Szemantikusan annotált dokumentumok létrehozása szövegfeldolgozó eszközök támogatásával

Héder Mihály

MTA SZTAKI

mihaly.heder@sztaki.hu

1. Bevezető

A szemantika fogalmát a nyelvészettől kezdve a formális és matematikai logikán át az intencionális logikáig terjedően több tudományág definiálja, többé-kevésbé hasonló módon. Ezen definíciók közös jellemzője, hogy a szemantikát mint egy relációt határozzák meg, amely egy adott valóságos entitást és ennek valamely nyelven megadott reprezentációját köti össze. Az említett reprezentációt gyakran nevezik az entitás szintaktikájának.

A világhálón a reprezentáció általában írott szöveg és jelölőnyelv egyvelege. Ezen adaton a számítógép végre tud hajtani néhány funkciót: megjelenítés, továbbítás, tárolás, illetve keresés kulcsszavakkal vagy mintaillesztéssel.

A gépi következtetés és a valóságos tartalmi keresés túlmutatnak ezen funkciókon, mivel feltételük a reprezentáció egyfajta korlátozott, gépi megértése. Gépi megértés alatt itt szerényen azt a tényt értsük, hogy a számítógép képes felismerni a reprezentáció egyes elemei és az általa ismert való világbeli entitások közötti kapcsolatot. Természetesen a való világbeli entitásokat csak előzetes programozással tudjuk bevinni a gépbe, melynek során ontológiák, illetve tezauruszok segítségével megadjuk ezek egymáshoz való viszonyát, és meghatározzuk az egyes entitás típusokon végrehajtható funkciókat.

A szemantikus web témaköre azzal foglalkozik, hogy hogyan érdemes mindezt a világhálón található adatokkal és számítógépekkel megvalósítani.

2. A szemantikus web

A szemantikus web elképzelés egy, a web jelenlegi technológiáira és az RDF leírónyelvre épülő keretrendszert jelent, amely strukturált, gép által is értelmezhető, egymással jól definiált relációban álló adatok tömegét tárolja elosztott módon. A cél az, hogy az adatok megoszthatók, újrafelhasználhatók legyenek a vállalati és közösségi határokon át is.

A rendszer --- a területen a legtöbb fejlesztést végző W3C elképzelésében --- több rétegű.
A legalsó réteg a nyers adatok tárolási formátumait (pl.: XML, HTML) tartalmazza. E felett szükség van egy jelölőnyelve, amely a nyers

adatokban rejlő információt a gép számára felismerhetővé teszi. Erre a feladatra alkalmas a W3C által specifikált RDF nyelv. A nyelvben kijelentéseket írhatunk le, minden kijelentés egy hármasból áll:

<Alany>
<egy reláció vagy tulajdonság>
<objektum vagy érték>.

-Az alany azt az entitást jelöli, amire vagy akire a kijelentés vonatkozik. Az RDF nyelvben ez bármi lehet, ami rendelkezik egy univerzális erőforrás azonosítóval (URI). Ez nem túl erős megkötés, ugyanis URI-ja bárminek lehet. Egy RDF állítás alanya lehet például egy személy, egy tárgy, egy esemény, egy műalkotás, stb. Különleges esetekben egy másik RDF kifejezés, vagy egy tulajdonság is.
-A hármas második tagja egy tulajdonság vagy egy reláció neve lehet. Egy tulajdonság neve azt jelöli, hogy az alany mely tulajdonságáról beszélünk. Egy személy esetén lehet a kora, neme, stb. A tulajdonság mindig egy URI-val azonosítható. Reláció esetén ezen a helyen azon reláció megnevezése áll, amely az alany és az objektum között fennáll. Egy személy például állhat ``testvére'' relációban egy másik személlyel.
-A hármas utolsó tagja egy tulajdonság értéke vagy egy objektum azonosítója. Ha például az alany egy személy, a tulajdonság egy életkor, akkor az érték egy szám lehet. Objektumok esetén URI-t használunk az azonosításra.

