyczne kody Soundeksu). Mówi się, że Soundex jest historycznie pierwszym algorytmem fonetycznym, tj. takim, który wynajduje homofony: słowa o różnej pisowni, lecz identycznej wymowie. Zwróćmy jednak uwagę, że Soundex zaprojektowany został pod kątem języka angielskiego, a opracowanie jego wariantów dla innych języków (dla każdego osobno) wymaga żmudnej pracy.

Wracając do naszego algorytmu: w pierwszej fazie filtracji pozyskujemy te elementy słownika $E(s)$, dla których kod Soundeksu jest identyczny z kodem Soundeksu frazy bieżącej s. Jeśli fraza s należy do $E(s)$, to oczywiście zwracana jest jako pierwsza sugestia i usuwana z $E(s)$. W dalszej analizie używamy innego znanego algorytmu fonetycznego, Metaphone [5]. Jeśli jego kod dla s ma długość 1 ($|mtph(s)| = 1$), to dodatkowo nakładany jest warunek $Lev(s, |e_i|) < |s|$, gdzie $Lev(\cdot)$ jest miarą Levenshteina, na frazy e_i dołączane do listy sugestii. Jeśli jednak zachodzi $|mtph(s)| > 1$, to do listy sugestii dodajemy te elementy e_i, dla których $mtph(e_i) \subset mtph(s)$. Dodawane do listy elementy są usuwane z $E(s)$.

Przyjęliśmy, że lista sugestii może liczyć 7 pozycji (dłuższa lista raczej rozpraszałaby tłumacza). Jeśli w wyniku dotąd opisanej procedury otrzymana lista liczy mniej niż 7 pozycji, to uzupełniamy ją frazą z $E(s)$ mającą najdłuższy afiks (tj. prefiks lub sufiks – decyduje długość) wspólny z s; w przypadku remisów wszystkie „zwycięskie” frazy są dodawane do listy

sugestii, przerywając jednak ich dodawanie, gdy lista ta osiągnie założoną maksymalną długość.

Jako ostatnia, użyta jest miara m podobna do długości najdłuższego wspólnego afiksu. Konkretnie, miara $m(s_1, s_2)$ może być zdefiniowana jako $max(m_1(s_1, s_2), m_2(s_1, s_2))$, gdzie $m_1(s_1, s_2) = |s_2|$, jeśli s_2 jest w pełni zawarte w s_1, oraz 0 w przeciwnym przypadku, zaś $m_2(s_1, s_2)$ jest długością najdłuższego wspólnego afiksu s_1 i s_2. Obliczamy $m(s, e_i)$ dla wszystkich (pozostałych) elementów e_i z $E(s)$ i wybieramy element, dla którego $m(s, e_i)$ jest największe; i tu mogą być remisy, których obsługa jest analogiczna do przypadku opisanego wcześniej. Wszystkie operacje na łańcuchach tekstowych w opisanej procedurze znajdowania sugestii wykonywane są po konwersji tych napisów na małe litery.

Tabela 1. Sugestie dla słów krótkich, zestawienie wyników

SŁOWO	LICZBA SUGESTII SOUNDEKSU	PIERWSZA SUGESTIA (WG SOUNDEKSU)	LICZBA SUGESTII WG NASZEGO ALGORYTMU	PIERWSZA SUGESTIA (WG NASZEGO ALGORYTMU)
80mm	425	140 Min	7	80mm
Axial	2	Axial	2	Axial
8 pin	9	PAMM	3	8 pin
kA	6	Key	2	kA
n/a	6	N/A	2	N/A
0,6 s	31	SCC	4	0,6 s
6yr	3	3yr	2	6yr
any	4	Any	3	Any
b.16	14	B.	7	B.16
Bells	11	black	1	Bells
Cap	11	C23WCI 3PF	0	–
Coil	3	COIL	2	COIL
clear	5	Colour	4	Clear
CSWFF	11	C23WCI 3PF	1	CSWFF
F3ACK	22	F3ACK	1	F3ACK
fixed	109	F = Fixed	7	fixed
hooks	4	H x W	1	HOOKS
nuts	3	Notes	3	Nuts
PAMM	9	PAMM	5	PAMM
pole	57	3 pole	7	Pole

		358		
quad	3	Qty	1	quad
red	7	Red	4	Red
SI:	23	SCC	2	SI:
Sun	3	100 s. On	1	Sun
Watt	12	Watt	3	Watt

Algorytm został zaimplementowany w PHP 5.2.4 i używa standardowych funkcji i struktur tego języka. Pierwszy etap filtracji wykonywany jest głównie na poziomie bazy danych (przy użyciu funkcji MySQL o nazwie *soundex*()). Następnie, wykorzystywane są funkcje PHP: *metaphone*() oraz *levenshtein*(). W dalszym etapie korzystamy z własnej funkcji, nazwanej *affixLen*(), dostosowanej do napisów Unicode.

Uruchomiliśmy nasz algorytm na przykładowej dużej tabeli zawierającej dane o urządzeniach elektrycznych. Tabela 1 zawiera pewne charakterystyki otrzymanych list sugestii dla wybranych fraz wejściowych (słów o długości nie przekraczającej 5 znaków). W drugiej kolumnie widzimy liczebności fraz w tabeli z takim samym kodem Soundex jak kod frazy wejściowej. Trzecia kolumna pokazuje pierwszą (tj. dość przypadkową spośród zwróconych) sugestię Soundeksu – chcemy tym samym pokazać, jak odmienna może być ta sugestia od frazy wejściowej, a zatem jak nieadekwatny dla naszego celu jest mechanizm Soundex, użyty samodzielnie. Ostatnie dwie kolumny zawierają liczbę sugestii wyjściowych otrzymanych przy użyciu naszego algorytmu (maksymalnie 7, ale często dużo mniej, w skrajnych przypadkach brak sugestii) oraz pierwszą na liście, tj. najlepszą sugestię.

