SZAKDOLGOZAT

VANDLIK ÁDÁM

Debrecen
2010

DEBRECENI EGYETEM
INFORMATIKA KAR

KONKORDANCIÁK MEGJELENÍTÉSE JAVA APPLETEKBEN

Témavezető:
DR. HABIL. BODA ISTVÁN
Tanszékvezető egyetemi docens
Főiskolai tanár

Készítette:
VANDLIK ÁDÁM
Programozó matematikus hallgató

Debrecen
2010

Tartalomjegyzék

1. Bevezetés

Bizonyára sokan szembesültek már a napjainkban bennünket körülvevő gigantikus információmennyiséggel. A túl sok információ nemhogy segítene, de inkább hátráltat a világ megértésében, pontosabban a minket érdeklő releváns ismeretek megszerzésében. Összezavarhat és a káosz érzetét keltheti bennünk. Főleg ha mindez egyszerre, zuhatagként ér. Ezért elengedhetetlen ezek szűrése, és szelektálása.

Ezzel a területtel egyebek közt a szövegbányászat foglalkozik. A szövegbányászat alapgondolata az, hogy a ma széles körben szabadon hozzáférhető nagy mennyiségű strukturálatlan szöveges állományok kulcsfontosságú tudásanyagot rejtenek, és ezt meg kell próbálni gépi intelligencia segítségével minél hatékonyabban hasznosítani és felszínre hozni.

A nyelvészeti, könyvtári vagy egyéb elemzéseknél használt információlekérdezés sebessége, minősége (teljessége és pontossága) nagymértékben függ az alkalmazott programok megválasztásától. Ezért körültekintően kell ezeket megtervezni és elkészíteni.

Az adatkinyeréssel és feldolgozással már jóval a számítógépek megjelenése előtt elkezdtek foglalkozni. Régebben elsősorban hagyományos megjelenésű anyagok álltak rendelkezésre, és indexek, katalógusok segítették a megfelelő információ megtalálását. A számítógépek megjelenése után exponenciálisan nőtt — és nő ma is — az információkereső tevékenység volumene, ma már ezzel a kérdéskörrel egyre többen és többen foglalkoznak. Mindez felkeltette az érdeklődésemet a téma iránt.

Az egyik központi probléma a kereséskor használt kulcsszavak megválasztása. Mivel természetes nyelven megfogalmazott kereséseknél a szavak számos jelentése és jelentésárnyalata, valamint a szinonimák és rokon értelmű szavak nagy száma jelentősen megnehezíti a megfelelő kulcsszavak kiválasztását, nagy gyakorlati jelentőséggel bírnak azok az eszközök, amelyek eligazítanak bennünket a szavak szemantikai útvesztőjében. Egy széles körben használt eszköz a szövegek konkordancia szótára, amelynek kialakulását és főbb sajátosságait a következő fejezetben ismertetem. Szakdolgozatomban szeretném bemutatni, hogy ha már tisztában vagyunk a saját igényeinkkel, hogy számunkra mi a releváns

információ — vagyis tulajdonképpen mit is akarunk keresni —, akkor a konkordanciák, és az ezeket interaktívan előállító konkordancia programok kellő segítséget adnak.

A témaválasztáskor részben az motivált, hogy a dolgozat megírása mellett magam is programot készítsek ebben a témában. Nagy fantáziát látok egy ilyen program megírásában. Tehát azt tervezem, hogy nemcsak elméleti területen nézek utána a kérdéskörnek, hanem egy célirányú konkrét konkordancia program megírásával, amelyet később fejleszteni kívánok, én magam is aktív részt vállalok benne.

Ma már nemcsak az egyetemek, könyvtárak (Magyarországon például a DEENK, OSZK, stb.), továbbá a nemzeti és nemzetközi információs központok, hanem a nagyobb cégek[1] és intézmények is egyre nyitottabbak a szövegbányászati eszközök használata iránt, amit mutat az is, hogy nagy külföldi IT cégek is komoly üzletet látnak ebben a technológiában.

Az ipari és kutatási alkalmazási területek nagyon széleskörűek, de emellett gyakran jelentkeznek másféle igények is, például a tanulás, oktatás, nyelvészet területén. Elég, ha itt csak a fordítástechnikai elvárások gyors és rutinszerű megoldására gondolunk.

Szakdolgozatomban egy olyan konkordanciaprogram fejlesztését tűztem ki célul, mely az elvárható alapfunkciókon kívül (például a feldolgozott szöveg vagy szövegek szavainak és szövegkörnyezetüknek a kilistázása) egyéb szolgáltatásokat is nyújt a felhasználók számára. A dolgozat első részében először a konkordanciák történetéről lesz szó, majd főbb felhasználási területeiről, régen és ma. Ezután a konkordanciák lehetséges megjelenítési formáinak bemutatása következik. Legvégül pedig a programom tervezésének és implementálásának főbb lépéseit fogom ismertetni.