Egy speciális tulajdonság az RDF-ben az alany osztálya (a típus és a szerep szó is használatos az osztály helyett). A szemantikus web elképzelésben az RDF feletti szint az ontológiák szintje, amelyben az osztályok hierarchiáját írhatjuk le, illetve általában fogalmazhatunk meg állításokat amelyek az osztály minden tagjára igazak. Ontológia leíró nyelv például az RDF Séma és a Web Ontology Language.

Az RDF-ben megadott információ és az ontológiák segítségével már következtetéssel válaszolhatunk meg kérdéseket vagy hozhatunk létre új állításokat.

Szemantikus annotáció alatt egyfajta a reprezentáció mellé helyenként felírt széljegyzetet értünk, amely egy kötött szintaktikával elmagyarázza a gép számára hogy mi a jelölt rész szemantikája, például RDF hármasokban kifejezve. Az RDF elképzelés szerinti alany, tulajdonság, érték hármasból álló adatmodellnek számos reprezentációja lehetséges, a céltól függően. Ezeket a reprezentációkat mutatja be a következő szekció.

3. Szemantikus adatok tárolása a szövegben

HTML metaadat

Bármely HTML dokumentumban van lehetőségünk RDF hármasokat leírni a meta elem segítségével. E módszer hátránya, hogy kizárólag a dokumentum egy megadott pontján, a head elemen belül lehet elhelyezni, így a szöveg egy adott részletéhez nem csatolható. Példa HTML Metaadatra:

```
 <HEAD profile="http://www.acme.com/profiles/core">
```

```
  <TITLE>How to complete Memorandum cover sheets</TITLE>
  <META name="author" content="John Doe">
  <META name="copyright" content="&copy;Acme Corp.">
  <META name="keywords" content="corporate,cataloging">
  <META name="date" content="1994-11-06T08:49:37+00:00">
 </HEAD>
```

RDF

Az RDF szabvány definiál egy XML szintaxist a hármasok kifejezésére. Egy RDF fájlban tetszőleges számú állítást tehetünk. Az XML fájl sémája azonban nem engedi meg, hogy az állításokat nagy mértékben vegyítsük egyéb névtérből vett XML részletekkel, például egy cikk tartalmával. Egy RDF XML fájl részlete:

```
<rdf:Description
     rdf:about="http://www.w3.org/">
  <ex:editor>
    <rdf:Description>
      <ex:homePage>
        <rdf:Description
       rdf:about="http://purl.org/net/dajobe/">
        </rdf:Description>
      </ex:homePage>
    </rdf:Description>
  </ex:editor>
</rdf:Description>
```

GRDDL

(Gleaning Resource Descriptions from Dialects of Languages) egy nagyon egyszerű W3C szabvány, amely egy tetszőleges XML (pl.: XHTML) dokumentumban csak egyetlen egy attribútum meglétét írja elő. Ez az attribútum egy XSL transzformáció helye, mely transzformációt a dokumentumon alkalmazva megkapjuk annak RDF XML reprezentációját. Ez a megközelítés lehetővé teszi, hogy a saját jelölésrendszerünket használjuk a szemantikus annotációk elhelyezésekor és mégis szabványosak maradjunk. Példa a GRDDL használatára:

```
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:grddl='http://www.w3.org/2003/g/data-view#'
   grddl:transformation="glean_title.xsl
   http://www.w3.org/2001/sw/grddlwg/td/getAuthor.xsl"
>
<head>
<title>Are You Experienced?</title>
[...]
</html>
```

Mikroformátumok

Egyre nagyobb népszerűségnek örvendenek az úgynevezett mikroformátumok. Ez a technológia az (X)HTML rendkívül rugalmasan használható class mechanizmusára épül. Az elgondolás lényege, hogy bizonyos class értékek speciális jelentéssel rendelkeznek, amely megadja, hogy mi az adott elem tartalmának gépileg is értelmezhető

szemantikája. Az elgondolás előnye, hogy a dokumentumaink továbbra is szabványos HTML dokumentumok maradnak, továbbá CSS segítségével rögtön praktikus megjelenítést is adhatunk a szemantikus információnak. Mikroformátum például a hCard, a vCard HTML-be ágyazható verziója. A vCard maga az RFC2426-ban került definiálásra. Egy hCard részlet:

```
<span class="tel">
 <span class="type">home</span>:
 <span class="value">+1.415.555.1212</span>
</span>
```