Tabela 2. Listy sugestii

ŹRÓDŁO (ANGIELSKI)	SUGESTIE (ANGIELSKIE I POLSKIE TŁUMACZENIA)
80mm	**80mm**, 80mm **mm**, mm **35mm**, 35mm **60mm**, 60mm **110mm**, 110mm **160mm**, 160mm **210mm**, 210mm
[cd]	**Code**, Nr kat.
kA	**kA**, kA

	: 100 kA Icu = Ics = Icw, : 100 kA Icu = Ics = Icw
n/a	**N/A**, N/A **no**, nie
0,6 s	**0,6 s**, 0,6s **s**, s **0.06 s**, 0.06 s **0.065 s**, 0.065s
6yr	**6yr**, 6 lat **3yr**, 3 lata
Cap	–
Coil	**COIL**, CSWL **Coil**, Cewka
Red	**Red**, Czerwona **Red**, Czerwone **red**, czerwony **RED:**, CZERWONY
SI	**SI:**, SI: **s**, s

Tabela 2 zawiera pełne listy sugestii otrzymane przy użyciu naszego algorytmu. Zachowana jest kolejność, w jakiej algorytm je zwraca. Językiem źródłowym był angielski, zaś docelowym polski. Każdej sugestii (w prawej kolumnie, pogrubioną czcionką) towarzyszy jej polskie tłumaczenie (czcionka domyślna).

Wyniki pozwalają na wyciągnięcie następujących wniosków. Jeśli istnieje dopasowanie dokładne, to jest ono zwracane pierwsze. Długość list sugestii wynika z liczby fraz podobnych do frazy bieżącej. Wielkość liter w zasadzie nie ma znaczenia przy znajdowaniu sugestii, choć jeśli zostają znalezione sugestie różniące się tylko wielkością liter, to wyżej na liście zwracana jest wersja zgodna z frazą źródłową (np. *Red* w tabeli 2 zostało zwrócone wcześniej niż *red*, gdy *Red* było słowem wejściowym). Obecna implementacja jeszcze nie usuwa duplikatów z list sugestii, co powinno znacząco skrócić niektóre listy, bez uszczerbku na ich jakości.

Zwracane wyniki nie zawsze są bliskie ideału (por. *[cd]* → *Code* w tabeli 2). Nietrudno zauważyć, że doskonałości w tym względzie nie osiągniemy rozpoczynając filtrację od algorytmu Soundex. Dla przykładu, *cap* zwróciło zero sugestii (por. tabela 2), mimo iż istniało słowo *Caps* w słowniku. Przyczyna tej sytuacji jest prosta: kody Soundex dla tych dwóch słów są różne (odpowiednio C100 i C120). W przyszłości zamierzamy zatem opracować bardziej adekwatną dla problemu filtrację wstępną fraz, uruchamiając ją i przechowując zwrócone

wyniki na serwerze bazy danych. Etap ten może być kosztowny (i powinien być periodycznie powtarzany, gdyż słowniki mają oczywiście naturę dynamiczną), ale procedura ta powinna się zamortyzować przy wielu oczekiwanych operacjach wyszukiwania.

3.3. Rozpoznawanie nietłumaczonych symboli

Niektóre słowa lub frazy w tekście nie powinny podlegać tłumaczeniu; dotyczy to liczb (wyrażonych cyframi), jednostek fizycznych (np. *mm*, *kg*, *V*), kodów i numerów fabrycznych urządzeń, a także zastrzeżonych znaków towarowych, których detekcja będzie opisana w sekcji 3.4. W odniesieniu do – chwilowo – pojedynczych słów tego typu będziemy używali terminu „symbol".

Wykrycie tak zdefiniowanych symboli ma dwa zastosowania: (*i*) wskazuje pola, które mają być zwyczajnie skopiowane z języka źródłowego, oraz (*ii*) pozwala na tłumaczenie z dziurami (ang. *gaps*), jeśli fraza z usuniętym symbolem została wcześniej zapamiętana w TM. Jako przykład (z rzeczywistego słownika) podajmy frazę w języku źródłowym *Shunt trip 1*; jeśli słowo *1* zostanie rozpoznane jako symbol, w takim wypadku słownik (TM) zostanie przeszukany pod kątem fraz *Shunt trip* i *Shunt trip [symbol]*, i jeśli wcześniej do słownika została dodana fraza *Shunt trip 2* (gdzie oczywiście słowo *2* również zostało rozpoznane jako symbol), to tłumaczenie dla tej frazy zostanie użyte w miejscu bieżącym, z tą różnicą, że symbol *2* zostanie zastąpiony symbolem *1*.

W prostym rozwiązaniu można założyć (jak powyżej), że symbolami są pojedyncze słowa [6], gdzie przez pojęcie słowa rozumiemy maksymalną sekwencję znaków ograniczoną białymi znakami (lub początkiem / końcem całej frazy). Słowo jest uważane za symbol na podstawie statystyki występujących w nim znaków: dominacji cyfr i / lub wielkich liter oraz obecności (we frakcji większej niż progowa) nietypowych dla danego języka digramów (dwuznaków), takich jak np. *bz* dla języka polskiego. Statystyki digramów zostały zgromadzone na podstawie pewnych książek w czystym formacie tekstowym (ang. *plain text*) dostępnych w serwisie Project Gutenberg. Szczegóły tego rozwiązania zostały opisane w [6].

Pewnym ulepszeniem [7] tego rozwiązania była analiza słów złożonych z segmentów. Przykładem jest słowo *16-way*, gdzie analiza oparta na częstościach digramów i klas znaków (takich jak, w tym przykładzie, cyfry) nie sprawdza się. Lepszym rozwiązaniem jest rozbicie słowa na znaku łącznika i analiza obu powstałych segmentów osobno. Dodatkowo uwzględniamy skróty charakterystyczne dla danego języka, takie jak *wrt* (*with respect to*), *i.e.* (łacińskie „id est" = „to jest") czy *w/o* (*without*) w języku angielskim; *np.* (*na przykład*), *tj.* (*to jest*) w języku polskim; *SVP* ("S'il vous plaît" = „proszę") w języku francuskim. W tych przypadkach sama analiza digramów mogłaby zawieść, gdyby tekst będący źródłem statystyk nie zawierał wielu albo nawet

żadnego wystąpienia zwrotów tego typu (sytuacja dość typowa przy skrótach „internetowych", takich jak *btw* (*by the way*) czy *fyi* (*for your information*) w języku angielskim, które naturalnie w utworach np. Dickensa nie występują). Zauważyć należy, iż lista tego typu popularnych skrótów nie będzie liczyć więcej niż kilkadziesiąt (może kilkaset dla języka angielskiego) słów, a zatem nie tylko nie zajmuje dużo miejsca w pamięci RAM, ale też z łatwością mieści się w pamięci cache procesora, co jest korzystne dla szybkości operacji odwołujących się do tej listy.