[1] Arcanum Kft., EPPA Magyarország Kft., IFUA Horváth & Partners, SAS Hungary, Vázsonyi Inc.

2. A konkordanciáról

A konkordancia leginkább egy szógyakorisági listához hasonlítható, ahol a szöveg fontosabb szavait egymástól különválasztjuk, így a szöveget építőkockáira bontjuk le. A szöveget alkotó szavakról egy általános képet (vázlatot) kapunk, viszont elveszítjük a kontextus jellegű információkat, amelyek a szavak eredeti értelmét, sajátosságait, tartalmát, természetét mutatnák be. A számítógépes nyelvészetben a gyakorisági lista alatt a szavaknak és szócsoportoknak egy olyan rendezett listáját értjük, ahol a szavak mellett feltüntetjük azok előfordulási gyakoriságát (a korpuszban hányszor fordulnak elő).[2] Ha pedig minden egyes szót visszahelyezzük az eredeti környezetébe, a használatának a részletei, és a szavak mondatbeli alakja alaposabban vizsgálható. Mielőtt a konkordanciák egyéb felhasználási területeiről és alkalmazásairól szó esne, szeretnék a konkordancia fogalmáról részletesebben írni.

2.1. A konkordancia fogalma

Konkordancia általános jelentése harmónia, összhang, egyetértés.[3] A konkordancia szóval találkozhatunk például az egészségügyben, ahol az orvos és a páciens közötti egyetértést, megegyezést jelenti.[4] Vagy statisztikai számításnál a Kendall-féle konkordanciamutató (egyetértési mutató) kiszámításánál is előfordul. Hasonló összhang létezik egy adott konkrét szó, kifejezés vagy mondatrész és környezete – vagyis az őt körülvevő nagyobb szövegtörzs.– között, amelyből kiemeltük. A konkordancia **nyelvészeti definíciója** is ehhez kapcsolódik:

*Konkordanciának nevezzük egy szó (a továbbiakban **kulcsszó**) előfordulásait egy adott szövegben vagy adott szempontok szerint kiválasztott szövegekben vagy **szövegkorpuszban**, a szó **szövegkörnyezetével** együtt.*

[2] http://en.wikipedia.org/wiki/Frequency_list (2010.04.26.)
[3] http://www.thefreedictionary.com/concordance (2010.03.05)
[4] http://www.medterms.com/script/main/art.asp?articlekey=38929 (2010.04.05)

A korpuszt alkothatja egy teljes könyv szövege vagy egyéb írás, irodalmi mű, egy költő összes verse, stb. A szövegkörnyezet nagysága konkordanciánként változó lehet, a lényeg az, hogy elegendő információt adjon a konkordancia felhasználóinak a vizsgált kulcsszavak szövegbeli használatáról — a szövegkörnyezet nagyságát értelemszerűen a célnak megfelelő méretűre érdemes választani. A szövegkörnyezet megadható például a szó előtt és után lévő szavak vagy sorok számával, de például vers esetén akár a szót tartalmazó versszak is lehet. A kulcsszavakat a konkordanciák ábécé sorrendbe rendezik, egy kulcsszón belül rendszerint szövegkörnyezetük alapján.[5] A konkordanciákat vagy kinyomtatják, vagy interaktívan, számítógép segítségével jelenítik meg. Újabban szinte kizárólag számítógéppel létrehozott konkordanciákkal találkozunk.

2.2. Klasszikus konkordancia

A konkordanciákat korábban „kézzel”, manuálisan készítették. Mivel egy ilyen konkordancia összeállítása a számítógépek megjelenése előtti korszakban rendkívül költséges és időigényes feladat volt, csak különösen fontos munkákhoz készítették el, mint például a Biblia vagy a Korán. Ennek felel meg a konkordancia **„klasszikus” definíciója**:

Konkordancia (a latin cor, 'szív' és a concors, 'összhang' szóból): a szentírástudomány segédeszköze, a Szentírásban előforduló szavak ábécé-rendbe állított gyűjteménye a lelőhellyel együtt. Az egzegétáknak a Szentírás nyelvének és mondanivalójának tanulmányozásában, szónokoknak témájuk szentírási szövegeinek megkeresésében segít.[6]