RDFa

Ez a W3C által fejlesztett igen ígéretes technológia 2008-ban érte el az ajánlás státuszát. Az RDFa egy módszert ad arra, hogy tetszőleges XML fájl tetszőleges részletét RDF hármassá alakítsuk néhány plusz attribútum segítségével. A szabványtervezet számos hasznos példát tartalmaz arra, hogy hogyan érdemes az RDFa-t használni XHTML dokumentumokban, de nem zár ki bármely más XML formátumot sem. Egy RDFa-val annotált XHTML dokumentum részlete:

```
<div xmlns:dc="http://purl.org/dc/elements/1.1/">
   <h2 property="dc:title">The trouble with Bob</h2>
   <h3 property="dc:creator">Alice</h3>
   ...
</div>
```

4. Robusztus annotációk

A szemantikus annotációk életciklusának későbbi részei épp oly fontosak, mint a létrehozás pillanata. Ha fennáll a lehetősége annak, hogy a tartalom még változni fog, akkor külön figyelmet kell fordítani arra, hogy az annotációk ne kerüljenek rossz helyre és ne szenvedjenek sérülést.

Ami az annotációk strukturális épségét illeti, Yan Bodain és Jean-Marc Robert 2007-ben megfogalmazott öt követelményt magukra az annotációkra:

-Robusztus horgonyok : az annotációnak együtt kell maradnia az annotált szöveggel, akkor is, ha áthelyezik azt.
-Átlátszóság : A felhasználónak látnia kell az annotációk helyét, de a teljes RDF leírást nem feltétlenül. Fontos, hogy az interfész elsődlegesen magát az annotált dokumentumot mutassa.
-Többféle ontológia : Lehetőleg hagyni kell, hogy a felhasználó tetszőleges ontológiából származó RDF fogalmakat használhasson
-Változó granularitás : Kis szövegrészektől kezdve akár az egész dokumentumig terjedő méretű részeket is képes legyen annotálni a felhasználó.
-Annotáció-frissítések : a szöveg változásakor fel kell tételezni hogy az annotációk is módosításra szorulhatnak, ezen módosítások elvégzésében a felhasználót segíteni kell.

Azon annotációkat, amelyek kielégítik a fentieket, a szerzők robusztusnak nevezik.

5. A Docuphet működése

A Docuphet-tel a felhasználó egy web-es szerkesztőn keresztül kerül kapcsolatba. Az AJAX-szal ellátott webalkalmazás bizonyos időközönként elküldi a szervernek a már elkészült tartalmat. A szerver a szöveget különféle moduloknak továbbítja, melyek mindegyike valamely szempontból vizsgálja azt. Az egyes modulok annotációs javaslatokat fogalmazhatnak meg. Az annotációs javaslatok tartalmaznak egy konfidencia-értéket arra vonatkozóan, hogy a modul mennyire tartja valószínűnek, hogy az adott javaslat helyes. Egy bizonyos szint alatt a javaslatok automatikusan elvetésre kerülnek, és lehetőség van arra is, hogy egy adott szint felett automatikus elfogadásra kerüljenek kérdés nélkül. A szintek konfigurálhatók a rendszer finomhangolásakor. Egy annotációs javaslat tartalmaz egy eldöntendő vagy választásos kérdést. Eldöntendő kérdés esetén a javaslat egy szemantikus annotációt tartalmaz, amelyet a válasz függvényében a szerkesztőprogram vagy elvet, vagy beilleszt a szövegbe. Amennyiben több válaszlehetőség van, mindegyikhez tartozik egy annotációs javaslat, továbbá a felhasználó dönthet úgy, hogy nem fogadja el egyik javaslatot sem.
Ha a felhasználó befejezte a szerkesztést, a szöveget az annotációkkal együtt menti el a szerver.