Inne uwzględnione przez nas działania przy analizie potencjalnych skrótów (które nie zostały znalezione na wyżej opisanej liście) to usuwanie par znaków: [], (), {}, <>, '', „" wokół badanego słowa (jeśli taka sytuacja ma miejsce), a następnie rozbijanie pozostałej sekwencji na znakach z klasy: [-/:;] (znów, jeśli takie znaki występują w sekwencji). Powstałe segmenty analizowane są osobno, algorytmem z pracy [6]. Jeśli wszystkie segmenty są uznane za symbole, to całość jest uznawana za symbol, w innym przypadku całość danej sekwencji podlega tłumaczeniu. Rozwiązanie to jest bezpieczne, w tym sensie iż preferuje raczej fałszywe negatywy (zaetykietowanie symbolu jako nie-symbol) niż fałszywe pozytywy (uznanie za symbol sekwencji, która symbolem nie jest). Lepiej jest bowiem zostawić nieco więcej pracy tłumaczowi niż wprowadzić „z automatu" błąd, który może zostać przeoczony.

Rozważmy teraz przykład hipotetycznego słowa *1073741824-way*. Będzie ono rozbite na człony *1073741824* i *way*, a ponieważ statystyka digramów (przy wskazanym języku angielskim) dla członu *way* nakaże oznaczenie tego członu jako nie-symbol, to cała sekwencja wejściowa zostanie tak oznaczona i pozostawiona do przetłumaczenia człowiekowi. Rozwiązanie takie prawdopodobnie nie miałoby miejsca w przypadku „globalnej" analizy sekwencji, z uwagi na dominację cyfr (które sugerują symbol).

3.4. Rozpoznawanie znaków towarowych

Znaki towarowe to prawnie chronione sekwencje słów zakończone symbolem ™ lub ®. Firmy używają ich, by jednoznacznie określić swój produkt lub usługę wśród konsumentów.

Znaki towarowe powinny zostać wyodrębnione z tekstu, tak by nie podlegały tłumaczeniu. O ile posiadamy jednoznacznie określony warunek końca znaku towarowego (symbol ™ lub ®), o tyle problematyczne jest znalezienie jego początku. W sekwencji słów: „słowo1 słowo2 słowo3 ™" nie wiemy czy znakiem towarowym jest „słowo3", „słowo2 słowo3", czy też „słowo1 słowo2 słowo3".

Poniżej opisany algorytm analizuje frazy (sekwencje słów) w języku angielskim i zwraca listę unikalnych znaków towarowych (przekształconych do małych liter) w nich zawartych. Oparty on jest w głównej mierze na wydzieleniu

części wspólnej z fraz zakończonych tym samym znakiem towarowym [7]. Algorytm można opisać w następujących krokach:

1. Wydziel z listy fraz podfrazy zakończone symbolem ™ lub ®. Usuń (o ile występują) spacje przed symbolem ™ lub ®. Wydzielone podfrazy muszą zawierać dokładnie jeden z symboli ™ lub ® na swoim końcu oraz być maksymalne pod względem długości (wyrażonej w słowach).
2. Odwróć kolejność słów w podfrazach (separatorem jest ciąg białych znaków). Słowa w ramach podfrazy łącz ze sobą jednym znakiem spacji.
3. Dla każdej w ten sposób uzyskanej podfrazy, przeanalizuj jej słowa:
    a. jeśli słowem jest „the" (wielkość liter nie jest istotna) lub słowo zawiera przecinek, usuń to słowo i wszystkie słowa po nim następujące;
    b. jeśli słowo zawiera nawias otwierający lub zaczyna się znakiem cudzysłowu, usuń wszystkie słowa po nim następujące;
    c. usuń nawiasy i znaki cudzysłowu ze słów;
    d. skróć każdą podfrazę, tak by składała się z maksymalnie trzech słów;
    e. w przypadku co najmniej dwóch słów, sprawdź czy pierwsze słowo nie rozpoczyna się dużą literą lub cyfrą. Jeśli tak jest, to następne słowo w tym ciągu, które nie rozpoczyna się dużą literą lub cyfrą jest usuwane wraz ze słowami następującymi po nim.
4. Odwróć kolejność słów w podfrazach.
5. Posortuj tablicę podfraz wg ich liczby słów (od podfraz złożonych z jednego słowa do podfraz złożonych z trzech słów).
6. Zbuduj zbiór znaków towarowych w ten sposób, że do zbioru dołączana jest podfraza jeśli nie jest ona lub nie kończy się istniejącym już w zbiorze znakiem towarowym.
7. Przekształć zgromadzone znaki towarowe na pisane małymi literami (usuń duplikaty).
8. Zwróć listę tak otrzymanych znaków towarowych.

Krok trzeci powyższego algorytmu jest wynikiem następujących założeń:

1. Znak towarowy w języku angielskim nie powinien zawierać słowa „the" ani przecinka (lista zabronionych słów i znaków może zostać rozszerzona lub zmieniona dla innych języków).
2. Znak towarowy, jeśli jest pisany z wielkich liter, to w całości.
3. Znak towarowy, jeśli zawiera słowa rozpoczynające się od cyfr, to pisany jest z wielkich liter.
4. Maksymalna długość znaku towarowego w języku angielskim wynosi 3 (tę wielkość można traktować jako parametr algorytmu i odpowiednio zmieniać dla innych języków).

3.5. Tłumaczenie z dziurami

Tłumaczenie z dziurami to automatyczne tłumaczenie tych fraz, które spotykane są po raz pierwszy, ale w słowniku istnieją frazy różniące się od frazy bieżącej tylko jednym lub większą liczbą symboli. Tym samym, fraza bieżąca *word1 word2 symbolA word3 symbolB word4* oraz fraza ze słownika *word1 word2 symbolX word3 symbolY word4* są w pewnym sensie identyczne, gdy przekształcimy je do postaci: *word1 word2 [gap1] word3 [gap2] word4*. Jeśli znajdziemy taki wzorzec (wraz z istniejącym w słowniku tłumaczeniem), dalsze postępowanie jest trywialne (wystarczy zamienić *symbolX* na *symbolA* oraz *symbolY* na *symbolB*). Jest mało prawdopodobne, aby poprawna kolejność symboli „na wyjściu" była inna niż przy parze referencyjnej, tj. ze słownika. Również mało prawdopodobne (ale możliwe), aby całość zaproponowanego tłumaczenia była błędna. Potrzebna jest jednak możliwość wyłączenia mechanizmu tłumaczenia z dziurami, gdyby sytuacje opisane miały stosunkowo często miejsce.