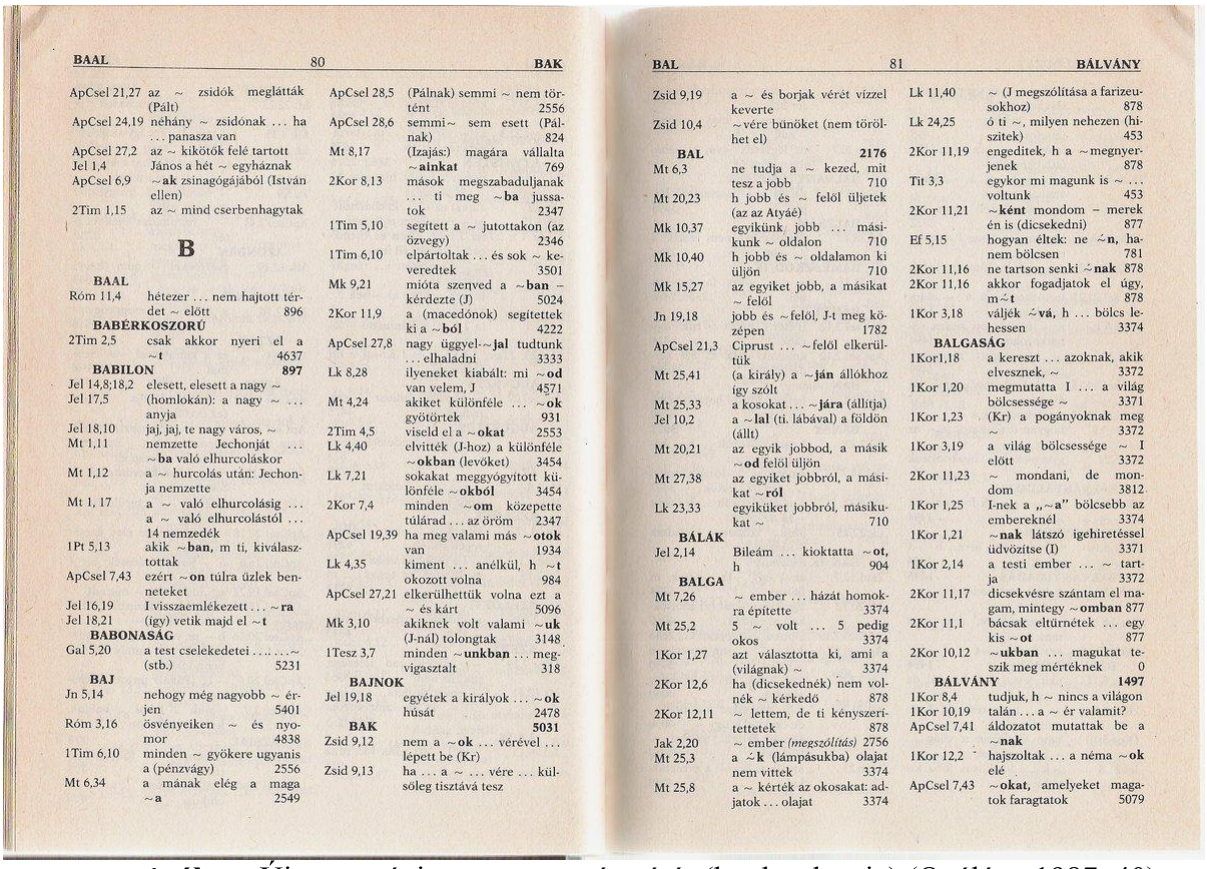
Konkordanciákat először domonkos szerzetesek készítettek a Bibliához: eredeti céljuk az volt, hogy a napi prédikációkhoz minél gyorsabban lehessen megfelelő idézeteket találni.[7] Így ugyanis elég volt csak ebből a rendezett listából kikeresni a kulcsszót, és már rendelkezésre is álltak a legfontosabb szövegrészletek, amelyeket aztán a prédikációk szövegébe lehetett illeszteni.

---

[5] http://hu.wikipedia.org/wiki/Konkordancia (2010.03.05)
[6] http://lexikon.katolikus.hu/K/konkordancia.html (2010.03.05)
[7] http://web.ceu.hu/medstud/manual/MMMhu/frame14.html (2010.04.20)

A konkordanciák manuális elkészítése rendkívül munkaigényes volt. Az első konkordanciát — mai szóhasználattal pontosabb lenne, ha konkordancia-szerű bibliaindexről beszélnénk — a latin Vulgata szövegéből 1230-ban írták meg St. Cher-i Hugó felügyelete alatt, aki a feljegyzések szerint 500 szerzetest kért fel, hogy segítse őt. Érdekesség, hogy idézetek ebben nem voltak, csupán utaltak egyes kulcsszavak esetén azokra a bibliai helyekre (a könyv- és fejezetcímre stb.), amelyekben a kulcsszavak előfordultak. Húsz év múlva ezért három domonkos szerzetes két év alatt elkészítette a bibliai idézetek teljes jegyzékét.[8]

Két további példa bibliai konkordanciákra:



BAAL 80 BAK

ApCsel 21,27 az ~ zsidók meglátták (Pált)
ApCsel 24,19 néhány ~ zsidónak … ha … panasza van
ApCsel 27,2 az ~ kikötők felé tartott
Jel 1,4 János a hét ~ egyháznak
ApCsel 6,9 ~**ak** zsinagógájából (István ellen)
2Tim 1,15 az ~ mind cserbenhagytak