5.1. Névelem-felismerés

Egy tipikus információkinyerési feladat a névelemek felismerése a szövegben. Névelem például egy tulajdonnév, egy cégnév, egy város, egy termék neve. De ide kapcsolódik a dátumok felismerése is. Névelem továbbá a fentieken kívül minden olyan karaktersorozat, amely egy entitást egyértelműen azonosít, például egy telefonszám vagy egy e-mail cím.
A névelemek felismerésének nehézsége függ a típusuktól. A telefonszámokat, e-mail címeket és dátumokat viszonylag könnyű szabályok segítségével, például reguláris kifejezésekkel felismerni. A tulajdonnevek egy részét hatékonyan azonosíthatjuk egyszerű listák segítségével, például egy keresztnév-lista felhasználásával. Cégek és termékek neveit azonban csak jóval nehezebb módon, sok tényező figyelembevételével tudjuk felismerni, mivel ezeket mindenféle megkötés nélkül találják ki. Ilyen esetekben figyelembe vehetők alaki jegyek (például nagy kezdőbetű), az előfordulás helye a mondaton, illetve a szövegen belül, szófaji elemzés, az előfordulás gyakorisága, a névelemek környezete.

A Docuphet rendszer részeként megvalósításra került egy általános névelem-felismerő rendszer, a Java nyelven implementált JNER. Ez a rendszer kiterjesztések segítségével működik, mindegyik kiterjesztés a saját módszerével és a saját szempontjai szerint elemzi az előzetesen már tokenekre bontott szöveget. Egyes kiterjesztések reguláris kifejezések segítségével működnek, mások katalógusok alapján (pl.: Keresztnév-katalógus, cégnév végződés katalógus, helységkatalógus) és lehetőség van tetszőleges új kiterjesztés írására is. Ez lehetővé teszi külső eszközök, például egy morfológiai elemző használatát. Ugyanilyen külső kiterjesztés lehet

még egy weben elérhető adatbázis (pl.: IMDB, DMOZ),vagy keresőmotor (pl.: Google, Wikia Search)
Bár nem tekinthetők szigorú értelemben vett névelemeknek, a JNER segítségével ismerjük fel a foglalkozásneveket (festő, zeneszerző, stb), illetve az emberek jellemzésére használt jelzőket (magas, szőke) is.

5.2. Információkeret-felismerés

Egy másik lehetséges információkinyerési alkalmazás az információkeretek felismerése. Egy információkeret egy olyan RDF

<alany>
<reláció>
<objektum>.

hármas, amelynek legalább egyik tagja ismert, a másikat vagy másik kettőt változónevek helyettesítik. Egy példa:

<X (személy)>
<születési hely>
<Y (földrajzi hely)>.

Az információkeret felismerése azt jelenti, hogy a ``Petőfi Sándor Kiskőrösön született 1823. jan. 1-én.'' mondatban felismerjük a

<Peto"fi Sándor>
<születési hely>
<Kisko"rös>.

információkeret-példányt.

Látható, hogy ennél a típusú feladatnál nagyon hasznos, ha a bemeneti szövegben már ismerjük a névelemeket és azok típusait. Bizonyos esetekben további megkötéseket is tehetünk, például hogy az információkeret ismeretlenei egy bekezdésen vagy egy mondaton belül helyezkednek el. Ez a megkötés bár korlátozza a felismerhető keretek számát, több praktikus előnnyel jár: meg lehet kezdeni az elemzést még az előtt, hogy a teljes szöveg rendelkezésre állna; továbbá jelentősen csökkenti a végrehajtási időt.

5.3. Mondathatár-felismerés

A Docuphet rendszer az információkeretek felismerésének segítése érdekében tartalmaz egy mondatokra bontó komponenst, a JSentence-t.