Drugą, ciekawszą kwestią, jaka wiąże się z tą funkcjonalnością, jest sposób indeksowania fraz umożliwiający szybkie znajdowanie par fraz podobnych w opisanym sensie. Rozwiązanie nasze wykorzystuje tablicę asocjacyjną, w której kluczem jest para kolejnych słów frazy (zakładamy, że fraza wejściowa składa się z przynajmniej dwóch słów, pomijając ewentualne symbole), a wartością słownik (czyli również tablica asocjacyjna) wszystkich fraz, które zawierają *w dowolnym miejscu* daną parę kolejnych słów. Rozwiązanie takie zostało przyjęte ze względu na kompatybilność struktury z funkcjonalnością pasowania rozmytego, opisywanego w następnym podrozdziale. Gdyby efektywność samego tłumaczenia z dziurami była priorytetem, to należałoby wszystkie frazy języka źródłowego w słowniku oraz w bieżącym dokumencie poddać transformacji opisanej wyżej (tj. w obrębie fraz zamienić symbole na tokeny *[gap1]*, *[gap2]* itd.), a następnie użyć wyszukiwania dokładnego, dla którego istnieją proste i efektywne indeksy.

3.6. Pasowanie rozmyte

Funkcjonalnością zbliżoną do tłumaczenia z dziurami jest pasowanie rozmyte (ang. *fuzzy matching*). Ideą tego mechanizmu jest dostarczenie sugestii w przypadku, gdy frazy bieżącej nie ma w słowniku, nawet gdy pominiemy symbole (użyjemy mechanizmu dziur). W takiej sytuacji pozostaje jeszcze znalezienie fraz dostatecznie podobnych do bieżącej, aby rzut oka na zapisane w słowniku ich tłumaczenia pomógł tłumaczowi w podjęciu decyzji jak przetłumaczyć np. jakieś specjalistyczne słowo. Podamy tu kilka przykładów z realnego dokumentu i słownika, aby czytelnik miał lepsze wyobrażenie zagadnienia.

Fraza:
Cabinet glazed door polycarbonate window
Możliwe sugestie:
High protection degree according cabinets glazed door
Glazed door cabinet
Glazed door

Fraza:
two voltages
Możliwe sugestie:
all voltages
Standard voltages Us:

Fraza:
Frame size ranging
Możliwe sugestie:
Frame size
frame size only
M-Pact Plus frame size fixed mounted:

Zagadnienie wymaga rozwiązania dwóch problemów: (*i*) jak indeksować frazy, aby zapewnić szybkie zwracanie sugestii, (*ii*) dla zwróconej listy sugestii, jaką miarę podobieństwa zastosować, aby na szczycie listy były sugestie najtrafniejsze, najbardziej użyteczne.

Porównaliśmy dwa mechanizmy indeksowania: jeden oparty na ideach własnych, drugi wykorzystujący standardowy w MySQL mechanizm *match-against*.

Nasz indeks działa następująco. Frazy rozbijane są na słowa (na granicy białych znaków), usuwamy również wybrane znaki interpunkcyjne. Następnie usuwamy symbole oraz słowa pospolite (ang. *stop words*). Potrzebne są tu listy najpopularniejszych, lecz nie niosących osobnej informacji, słów w wykorzystywanych językach. Użyta w naszych eksperymentach lista słów pospolitych dla języka angielskiego liczy ponad 40 pozycji, m.in. *the*, *of*, *to*, *it*, *this*, *that*, *and*, *have*, *up*. Jeśli fraza otrzymana w wyniku tych filtracji liczy mniej niż dwa słowa, to nie indeksujemy jej. W przeciwnym razie do indeksu dodajemy jako klucze wszystkie pary sąsiadujących słów z wynikowej frazy, z nakładkami, a jako wartości frazę bieżącą. Jeśli dany klucz już w indeksie istnieje, to zbiór jego wartości powiększamy o frazę bieżącą. Zilustrujmy tę procedurę przykładem.

Bieżąca fraza to *Minimum space to earth metal*. Fraza nie zawiera symboli ani znaków interpunkcyjnych, ale należy usunąć słowo pospolite *to*, w wyniku

czego fraza redukuje się do czterech słów, a zatem wyodrębnić z niej należy trzy pary sąsiadujących słów: *Minimum space*, *space earth*, *earth metal*. Każda taka para słów jest kluczem (nowym lub już istniejącym, w zależności od sytuacji) w słowniku (tablicy asocjacyjnej), zaś do zbioru skojarzonych wartości dodawana jest cała fraza bieżąca, tj. *Minimum space to earth metal*.

Wyszukiwanie w naszym algorytmie w pierwszej fazie dokonuje ekstrakcji par słów danej frazy, zgodnie z metodologią opisaną wyżej, ale także rozszerza tę listę par słów o pary słów w kolejności odwróconej oraz pary z dziurą jednowyrazową. Przykładowo, dla frazy *Minimum space to earth metal and for arc chute removal* byłyby to (przypominamy o usunięciu słów pospolitych): *Minimum space*, *space earth*, *earth metal*, *metal arc*, *arc chute*, *chute removal*, jako pary słów kolejnych; *space Minimum*, *earth space*, *metal earth*, *arc metal*, *chute arc*, *removal chute* jako pary słów w kolejności odwróconej; oraz *Minimum earth*, *space metal*, *earth arc*, *metal chute*, *arc removal* jako pary z dziurą jednowyrazową. Jak pokazuje ten przypadek, wiele takich par wyrazowych może być bezsensownych, jednak przy dobrej procedurze dalszego szeregowania ich wg miary podobieństwa, nie ma to znaczącego wpływu negatywnego, poza (umiarkowanym) spowolnieniem wyszukiwania. Z drugiej strony, odwracanie kolejności słów i dziury międzywyrazowe mogą przydać się np. w języku polskim, gdzie np. kolejność epitetów określających podmiot w zdaniu jest często dość dowolna. Jako praktyczne zabezpieczenie przez bardzo długimi frazami opisana procedura nie wykracza poza prefiks frazy o długości 20 słów. Oznacza to, że nawet bardzo długie frazy mają szansę otrzymania sugestii, o ile tylko w indeksie zostały zapisane pary słów z ich początku. Dodajmy jednak, że użyteczność sugestii dla bardzo długich fraz jest i tak wątpliwa, gdyż czytanie kilku długich napisów i śledzenie na ile są one podobne do (długiej) frazy zadanej, na pewno nie jest wygodne i na dłuższą metę obciąża wzrok.