B

BAAL
Róm 11,4 hétezer … nem hajtott térdet ~ előtt 896
BABÉRKOSZORÚ
2Tim 2,5 csak akkor nyeri el a ~**t** 4637
BABILON **897**
Jel 14,8;18,2 elesett, elesett a nagy ~
Jel 17,5 (homlokán): a nagy ~ … anyja
Jel 18,10 jaj, jaj, te nagy város, ~
Mt 1,11 nemzette Jechonját … ~**ba** való elhurcoláskor
Mt 1,12 a ~ hurcolás után: Jechonja nemzette
Mt 1, 17 a ~ való elhurcolásig … a ~ való elhurcolástól … 14 nemzedék
1Pt 5,13 akik ~**ban**, m ti, kiválasztottak
ApCsel 7,43 ezért ~**on** túlra üzlek benneteket
Jel 16,19 I visszaemlékezett … ~**ra**
Jel 18,21 (így) vetik majd el ~**t**
BABONASÁG
Gal 5,20 a test cselekedetei …. …~ (stb.) 5231
BAJ
Jn 5,14 nehogy még nagyobb ~ érjen 5401
Róm 3,16 ösvényeiken ~ és nyomor 4838
1Tim 6,10 minden ~ gyökere ugyanis a (pénzvágy) 2556
Mt 6,34 a mának elég a maga ~**a** 2549
ApCsel 28,5 (Pálnak) semmi ~ nem történt 2556
ApCsel 28,6 semmi~ sem esett (Pálnak) 824
Mt 8,17 (Izajás:) magára vállalta ~**ainkat** 769
2Kor 8,13 mások megszabaduljanak … ti meg ~**ba** jussatok 2347
1Tim 5,10 segített a ~ jutottakon (az özvegy) 2346
1Tim 6,10 elpártoltak … és sok ~ keveredtek 3501
Mk 9,21 mióta szenved a ~**ban** – kérdezte (J) 5024
2Kor 11,9 a (macedónok) segítettek ki a ~**ból** 4222
ApCsel 27,8 nagy üggyel-~**jal** tudtunk … elhaladni 3333
Lk 8,28 ilyeneket kiabált: mi ~**od** van velem, J 4571
Mt 4,24 akiket különféle … ~**ok** gyötörtek 931
2Tim 4,5 viseld el a ~**okat** 2553
Lk 4,40 elvitték (J-hoz) a különféle ~**okban** (levőket) 3454
Lk 7,21 sokakat meggyógyított különféle ~**okból** 3454
2Kor 7,4 minden ~**om** közepette túlárad … az öröm 2347
ApCsel 19,39 ha meg valami más ~**otok** van 1934
Lk 4,35 kiment … anélkül, h ~**t** okozott volna 984
ApCsel 27,21 elkerülhettük volna ezt a ~ és kárt 5096
Mk 3,10 akiknek volt valami ~**uk** (J-nál) tolongtak 3148
1Tesz 3,7 minden ~**unkban** … megvigasztalt 318
BAJNOK
Jel 19,18 egyétek a királyok … ~**ok** húsát 2478
BAK **5031**
Zsid 9,12 nem a ~**ok** … vérével … lépett be (Kr)
Zsid 9,13 ha … a ~ … vére … különséleg tisztává tesz