6. Mintaalkalmazások

Ebben a fejezetben a Docuphet két kísérleti alkalmazása kerül bemutatásra. Az egyik egy életrajzokat gyűjtő oldal, a BioBase. Itt ismert személyek biográfiáit gyűjtjük fél oldaltól legfeljebb két oldalig nyúló terjedelemben, hasonlóan a Magyar Életrajzi Lexikon bejegyzéseihez. A Docuphet ebben a konfigurációjában kizárólag a felhasználó által bevitt szöveg elemzésére szorítkozik. Ezzel szemben a másik megoldásban, a lakáshírdetéseket közvetítő FlatBase-ben rendelkezésre áll egy lista, amely felsorolja, hogy milyen alapvető információk megléte elvárt egy hírdetésben. Amíg a

felhasználó szerkeszt, a rendelkezésre álló tartalomban ezeket az információkat megpróbáljuk felismerni. Amikor a felhasználó menteni akar, ellenőrizzük a listát és ha arról hiányoznak még elemek, a megkérjük, hogy írjon még (továbbra is szabad szövegben) vagy direkt kérdéseket fogalmazunk meg. Mindkét alkalmazás magyar nyelvre van felkészítve.

6.1. BioBase

A BioBase rendszerben kétrétegű információkinyerés történik. Az első rétegben a névelemek felismerése zajlik:

-A JNER komponenes az összes magyar nyelvhez rendelkezésre álló szabálya segítségével elemzi a szöveget. Ezek a szabályok jelenleg: férfi és női név felismerés reguláris kifejezések és keresztnév-katalógusok segítségével, helységnév-felismerés katalógus segítségével, e-mail cím és telefonszám felismerés reguláris kifejezések segítségével, foglalkozásnév-felismerés katalógus segítségével, dátum felismerés reguláris kifejezések, katalógusok és egy egyedien erre a célra készített Java komponens segítségével, nemzetiség-felismerés katalógussal.

-A JNER minden felismert NE-hez egy pontszámot is ad. Minden 0.95 pontszám feletti NE alapján egy annotációs javaslat készül, amely az alábbi RDF hármast tartalmazza:

<cikk id> <kapcsolatos [NE típus]> <[NE érték]> .

Ahol a NE típus és NE érték praraméterek valódi értékkel kerülnek kitöltésre.
Az annotációs javaslatnak, az NE javaslathoz hasonlóan van egy pontszáma. Ez ebben az esetben 0.95-re van állítva. A Content Editor minden 0.9 feletti javaslatot automatikusan elfogad, ezzel elkerülve a felhasználó felesleges zavarását túlzottan triviális kérdésekkel. Az annotációk target értéke (az a hely a dokumentumon belül, ahová az annotáció kerül) az adott bekezdés előtti DOM csomópontra van állítva. A cikk elmentése után pusztán ezekből az annotációkból már lehetséges egy ``Tag felhő'' létrehozása, amelyben a tag-ek típussal(Név, Helyiségnév, stb.) is rendelkeznek.
- A 0.8 és 0.95-ös pontszám közötti NE-k alapján újabb, legfeljebb három darab annotációs javaslat készül, 0.8-as pontszámmal. Ezekhez a javaslatokhoz az ``Ehhez a cikkhez kapcsolódik az alábbi Személy/Helyszín (Érték)?'' kérdést rendeljük, a target és az RDF hármas ugyanaz, mint az előző pontban.

A felismert NE-k alapján az alábbi az alábbi információkinyerési kísérleteket tesszük:

- Ebben a lépésben megpróbáljuk azonosítani a személyt akiről a cikk szól. Ehhez egy javaslatot fogalmazunk meg az első személy típusú NE-ről akit találunk (ez gyakran a címben szerepel). A javaslatban szereplő RDF hármas:

<cikk id> <leírja> <[NE érték]>.

A target a cikk eleje, a pontszám 0.8. Ha az annotáció nem sikeres, a további személyekkel is megpróbáljuk ugyanezt.

- Ha a cikk alanya ismert, a születési, illetve halálozási dátum és idő kerül meghatározásra. Ehhez az alany környezetében előforduló dátum és helység típusú NE-ket vizsgáljuk. A ``született'', illetve ``elhúnyt'' múlt idejű lemmák jelenléte az adott NE környezetében magasabb konfidenciát jelent (bár ezek gyakran hiányoznak).