Dla wszystkich otrzymanych w opisany sposób par słów odczytujemy z indeksu skojarzone z nimi zbiory fraz i dokonujemy ich agregacji, z pominięciem duplikatów. Następnym etapem jest uszeregowanie otrzymanych sugestii z wykorzystaniem opracowanej przez nas miary podobieństwa.

Miara ta nie rozróżnia małych i wielkich liter. Dodatkowo wykonywany jest (bardzo prymitywny, ale jak się wydaje wystarczający dla naszych potrzeb) stemming słów: słowa 4-literowe redukowane są do pierwszych 3 liter; słowa 5-literowe i dłuższe do pierwszych 4 liter. Operacja ta, choć może prowadzić do pewnych fałszywych podobieństw, zmniejsza wrażliwość algorytmu na formy gramatyczne słowa (np. liczba pojedyncza kontra liczba mnoga), stojące nieraz po słowie kropki etc.

Następnie obliczamy podobieństwo tak przekształconych fraz, rozbitych na słowa, na podstawie długości ich wspólnego afiksu (prefiksu lub sufiksu), w zależności od tego, który dłuższy, przy czym w przypadku remisu preferowany jest prefiks, gdyż „wychwycenie" wzrokiem sugestii z długim prefiksem

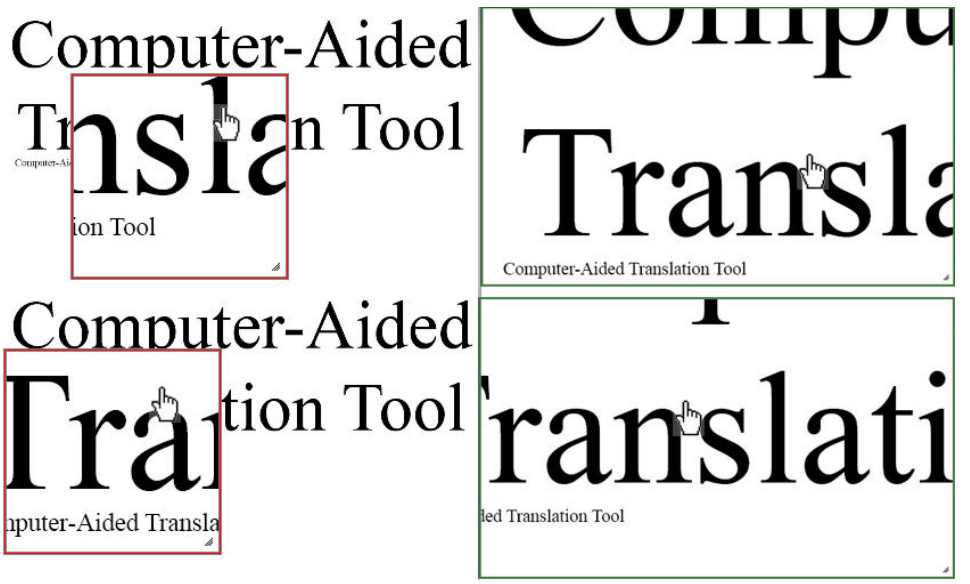
wspólnym z frazą bieżącą jest dla człowieka nieco łatwiejsze, bardziej naturalne niż w przypadku podobieństwa sufiksów. Poza wspólnymi afiksami zliczane są również słowa należące do obu zestawianych fraz (w dowolnej kolejności). Końcowa miara, o wartościach pomiędzy 0 a 1 (gdzie 1 oznacza dopasowanie dokładne, z dokładnością do opisanych przekształceń jak np. stemming), jest maksimum z trzech wielkości: długości wspólnego prefiksu i liczby wspólnych słów w pozostałym sufiksie (wziętą z pewną wagą < 1), długości wspólnego sufiksu i liczby wspólnych słów w pozostałym prefiksie (wziętą z pewną wagą < 1) oraz ogólnej liczby wspólnych słów (wziętej z pewną wagą < 1). Tak otrzymana liczba jest dzielona przez długość dłuższej z zestawianych fraz, w słowach. Iloraz ten udaremnia uzyskiwanie wysokich miar podobieństwa frazom dużo dłuższym od frazy zadanej. Ponownie uwzględniamy percepcję człowieka: umiarkowany jest pożytek z sugestii, która nawet zawiera krótką (np. 3-wyrazową) frazę wejściową w całości (z zachowaniem kolejności słów), jeśli sugestia ta liczy np. 20 słów i odszukanie w niej danej podsekwencji oraz następnie odszukanie tłumaczenia podsekwencji w zwróconym, i zapewne podobnie długim, tłumaczeniu sugestii, to strata np. kilkunastu sekund.

3.7. Lupa

W procesie lokalizacji dokumentów DTP istotną sprawą jest kontekst tłumaczenia, którego natura jest zwykle bardziej skomplikowana niż dla tekstu linearnego. W przypadku czystego tekstu zespół czynników wpływających na kontekst tłumaczenia jest zwykle mniejszy. Połączenie tekstu i obrazu rozszerza kontekst przez szereg wzajemnych relacji. Lokalizacja dokumentów DTP jest więc utrudniona z powodu braku relacji obrazu i treści, z czego może wynikać również niższa jakość tłumaczenia i powstające w nim błędy. Jednoczesne umieszczenie tekstu do tłumaczenia oraz kontekstu graficznego jest utrudnione m.in. poprzez percepcję człowieka, która nie pozwala na objęcie jednym spojrzeniem wielu szczegółów oraz z drugiej strony przez możliwości typowych obecnie monitorów. Problem rozwiązaliśmy stosując podgląd graficzny bardziej ogólny dla pierwszego spojrzenia (w mniejszej rozdzielczości), a bliższy i bardziej szczegółowy kontekst dostępny jest dzięki mechanizmowi lupy.