BAL 81 BÁLVÁNY

Zsid 9,19 a ~ és borjak vérét vízzel keverte
Zsid 10,4 ~vére bünöket (nem töröl-het el)
BAL **2176**
Mt 6,3 ne tudja a ~ kezed, mit tesz a jobb 710
Mt 20,23 h jobb és ~ felől üljetek (az az Atyáé)
Mk 10,37 egyikünk jobb … másikunk ~ oldalon 710
Mk 10,40 h jobb és ~ oldalamon ki üljön 710
Mk 15,27 az egyiket jobb, a másikat ~ felől
Jn 19,18 jobb és ~felől, J-t meg középen 1782
ApCsel 21,3 Ciprust … ~felől elkerültük
Mt 25,41 (a király) a ~**ján** állókhoz így szólt
Mt 25,33 a kosokat … ~**jára** (állítja)
Jel 10,2 a ~**lal** (ti. lábával) a földön (állt)
Mt 20,21 az egyik jobbod, a másik ~**od** felől üljön
Mt 27,38 az egyiket jobbról, a másikat ~**ról**
Lk 23,33 egyikűket jobbról, másikukat ~ 710
BÁLÁK
Jel 2,14 Bileám … kioktatta ~**ot**, h 904
BALGA
Mt 7,26 ~ ember … házát homokra épitette 3374
Mt 25,2 5 ~ volt … 5 pedig okos 3374
1Kor 1,27 azt választotta ki, ami a (világnak) ~ 3374
2Kor 12,6 ha (dicsekednék) nem volnék ~ kérkedő 878
2Kor 12,11 ~ lettem, de ti kényszerítettetek 878
Jak 2,20 ~ ember *(megszólítás)* 2756
Mt 25,3 a ~**k** (lámpásukba) olajat nem vittek 3374
Mt 25,8 a ~ kérték az okosakat: adjatok … olajat 3374
Lk 11,40 ~ (J megszólítása a farizeusokhoz) 878
Lk 24,25 ó ti ~, milyen nehezen (hiszitek) 453
2Kor 11,19 engeditek, h a ~megnyerjenek 878
Tit 3,3 egykor mi magunk is ~ …, voltunk 453
2Kor 11,21 ~**ként** mondom – merek én is (dicsekedni) 877
Ef 5,15 hogyan éltek: ne ~**n**, hanem bölcsen 781
2Kor 11,16 ne tartson senki ~**nak** 878
2Kor 11,16 akkor fogadjatok el úgy, m~**t** 878
1Kor 3,18 váljék ~**vá**, h … bölcs lehessen 3374
BALGASÁG
1Kor1,18 a kereszt … azoknak, akik elvesznek, ~ 3372
1Kor 1,20 megmutatta I … a világ bölcsessége ~ 3371
1Kor 1,23 (Kr) a pogányoknak meg ~ 3372
1Kor 3,19 a világ bölcsessége ~ I előtt 3372
2Kor 11,23 ~ mondani, de mondom 3812
1Kor 1,25 I-nek a „~**a**" bölcsebb az emberekénél 3374
1Kor 1,21 ~**nak** látszó igehirdetéssel üdvözítse (I) 3371
1Kor 2,14 a testi ember … ~ tartja 3372
2Kor 11,17 dicsekvésre szántam el magam, mintegy ~**omban** 877
2Kor 11,1 bácsak eltürnétek … egy kis ~**ot** 877
2Kor 10,12 ~**ukban** … magukat teszik meg mértéknek 0
BÁLVÁNY **1497**
1Kor 8,4 tudjuk, h ~ nincs a világon
1Kor 10,19 talán … a ~ ér valamit?
ApCsel 7,41 áldozatot mutattak be a ~**nak**
1Kor 12,2 hajszoltak … a néma ~**ok** elé
ApCsel 7,43 ~**okat**, amelyeket magatok faragtatok 5079

1. ábra: Újszövetségi szövegmutató szótár (konkordancia) (Opálény 1987, 40)

[8] http://web.ceu.hu/medstud/manual/MMMhu/frame14.html (2010.04.20)

segítőjévé lettek a Lót fiainak. Szela.
Hasonlóképen mint a Lót napjaiban is lett;
De a mely napon kiment Lót Sodomából, tűz és kénkő
Emlékezzetek Lót feleségére!
az igaz Lótot, a ki az

laknak vala ezek: Lótán, Sóbál, Czibhón, Anáh.
Hóri, Hémám, és Lótánnak húga, Timna.
Edóm országában. *22* Lótánnak pedig fiai voltak:
fejedelmek pedig ezek: Lótán fejedelem, Sóbál
Széir fiai: Lótán és Sóbál, Czibhón,
Eczer és Disán. *39* Lótán fiai pedig: Hóri és
pedig: Hóri és Hómám: Lótánnak huga pedig Timna.

ott játszadozik. *16* Lótuszfák alatt heverész, a
Befedezi őt a lótuszfák árnyéka, és

ló körmébe harap, hogy lovagja hanyatt esik.
ő vele szekerek is és lovagok, úgy hogy igen nagy
körmei A futás miatt, lovagjaik futásai miatt.
és ímé a szekerek és lovagok utólérék ótet.
ezer és hétszáz lovagot, és húszezer gyalog
és negyvenezer lovagot, Sobákot is, a
magának szekereket, lovagokat és ötven előtte
és tizenkétezer lovagja.
szekerek városait és a lovagok városait, és
és az ő szekereinek és lovagjainak fejei.
és tizenkétezer lovagja volt néki, a kiket
Salamon szekereket és lovagokat, úgy hogy ezer és
királya is lovon a lovagokkal együtt.
és megveré mind a lovagokat, mind a
Izráel szekerei és lovagjai! És nem látá őt
hegy rakva volt tüzes lovagokkal és szekerekkel
több nép, mint ötven lovag, tíz szekér és tízezer
Izráel szekerei és lovagjai! *15* És monda néki
a szekerek és a lovagok miatt?
szekereket és lovagokat Mésopotámiából,
és tizenkétezer lovagja, a kiket helyheztete
Salamon szekereket és lovagokat; és vala néki
a szekereknek és a lovagoknak minden városait;
s szekereinek és lovagjainak vezérei.
és tizenkétezer lovagjai valának, a kiket
és hatvanezer lovaggal, és
sok szekereik és lovagjaik? Mindazáltal,
kérni sereget és lovagokat oltalmunkra a mi
a seregből és lovagokat. *10* Mikor pedig
kineveti a lovat és lovagját.
lovas csapatot, páros lovagokat, szamaras
csapat jött, páros lovagok, és szólott és
szekeren emberekkel és lovagokkal, és Kir paizst
szekerekkel, és a lovagok nyomulnak a kapu
és a nagyon erős lovagokban; és nem néznek
lovat, ha ugyan tudsz lovagokat ültetni reájok.
a szekerekért és lovagokért.
össze a lovakat és lovagjaikat, és általad
ifjak mindnyájan, lovagok, lovakon ülők.
szépségben öltözőkhöz, lovagokhoz, lovakon ülőkhöz,
lovakkal, szekerekkel, lovagokkal, sereggel és sok
belep téged poruk; a lovagoknak, kerekeknek és
házából lovakat és lovagokat és őszvéreket
seregedet, lovakat és lovagokat, kik mindnyájan
Történő lovag, kardok villogása,
lovat rettegéssel, a lovagját pedig őrültséggel;
3:32 Másnap pedig hagyván a lovagokat tovább menni ő