- A nemzetiség, a foglalkozás és a szülők nevének azonosítása az előző pontban leírthoz hasonló módszerrel történik, bizonyos lemmák bevonával (pl: apja, anyja), ahol szükséges.

Ezen annotációk felhasználásval az alanyok kategorizálhatók foglalkozás, nemzetiség, helység, vagy a kor szerint amelyben éltek. Minden RDF property amelyet előállítunk a BioBase saját névterében van, de könnyen leképezhető más névterekre (például FOAF), ha szükséges.

Az FlatBase rendszerben is két rétegű információkeret-felismerés történik, de a BioBase-től teljesen eltérő módon. Az első szinten az alábbi, fontosabbnak ítélt információkat próbáljuk meg kinyerni a lakáshírdetésből:

-típus : Ház, Házrész, Lakás, Nyaraló, stb.
-a hírdetés típusa : Kiadás, eladás.
-elhelyezkedés : Az ország mely részén található az ingatlan.
-Árkategória
-Méret
-Az eladó elérhetőségei

Ha a fenti listából egy elem ismertté válik, az új inforáció alapján leszűkített, részletekre irányuló IF kinyerés indul. Néhány példa:

- Részletek az elhelyezkedésről (kerület, városrész pl.: Lágymányos, utcanév)
 -Az ingatlan építéséhez használt alapanyag (tégla, panel, stb.)
 -Lakásoknál: melyik emeleten helyezkedik el? Legfölső emeleti-e?
 -Lakások kilátása: utcai, udvari?
 -Házak esetében: kert mérete
 -Fűtés típusa (a lakás típusa függvényében más katalógust használunk)
 -Szobák elhelyezkedése (külön nyíló?)
 -Tömegközlekedés a közelben (várostól függő típusok)

Mindkét szinten egyedileg konfigurált JNER példányokat használunk. A konfiguráció kezdetben nagyobb földrajzi helyek, városok listáját tartalmazza, valamint reguláris kifejezés alapú szabályokat az árra, méretre és elérhetőségekre vonatkozóan. Az IF kinyerés a felismert NE-ken és bizonyos lemmákon alapul, például építőanyag nevek, fűtés típusok, stb. Egy-egy új információ függvényében a JNER-ek újrakonfigurálásra kerülhetnek.

A rendszerbe be van építve egy lista azokról az információkról, amelyek megadása mindenképpen elvárt a hírdetésben. Ha a felhasználó a cikk elmentésére készül, de még nem minden információ ismert erről a listáról, a rendszer megkéri, hogy írjon ezekről. Ezek után ha továbbra is hiányoznak az információk, a rendszer egy limitált számú, direkt kérdést tesz fel ezekről, de minden kérdést legfeljebb egyszer. Ha így sem áll elő a kívánt információ, akkor a hírdetést hiányosan mentjük el.

7. Konklúzió

7.1. Elméleti eredmények

A névelem-felismerésben, az információkeret-felismerésben és az RDFa technológiában szerzett tapasztalatokon kívül megfogalmazható néhány általános szabály a javaslatokon alapuló annotációs rendszerek működésére.

A 4. fejezetben már bemutatásra került, hogy milyen strukurális követelményeket lehetséges és érdemes megfogalmazni az annotációkkal szemben.

Ezek a követelmények jól megalapozottnak tűnnek. A Docuphettel végzett kísérletek alapján úgy tűnik, helyénvaló néhány további ajánlást is megfogalmazni bármely javaslatokon alapuló annotációs rendszer működésével kapcsolatban:

-Ugyanazt a kérdést soha nem szabad kétszer feltenni, mivel ez nagyon flusztráló a felhasználó számára. Ezen követelmény továbbiakat von maga után:
-Minden javaslatot valamilyen formában tárolni kell, még azokat is, amelyeket a felhasználó nem tudott megválaszolni vagy elutasított. A munkamenet alapú tárolás többnyire nem elegendő, mivel egy dokumentumot gyakran ugyanaz a felhasználó néhány órás-napos időközönként szerkeszti. További kísérletezést igényelne annak felderítése, hogy a felhasználónkénti vagy a dokumentumonkénti tárolás-e a célravezetőbb.
-Ez legalább két fajta annotációs javaslat kialakulásához vezet: azon javaslatok, amelyekből annotáció keletkezett, illetve azok, amelyeket a felhasználó elutasított. Érdekes kutatás lenne kialakítani egy módszert, amellyel az elutasított annotációkból származó ``negált'' információt is hasznosítani lehet.
-A publikációban leírt ötödik szabály alapján az annotációkat minden egyes módosításnál újra kell értékelni. Ezt a követelményt nagyon nehéz teljesíteni az 1. pontunk együttes figyelembe vételével (Miszerint nem ajánlott ugyanazt a kérdést többször feltenni). Elméletileg ugyanis egy módosítás az annotációk tetszőleges részhalmazát invalidálhatja . Ha teljes bizonyossággal minden annotáció igazságát meg akarjuk őrizni, akkor vagy teljesen meg kell értenünk, hogy a változtatás hogyan és melyik annotációkat érinti, vagy az elfogadott annotációkhoz tartozó összes kérdést újra fel kell tennünk. Az előbbi megoldás nyilvánvalóan nem reális a jelenlegi tudományunk segítségével, az utóbbi viszont gyakorlatilag használhatatlanná tenné a kérdéseket újra és újra feltevő eszközt.

Másrészt a tapasztalat azt mutatja, hogy a szövegmódosítások többségének csak lokális hatása van. Ezt a jelenséget kihasználva az alábbi két feltevést alkalmaztuk:

--Limitált hatókör: Két annotáció típust definiáltunk: alap és származtatott. Az alap annotációk csupán a szöveg egy részére vonatkoznak, például egy bekezdésre. Feltesszük, hogy a meghatározott részen kívül történő változtatások nincsenek hatással az annotációra. Hogy ez a feltevés a lehető legtöbb esetben igaz legyen, az alap annotációk nagyon egyszerűek: pl.: ``ez a dokumentum kapcsolatban van Budapesttel'', ha a Budapest token szerepel a szövegben. Az ilyen típusú annotációk általában névelem-felismeréssel jönnek létre, lásd a 6.1. fejezetet. Ezeket az annotációkat mindig újraértékeljük, ha a vonatkozó szöveg változik.

--Függőségek: A rendszerben egy függőségi gráfot tartunk nyilván. A származtatott annotációk függhetnek vagy alap annotációktól, vagy más származtatott annotációktól. A függőség nyilvántartása annotáció azonosító lista használatával történik minden származtatott annotáció esetében. Minden esetben, ha egy annotáció tartalma megváltozik, a tőle függő annotációkat újra kell értékelni. Ha egy annotáció törlésre kerül, akkor a származtatott annotációkat is törölni kell. Ugyanakkor egy származtatott annotáció újraértékelése szükséges lehet más annotációk változásaitól függetlenül is.

-Az egyszerre feltett kérdések számát limitálni kell. Ez akkor különösen fontos, ha a felhasználó külső forrásból nagy mennyiségű szöveget illeszt be. Ilyen esetekben több tíz javaslat is létrejöhet egyszerre. A vonatkozó kérdéseket ilyen esetben több fázisban érdemes feltenni.

7.2. Összefoglalás

Azáltal, hogy hogy az Internet a hétköznapi ember mindennapi használati eszközévé vált, a szöveges tartalom mennyisége rohamosan nő. Az elviekben rendelkezésre álló technológiák, jól definiált szabványok és erős kutatási szándék ellenére is még mindig igaz, hogy a tartalom túlnyomó része a gép számára közvetlenül nem értelmezhető.

A szerző olyan megoldást keresett, amely segítségével mindenki számára megnyílik a lehetőség, hogy szemantikusan annotált dokumentumokat hozzon létre, legalább a dokumentum-típusok egy apró szegletében.

Ehhez elsősorban a már létező technológiák megismerésére volt szükség. Ide tartozik a tartalom beviteli módjainak vizsgálata, a tartalom tárolás lehetséges módjainak áttekintése, a szemantikus adatok tárolási megoldásainak feltérképezése illetve a szemantikusan annotált tartalom létrehozására képes eszközök megismerése .