Nasze rozwiązanie polega na umieszczeniu prostokątnej lupy w prostokątnym podglądzie. Możliwe jest rozciąganie lupy (zmiana rozmiaru) w sposób ciągły za pomocą dolnego prawego rogu jej kontenera. Istnieją dwa tryby pracy ze względu na wielkość lupy. Pierwszy (rys. 2, po lewej) polega na przeciąganiu prostokąta przy użyciu znanej techniki *przeciągnij i upuść* w granicach większego prostokąta podglądu. Wyświetlane powiększenie jest wtedy fragmentem podglądu znajdującym się bezpośrednio pod lupą. Drugi tryb (rys. 2, po prawej) – pełnoekranowy – włączany jest w przypadku zbliżenia się

rozmiaru lupy do rozmiaru podglądu i skutkuje rozszerzeniem lupy na cały podgląd. Sterowanie zostaje przełączone na przeciąganie podglądem, a nie lupą.



Rys. 2. Różne tryby lupy. Po lewej: przesuwanie lupy w trybie normalnym (czerwone obramowanie lupy), po prawej: w trybie pełnoekranowym

Interfejs użytkownika dotyczący komponentu został stworzony za pomocą biblioteki jQuery UI zbudowanej na podstawie języka JavaScript. Podglądy w różnych rozdzielczościach generuje na żądanie system odpowiedzialny za współpracę z InDesign Server przy wykorzystaniu języka skryptowego ExtendScript oraz technologii SOAP. Trwałość danych oraz dynamizm zapewniają PHP oraz MySQL.

W przypadku rzeczywistej lupy ognisko przedmiotowe i ognisko obrazowe leżą na wspólnej prostej zwanej osią symetrii lupy. Środek powiększanego obszaru znajduje się w środku obszaru lupy, stanowiąc punkt stały przekształcenia (powiększenia). W naszym przypadku takie rozwiązanie dobrze sprawdzałoby się w środkowej części dokumentu, jednak aby wyświetlić róg lub brzeg dokumentu, obszar lupy musiałby częściowo znajdować się poza obszarem dokumentu. W opracowywanej aplikacji lupa nie może wykraczać poza obszar dokumentu, dlatego konieczne było znalezienie takiej konstrukcji lupy, która umożliwiłaby wyeliminowanie niewidocznego dla lupy obszaru brzegowego. W tym celu zastosowano dynamiczne przemieszczenie punktu stałego powiększanego obszaru względem obszaru lupy. Takie rozwiązanie pozwala na powiększanie dowolnego miejsca dokumentu, a zmiana położenia punktu stałego jest niedostrzegalna dla użytkownika.

3.8. Anotacja dokumentu

Prezentowany system zakłada, iż proces lokalizacji jest wieloosobowy. W związku z tym często istnieje konieczność komunikacji między członkami procesu na dowolnej linii miedzy tłumaczem, korektorem oraz menedżerem projektu. Poza konwencjonalnymi metodami komunikacji (np. drogą mailową) istnieje duża potrzeba komunikacji graficznej w kontekście tłumaczonego dokumentu, na co pozwala mechanizm anotacji dokumentu. Dodatkowa droga komunikacji pozwala przekazywać informacje dotyczące konieczności przesunięcia pewnych elementów na stronie, podmiany obrazów, zmiany kolejności.

Komponent składowy systemu pozwala na intuicyjne pisanie na podglądzie graficznym lokalizowanej strony dokumentu. Projekt zakłada korzystanie z podstawowych narzędzi edycyjnych, takich jak wpisywanie tekstu w wybranym miejscu. Każdy element anotacji znajdujący się na podglądzie ma możliwość przesuwania oraz zmiany rozmiaru. Pola tekstowe mają możliwość minimalizacji, która sprowadza je do ikon położonych w wybranym miejscu, co pozwala na zachowanie porządku w strefie podglądu. Anotacje widoczne są dla wszystkich członków biorących udział w tłumaczeniu: tłumacza, korektora, menedżera.

Komponent został oparty na bazie prawie takich samych technologii jak mechanizm lupy, z tą różnicą, że nie jest wykorzystywany InDesign Server.

3.9. Miara podobieństwa dwóch stron

Przy tłumaczeniu kolejnych wersji dokumentu często zdarza się, że większa część dokumentu składa się z tych samych, bądź bardzo podobnych stron, które w przeszłości zostały już przetłumaczone. Ważne jest, aby takie strony w całości lub w większych częściach były automatycznie tłumaczone przez system, tak by tłumacz mógł tylko przejrzeć tekst i w razie konieczności wprowadzić niezbędne poprawki. Potrzebny jest zatem sposób rozpoznawania takich stron przez system.

Wprowadzona miara podobieństwa stron PSM(x, y) (*Page Similarity Measure*) jest obliczana dla strony x aktualnie tłumaczonego dokumentu i strony y, wcześniej przetłumaczonej w ramach wybranego słownika tłumaczeń. Wartość PSM(x, y) należy do przedziału [0; 1] i może być interpretowana jako prawdopodobieństwo, że strony są takie same. Na jej podstawie można wybrać stronę y spośród wcześniej przetłumaczonych, najbardziej podobną do strony x aktualnie tłumaczonej i korzystając ze słownika strony y z odpowiednio wysokim prawdopodobieństwem dokonać automatycznego tłumaczenia strony x. PSM(x, y) jest zorientowana na podobieństwo stron pod kątem występujących na nich tekstów, jednakże z uwzględnieniem również ich geometrycznego położenia.

Zakładamy, że słownik tłumaczeń zawiera:

- wystarczającą informację by móc dokonać złożenia tekstów w obrębie poszczególnych ramek na stronie;
- informację o rozmieszczeniu geometrycznym ramek w obrębie strony (współrzędne lewego górnego rogu i prawego dolnego ramki podane w milimetrach);
- informację czy ramka jest ramką z obrazkiem.

Ponieważ ze struktury słownikowej przechowywanej w bazie danych zazwyczaj nie można odtworzyć prawidłowej kolejności całego tekstu w obrębie strony, podobieństwo tekstu będzie rozpatrywane w obrębie poszczególnych ramek tekstowych.