És oda lovagolt a lovas eléjük, és
az ő atyja, Akháb után lovagoltunk, és az Úr ő
futárok által, a kik lovagolnak a királyi ménes

Dedán a te kalmárod, lovagláshoz való

:9 lova, szekere, meg lovasai és serege
:17 által, szekerei és lovasai által.
:18 az ő szekerei és lovasai által.
:23 lovai, szekerei és lovasai a tenger közepébe.
:26 az ő szekereikre s lovasaikra. *27* És kinyújtá
:28 a szekereket és a lovasokat, a Faraónak minden
:1 mert fenséges ő, lovat lovasával tengerbe vetett.
:19 lovai, szekereivel és lovasaival együtt a
:21 mert fenséges ő, lovat lovasával tengerbe vetett.
5 szekerekkel és lovasokkal a Verestengerig.
11 és szekér vezetőivé és lovasaivá teszi őket, és
:5 szekér és hatezer lovas volt, a nép pedig oly
7 Jórám: Válaszsz egy lovast és küldj eléjök, és
8 És oda lovagolt a lovas eléjük, és monda: Azt
9 elküldött egy másik lovast, a ki hozzájok ment,
23 reájok adni annyi lovast. *24* Hogy verhetnél
8:4 szekeret és hétezer lovast és húszezer
0 És külde leveleket lovas futárok által, a kik
És látott lovas csapatot, páros
És ímé, lovas csapat jött, páros
térek attól. *29* A lovasoknak és a kézíveseknek
és üljetek fel ti lovasok, és legyetek
40 király szekerekkel, lovasokkal és sok hajóval,
által, sem lovak és lovasok által.
száguldoznak, mint a lovasok. *5* Rohannak a
és előtörtetnek az ő lovasai; és az ő lovasai

Hab 1:8 az ő lovasai; és az ő lovasai messziről jőnek,
ApCsel 23:23 Czézáreába, és hetven lovast, és kétszáz
Jel 9:9 viadalra száguldó sok lovas szekerek zúgása.
Jel 9:16 És a lovas seregek száma két
lő
1Sám 20:20 És én három nyilat lövök oldalához, mintha
1Sám 20:20 mintha magamtól czélba lőnék. *21* És ímé utánuk
2Sám 18:14 és Absolonnak szívébe lövé, minthogy még élt a
2Kir 9:24 az ívét, Jórámot hátba lövé a lapoczkák között úgy,
2Kir 13:17 monda Elizeus: Lőjj! És lőtt. Akkor monda:
2Kir 13:17 Elizeus: Lőjj! És lőtt. Akkor monda: Az Úrnak
2Kir 13:18 az Izráel királyának: Lőjj a földbe, és bele lőtt
2Kir 13:18 Lőjj a földbe, és bele lőtt a földbe háromszor;
2Kir 13:19 hatszor kellett volna lőnöd, akkor megverted volna
2Kir 19:32 városba, és nyilat sem lő bele, sem paizs nem
2Krón 35:23 a kézívesek nyilakat lövének Jósiás királyra, és
Ézs 37:33 e városba, nyilat sem lő reá, és nem szállja meg
Jer 50:14 mind ti íjjászok, lőjjetek reá, ne kíméljétek
Zsid 12:20 vagy nyíllal lövettessék le;
lök
Jer 22:28 nincsen? Miért lökték el őt és az ő magvát,
lől
Ez 16:8 mondja az Úr Isten, és lől az enyém.
Ez 16:15 és paráznává lől hired szerint,
lövel
Jób 41:20 Orrlyukaiból gőz lövel elő, mint a forró
Jób 41:21 és a szájából láng lövel elő. *22* Nyakszirtjén
Zsolt 144:6 füstölögjenek! *6* Lövelj villámot és hányd
lövöldözik, lövöldöző
1Móz 49:23 Keserítik, lövöldözik és üldözik a
4Móz 21:30 királynak. *30* De mi lövöldöztük őket, elveszett
2Sám 11:24 Azonban a kőfalról lövöldözének a nyilasok a te
Zsolt 11:2 az igazszívűekre lövöldözzenek.
Zsolt 64:5 nagy hirtelenséggel lövöldözik azt, és nem
Zsolt 64:5 beszédöket, *5* Hogy lövöldözzék titkon az
Péld 26:10 Mint a lövöldöző, a ki mindent
Péld 26:18 halálos szerszámokat lövöldöz, *19* Olyan az, a
Luczius
ApCsel 13:1 és a Czirénei Luczius és Manaen, ki
Róm 16:21 az én munkatársam, és Luczius, Jáson, és Sosipáter
Lúd
1Móz 10:22 Élám, Assur, Arpaksád, Lúd és Arám.
1Krón 1:17 és Assur, Arpaksád, Lúd, Arám, Úcz, Húl, Gether
Ézs 66:19 Tarsisba, Pulba és Ludba, az íjjászokhoz,
Lúdim
1Móz 10:13 Miczáim pedig nemzé: Lúdimot, Anámimot,
1Krón 1:11 Miczráim pedig nemzé Lúdiomot, Anámimot,
lúg
Ézs 1:25 és kiolvasztom mintegy lúggal salakodat, és
Jer 2:22 hajtásává? *22* Még ha lúgban mosakodnál is, vagy
Mal 3:2 és a ruhamosóknak lúgja! *3* És ül mint ötvös
Luhíth
Ézs 15:5 üszőig nyúltanak; mert Luhíth hágóján sírással
Jer 48:5 kiáltanak! *5* Mert a Luhit hágóján siralmat
Lukács
Kol 4:14 Köszönt titeket Lukács, ama szeretett orvos,
2Tim 4:11 Egyedül Lukács van velem. Márkust
Filem 1:24 Aristárkhus, Démás és Lukács, az én munkatársaim.
Lúz
1Móz 28:19 az előtt pedig Lúz vala annak a városnak
1Móz 35:6 Eljuta azért Jákób Lúzba, mely Kanaán földén
1Móz 48:3 Isten megjelenék nékem Lúzban, a Kanaán földén, és
Józs 16:2 És tovamegy Béthelből Lúzba, és átmegy az Arkiták
Józs 18:13 pedig átmegy a határ Luz felé, Lúznak azaz
Józs 18:13 a határ Luz felé, Lúznak azaz Béthelnek háta
Bír 1:23 A város pedig régenten Lúz volt. *24* És látának a
Bír 1:26 és elnevezé azt Lúznak. Ez annak neve mind e