Az elmúlt évtizedben kifejlesztett annotáló eszközök száma igen nagy. Ezek egyik összehasonlítási szempontja a felhasználók megcélzott csoportja. Ezen spektrum egyik végén a tudásmenedzsment szakértőknek szánt eszközök állnak, pl.: Protégé, TopBraid , a másik végén az egyszerűbb, néhány óra tanulás után már használható eszközök találhatók, mint például a biológusoknak szánt COHSE. A szerző által létrehozott keretrendszer, a Docuphet célja az, hogy még ennél is kisebb küszöböt kelljen a felhasználónak átlépnie. A koncepció szerint a háttérben működő technológia teljesen el van rejtve, a felhasználó csupán szöveges --- lehetőleg releváns --- állításokat erősít meg vagy vet el.

Annak érdekében, hogy a probléma fogalmilag konzisztens módon kezelhető legyen, az információkeret illetve az annotációs javaslat fogalma került definiálásra. Az információkeret fogalma az ún. eseménykeret koncepció általánosítása, illetve RDF nyelvre átalakítása révén jött létre. Az eredmény --- voltaképp nem meglepő módon --- nem sokban tér el például a SPARQL nyelvben használt RDF Triple Pattern fogalmától. Az információkeret-jelöltek kinyerését egy hatékony mondathatár-felismerő egy névelem felismerő és több, különféle elven működő, az előbbiek kimenetét felhasználó információkeret-felismerő komponens valósítja meg.

A tartalom bevitelele egy kényelmes, AJAX technológián alapuló What You See is What You Need szerkesztő segítségével történik.

A Docuphetben ---sok más eszköztől eltérően--- nem lehetséges egy időben annotálni a szöveget és építeni az annotációkhoz tartozó ontológiát. Ebből következik, hogy a docuphet alkalmazások csak előre definiált szerep- és tulajdonsághármasokkal tudnak dolgozni, ami meglehetősen rugalmatlan megoldás. De ugyanez a tulajdonság teszi lehetővé azt is, hogy a rendszer szövegesen fogalmazza meg a kérdéseit, illetve hogy a kérdések megválaszolásához semmilyen szemantikus technológia ismerete nem szükséges, és általában, a felhasználónak sosem kell explicit módon bevinnie a szemantikus információt.

A fentiek alapján kijelenthető, hogy a Docuphet akkor igazán hasznos, ha a szöveg várható témája előzetesen ismert. Ezen alkalmazások --- bár még erősen kísérleti stádiumban vannak --- használata bíztatóan egyszerű.

Ugyanakkor, ha ez a behangolás nem történik meg, egy korlátozott funkcionalitás akkor is elérhető. A docuphet tetszőleges kategóriába tartozó cikkekben előforduló névelemek felismerésére, típusba sorolására képes, illetve egy megfelelően nagy névelem-adatbázis esetén egyértelműsítésre is alkalmas.

Az alkalmazások készítése során a javaslatokon alapuló annotációkkal kapcsolatban sok érdekes tapasztalatot gyűjtött a szerző. Ez lehetővé tette néhány általában is megfontolásra érdemes követelmény megfogalmazását.

Jövőbeli terv a szerkesztő funkcióinak bővítése: képek kezelése, ergonomikusabb megjelenés, többféle dokumentum elem.

Ugyancsak tervezett további alkalmazási példák megvalósítása, például gazdasági vagy sporthírek, műszaki termékeket értékelő cikkek, földrajzi helyeket bemutató szövegek annotálása. Minden esetben szükséges a keretrendszer előzetes, az adott feladatnak megfelelő konfigurálása.

A Docuphet projekt nem kíván konkurenciát állítani a nagy adatbázisoknak, mint a Wikipedia vagy az IMDB, éppen ellenkezőleg, egy alternatív szerkesztőt szeretne szolgáltatni ezekhez. Folyamatban van egy konverziós eszköz fejlesztése, ami képes a Docbook/RDFa dokumentumokat egy MediaWikibe feltölteni, az újabban rendelkezésre álló API-n keresztül.