PSM(x, y) składa się z sumy czterech ważonych składowych:

1. Podobieństwo tekstu w obrębie ramek tekstowych – waga składowej: 0,8:
   a. każda ramka tekstowa strony x ma swoją wagę;
   b. waga ramki tekstowej jest równa odsetkowi liczby znaków, które zawiera ta ramka w stosunku do liczby wszystkich znaków zawartych w ramkach w obrębie rozważanej strony x. Zatem: ramka zawierająca więcej znaków będzie miała odpowiednio większą wagę od ramki zawierającej mniej znaków;
   c. wszystkie wagi ramek tekstowych strony x sumują się do jedynki;
   d. dla każdej ramki tekstowej ze strony x liczone jest podobieństwo pod kątem przechowywanego tekstu (słowa tworzące tekst rozdzielone są pojedynczym znakiem spacji) z każdą ramką ze strony y (wykorzystana została standardowa funkcja PHP *similar_text*(), implementująca algorytm z pracy [8]);
   e. ramki tekstowe są dobierane w pary (ramka ze strony x i ramka ze strony y) począwszy od par najbardziej podobnych, z tym że każda ramka może tworzyć tylko jedną parę;
   f. dla każdej pary ramek tekstowych, ich podobieństwo jest przemnażane przez wartość podobieństwa rozmieszczenia tych ramek (podobnie jak opisano niżej w punkcie 2, liczona jest odległość ramek i podstawiona do malejącej funkcji wykładniczej). Dzięki temu podobne pod kątem tekstu ramki będą miały mniejsze podobieństwo jeśli znajdują się w innych częściach strony. Tak uzyskana wartość będzie przemnażana przez wagę ramki ze strony x;
   g. dla każdej ramki ze strony x tak policzone wartości będą sumowane, dając ostateczną wartość składowej podobieństwa tekstu w obrębie ramek tekstowych.
2. Podobieństwo rozmieszczenia ramek – waga składowej: 0,1:

   a. dla każdej ramki ze strony x szukany jest najbliższy sąsiad-ramka ze strony y (z zachowaniem podziału: ramki tekstowe, ramki z obrazkami) w ten sposób, że odległości (w metryce euklidesowej) pomiędzy lewym górnym rogiem ramek i prawym dolnym rogiem są sumowane i wybierana jest suma najmniejsza;
   b. wynik jest podstawiany do malejącej funkcji wykładniczej a1b w miejsce zmiennej b (wartość podstawy a1<1 ustala się na drodze testów; ponieważ funkcja wykładnicza dla wartości dużo mniejszych niż 1 bardzo szybko zbiega do 0, powinno się rozważać wartości niewiele mniejsze od 1). Funkcja ta dla odległości 0 zwróci wartość 1, dla odległości większych od zera odpowiednio mniejszą wartość z przedziału (0; 1).
3. Liczba ramek tekstowych – waga składowej: 0,05:
   a. różnica pomiędzy liczbą ramek tekstowych ze strony x i strony y;
   b. wynik jest podstawiany do malejącej funkcji wykładniczej a2b w miejsce zmiennej b (wartość podstawy a2<1 ustala się na drodze testów, podobnie jak wartości a1 – patrz punkt 2b).
4. Liczba ramek z obrazkami – waga składowej: 0,05:
   a. różnica pomiędzy liczbą ramek z obrazkami ze strony x i strony y;
   b. wynik jest podstawiany do malejącej funkcji wykładniczej a3b w miejsce zmiennej b (wartość podstawy a3<1 ustala się na drodze testów, podobnie jak wartości a1 – patrz punkt 2b).

Należy zauważyć, że PSM(x, y) równa 1 nie oznacza, że strony są takie same. Oznacza jedynie, że teksty w obrębie ramek ze strony x są takie same jak w odpowiednich ramkach ze strony y (z dokładnością do białych znaków), dodatkowo ramki te są jednakowo rozmieszczone i liczba zarówno ramek tekstowych, jak i tych z obrazkami jest taka sama. Wartość równą 1 należy zatem traktować jako bardzo wysokie prawdopodobieństwo, że strony są takie same. Wiedza, czy strony są jednakowe nie jest nam jednak potrzebna. Wprowadzona miara ma jedynie wyznaczyć stronę y, która jest prawdopodobnie najbardziej podobna do strony x.

4. WNIOSKI KOŃCOWE I PLAN DALSZYCH PRAC

Przeprowadzone badania pozwoliły na ulepszenie kilku stosowanych wcześniej algorytmów, takich jak: rozpoznawanie zwrotów nie podlegających tłumaczeniu czy algorytm dynamicznego przemieszczania punktu stałego w transformacji powiększenia (mechanizm lupy) w zależności od pozycji obserwowanego miejsca na stronie. Opracowano również nowe algorytmy, dedykowane do tłumaczeń dokumentów DTP, takie jak: algorytmy generowania sugestii dla wyrażeń krótkich i dla wyrażeń długich, miara podobieństwa stron dokumentów DTP czy rozpoznawanie znaków towarowych, które pozwolą na

poprawę jakości pracy tłumacza w aplikacji lokalizacji dokumentów DTP wspomaganej komputerowo.

Dalsze prace, ukierunkowane będą na wdrożenie wyników zakończonych obecnie prac w jedną spójną aplikację, w której zaimplementowane zostaną wszystkie opracowane algorytmy oraz na zweryfikowanie poprawności opracowanych rozwiązań w warunkach „produkcyjnych", to znaczy w oparciu o dużą liczbę dokumentów oraz z udziałem wielu tłumaczy. Powinno to przyczynić się do wykrycia ewentualnych, nie rozpatrywanych przypadków lub wyjątków, a poprzez ich uwzględnienie doprowadzić do otrzymania poprawnie działającej i stabilnej wersji aplikacji.

Inny aspekt dalszych prac dotyczy oceny wydajności (skalowalności) zaproponowanych algorytmów. Na obecnym etapie zaproponowane algorytmy wydają się wystarczająco szybkie, niemniej użycie dużej liczby potencjalnych słowników może wymagać weryfikacji tego stwierdzenia. Przykładowo, w naszych testach algorytmu pasowania rozmytego, z indeksowaniem na bazie par sąsiadujących słów, średni czas zwracania listy sugestii dla danej frazy nie przekraczał 2 ms, a zatem generacja list sugestii dla przykładowej strony liczącej ok. 50 fraz zajmuje mniej niż 0,1 s, co jest opóźnieniem niedostrzegalnym dla użytkownika. Test ten jednak dotyczył słownika stosunkowo małego, liczącego ok. 7,5 tys. fraz, i choć wydłużenie czasu przy rosnącej wielkości słownika niewątpliwie będzie subliniowe, to jedynie eksperymenty mogą wykazać do jakiej wielkości bazy użyty algorytm nie powoduje zmniejszenia komfortu pracy z aplikacją.