Ly

lyuk
Ézs 2:21 elmenjen a sziklák lyukaiba és a hegyek
Ézs 11:8 a baziliskus lyuka felett terjengeti
Ézs 11:8 a csecsszopó a viperák lyukánál, és a csecstől
Ez 8:7 és látám, és ímé egy lyuk vala a falban.
lyukas
Agg 1:6 a bérlő is lyukas zacskóra bérel.
lyukaszt
Ez 12:5 Szemök láttára lyukaszd át a falat, és azon

M

ma
1Móz 4:14 Ímé elűzöl engem ma e földnek színéről, és a
1Móz 21:26 hallottam, hanem csak ma. *27* Vett azért Ábrahám
1Móz 22:14 Azért mondják ma is: Az Úr hegyén a
1Móz 24:12 Istene, hozd elém még ma, és légy kegyelmes az én
1Móz 24:42 Mikor ma a forráshoz érkezém,
1Móz 30:32 Nyájaidat ma mind végig járom, minden

2. ábra: Konkordancia a Károli Bibliához (Baranyi 2001, 973)

2.3. A konkordancia napjainkban

2.3.1. A konkordanciák a korpusznyelvészetben

A konkordanciák nagy segítséget nyújtanak a természetes nyelvű *korpuszok* tartalmának nyelvészeti elemzésekor. A korpusznyelvészet értelmezése szerint a korpusz „Meghatározott szempontok alapján kiválasztott szövegmennyiség, amelyen a nyelvész vizsgálatát végzi.”(Kugler és Tolcsvai 2000, 132)

A korpusz írott vagy (lejegyzett) beszélt nyelvi adatok gyűjteménye. A szövegeket vagy szövegrészleteket adott szempontok szerint válogatják ki. Egy korpusz nem feltétlenül teljes szövegeket tartalmaz és a szövegekkel együtt rendszerint tartalmazza azok metaadatait (például a szövegek legfontosabb bibliográfiai adatait), és a szövegek a szerkezeti egységeit (bekezdés, mondat, stb.) is. Emellett feltüntetheti a szavak mellett például a szófaji kódjukat is.[9]

A modern korpuszokat elektronikus formában tárolják, és a nyelv vizsgálatának céljából hozzák létre. A konkordanciák például alapját képezhetik a nyelvészeti elemzéseknél használt szóstatisztikai eljárásoknak.