Podobnie mechanizm miary podobieństwa stron wymaga inżynierskich prac pod kątem podwyższenia jego wydajności, co wydaje się ważne nawet jeśli założymy, że proces wprowadzania dokumentu do systemu i jego analizy ma charakter jednorazowy. Porównywanie każdej pary stron jest z pewnością dużo wolniejsze niż użycie odpowiedniego haszowania na poziomie fraz lub nawet większych bloków. Możliwa jest też prosta heurystyka przerywania procedury porównań, przy której strony danego dokumentu porównywane są w pierwszej kolejności z tym dokumentem (dokumentami), w którym (których) już zostało znalezione kilka bardzo podobnych stron. Jeśli w dokumencie referencyjnym znajdziemy stronę wystarczająco podobną do strony bieżącej, to pomijamy już dla bieżącego wyszukiwania inne strony zgromadzone w systemie.

Możliwe są dalsze, nie rozważane dotąd, funkcjonalności, które podwyższą jakość pracy tłumacza. Jedną z nich jest wyszukiwanie literówek w zgromadzonych słownikach. Podejście najprostsze, tj. bezpośrednie użycie dostępnego spellcheckera (np. Aspell) dalekie jest od doskonałości, z uwagi na dużą liczbę symboli czy nazw własnych w typowych dokumentach, które spellchecker będzie oznaczał jako „błąd". Rzecz jasna, symbole (oraz znaki towarowe) wykryjemy przy pomocy naszych algorytmów opisywanych w sekcjach 3.3 i 3.4, natomiast przesłanką do wskazania nazwy własnej jest użycie

wersalików czy pierwszej wielkiej litery, ale za wyjątkiem początku zdania czy sytuacji, gdy całe zdanie / fraza jest napisane(-a) wielkimi literami. Dodatkowo silną przesłanką literówki może być nietypowa (dla danego języka) zbitka liter, co można szybko wykryć przy pomocy przechowywanej w pamięci tablicy typowych dla danego języka digramów (lub trigramów). Na tym przykładzie widać, jak ważne jest opracowanie skutecznych i wydajnych obliczeniowo „cegiełek" algorytmicznych (np. mechanizmu wykrywania granic zdań, mechanizmu rozpoznawania symboli), które mogą się przydać jako elementy składowe bardziej złożonych funkcjonalności.

Inny możliwy mechanizm korekty znajdowałby w zgromadzonych słownikach te frazy, które są identyczne (przy pominięciu symboli, wielkości liter, możliwych znaków przestankowych na końcu etc.) w języku źródłowym, lecz istotnie różnią się w tłumaczeniach. Rozbieżność taka może być uzasadniona (np. polskie słowo *wzorzec* może, w zależności od tematyki dokumentu, być tłumaczone na angielski jako *pattern* albo *template*), ale w wielu przypadkach będzie wynikała z nieuwagi czy niekompetencji tłumacza i „ręczna" analiza tego typu przypadków powinna prowadzić do podwyższenia jakości słowników. Bardziej wyrafinowana wersja tego mechanizmu mogłaby zrezygnować z identyczności fraz w języku źródłowym na rzecz zaledwie ich podobieństwa (wykorzystanie pasowania rozmytego).

PODZIĘKOWANIA

Autorzy składają podziękowania pp. Bartoszowi Koziakowi, Maciejowi Smolczewskiemu, Damianowi Zarębskiemu i Tomaszowi Zielińskiemu za udział w implementacji wybranych modułów opisywanej aplikacji. Badania zostały przeprowadzone we współpracy z firmą Medialab i dofinansowane w ramach Programu Operacyjnego Innowacyjna Gospodarka, Działanie: Wsparcie na prace badawcze i rozwojowe oraz wdrożenie wyników tych prac, Nr: UDA-POIG.01.04.00-10-016/09.

LITERATURA

[1] **Poudat C.:** La traduction automatique en libre accès sur l'Internet. Le français dans le monde, 2001, no. 314, pp. 51-52.

[2] **Trujillo A.:** Translation engines: techniques for machine translation. London, Springer 1999.

[3] **Lagoudaki E.:** Translation Memories Survey 2006. Translation Memory systems: Enlightening users' perspective. Imperial College London, 2006, 39 str., raport dostępny pod http://www.atril.com/docs/tmsurvey.pdf

[4] **Knuth D.E.:** The Art of Computer Programming. Vol. 3, Second Edition. Reading, MA: Addison-Wesley, 1998.

[5] **Philips L.:** Lawrence Philips' Metaphone Algorithm. Dokument dostępny pod http://aspell.net/metaphone/

[6] **Nowak G., Grabowski Sz., Draus C., Zarębski D., Bieniecki W.:** Designing a computer-assisted translation system for multi-lingual catalogue and advertising brochure translations. Proc. 6th Int. IEEE Conf. MEMSTECH 2010, Lviv-Polyana, Ukraine, pp. 175-180.

[7] **Grabowski Sz., Draus C., Bieniecki W.:** Recognizing non-translatable symbols in a multi-lingual computer-assisted translation system for DTP documents. Automatyka 2010 (przyjęty do druku).

[8] **Oliver J.J.:** Decision Graphs – An Extension of Decision Trees. Proc. 4th Int. Conf. Artificial intelligence and Statistics, Fort Lauderdale, Miami, USA, 1993, pp. 343-350.

REVIEW OF ALGORITHMIC AND ENGINEERING PROBLEMS IN A COMPUTER-AIDED TRANSLATION APPLICATION HANDLING MULTI-LANGUAGE DTP DOCUMENTS

Abstract

We present and discuss a number of problems related to effective translation of product catalogues and advertising brochures with a CAT (Computer-Aided Translation) application. CAT tools usually work on small text phrases (snippets) organized into so-called Translation Memories (TM). Those tools make it possible to navigate freely over the document, automatically translate recognized phrases and prompt suggestions for translating phrases similar to ones already found in the system, search and update the TMs, and more. The problems and issues we consider here can generally be divided into those related to the user interface and those based on text algorithms. In particular, we solved the problems of symbol detection (where "symbols" are sequences of characters which should not be translated, like numbers, abbreviations of physical units, product codes, reference numbers, registered symbols and trademarks etc.), TM editing, document annotation, translating with gaps, fuzzy matching. Those functionalities speed up the work of a translator (e.g., by minimizing the probability of occurrence of some classes of errors in the translation process) and make the management and maintenance of the document and TMs easier. In this way, the document release cycle is shortened, which is of utmost importance for the DTP documents which require parallel translation into many languages (catalogues, advertising brochures).

Politechnika Łódzka
Katedra Informatyki Stosowanej