A korpusz nem pusztán szövegek véletlen halmaza, hanem tudatosan megtervezett gyűjtemény, amelynek összeállításakor az ezen elvégezni kívánt nyelvi elemzést tartjuk szem előtt. Minél jobban körülhatárolható kutatásunk tárgya, annál könnyebben lehet döntéseket hozni a korpusz tartamát illetően. Ha például egyetlen irodalmi művet kívánunk elemezni, akkor a cím megválasztásával a korpusz tartalmát is eldöntöttük. Az egy író vagy alkotó összes műveiből álló korpuszt is viszonylag könnyen elkészíthetjük.

A korpuszok – különösen a korai, kisebb korpuszok – sok esetben nem a teljes szöveget tartalmazzák, hanem csak egy töredékét minden szövegnek. Ez például jelentheti azt, hogy cikkek esetében a konklúzió soha nem kerül be a korpuszba vagy csak ritkán. [10]

[9] http://corpus.nytud.hu/mnsz/bevezeto_hun.html (2010.04.20)
[10] (Szirmai 2005, 23)

2.3.2. Szótárkészítés, nyelvtanítás, nyelvtanulás

A konkordanciákat nemcsak a korpusznyelvészetben, hanem más területeken is használják; lássunk erre néhány példát. A konkordanciák a szótárkészítésben, a nyelvtanulásban, valamint a fordításban egyaránt nagy segítséget jelentenek. Többek között segítik a szavak megértését és különféle szövegkörnyezetekben való értelmezését.

Úgy Magyarországon, akárcsak külföldön nagy múltja van az olyan törekvéseknek, hogy számítógéppel szótárakat szerkesszenek. Többek között a gyorsabb elkészülési idő, könnyebb javítás, átdolgozás és pontosabb egységesebb tartalom miatt. Kezdetben a maihoz viszonyítva a lexikográfusok technikailag elmaradottabb, lassúbb eszközökkel és kevés tapasztalattal kezdték el az ilyen munkákat. Egy-egy nagyobb szótár elkészülése több évtizedet is igénybe vett.[11][12]

A mostani korszerű és nagy sebességű számítógépekkel már lényegesen gyorsabban, jobb minőségű eredményeket tudtak elérni. Egynyelvű szótárak készítése ma már szinte elképzelhetetlen hatalmas számítógépes korpuszok és ezeken alapuló konkordanciák nélkül (például Oxford Advanced Learner's Dictionary, Collins COBUILD Dictionary, Longman Dictionary of Contemporary English — hogy csak néhányat említsünk). A szótárkészítés során konkordanciaelemző programok segítik a szótár címszavainak kiválasztását, tehát hogy a szótárban mely szavakra van szükség, és melyekre nincs. Ugyanígy nélkülözhetetlen a homonimák (azonos alakú szavak) elkülönítésében, és példamondatok hozzáadásánál.

A kutatócsoportok és a programfejlesztők hatékony megoldásokat kerestek a közkedvelt terjedelmesebb szótárak és enciklopédiák elektronikus hozzáférhetőségének biztosítására is. Egyrészt internetes online, másrészt a nagygépes és mobil eszközökkel való offline böngészéshez. Így az ezekben történő keresés és az eredmények megjelenítése különösen lényeges. A konkordancia programok pedig széles körű lehetőségeket nyújtanak ezen a téren (például szó-szomszédsági keresés, reguláris kifejezések használata). Ezáltal pontosabban megadhatjuk mire is keresünk, valamint a kontextus információk miatt gazdagabb információkhoz jutunk a nyelv vagy téma tanulmányozása során. Egyszerre több szótárban

[11] http://www.library.utoronto.ca/utel/ret/cawdrey/cawdrey0.html (2010.04.20)
[12] http://www.magyarszo.org/fex.page:c72da289-aafe-8f0f-7d8f-288b8dbcf881.settoxhtml (2010.04.20)

Akárcsak a szótárkészítéseknél, ennek a módszernek és technikának nagy jelentősége van a tanulásban, főleg nyelvek elsajátításában és a nyelvoktatás területén is. A konkordanciák segíthetik a tanulókat a szavak, kifejezések, mondatok és kisebb-nagyobb szövegrészek megírásában, jobb fogalmazási stílus és színvonal elérésében, a szövegalkotási stílus fejlesztésében a további más területeken is.

Ezek az eszközök és körülmények lehetővé teszik a felfedező jellegű tanulást. Oly módon mutatják be a tanulandó nyelvet, amely a tanulók önkreativitását is lehetővé teszik. Saját maguk is tanulhatnak a „tölcséres", vagy „magolós" módszerekkel szemben.

A konkordanciák használatával a tanulók nemcsak a tanár által javasolt mondatszerkesztésre, szövegfordításra szorítkoznak, hanem a bemutatott példák alapján ők maguk, akár önállóan is kialakíthatják az eredeti nyelvnek legjobban megfelelő szórendet, továbbá kereshetnek és beilleszthetnek a fogalmazásba „előre gyártott" szövegrészeket és kifejezéseket, amely egyúttal növeli tanulási kedvüket is. Egy konkordancia program – kombinálva egy odaillő forráskorpusszal – kiváló kiindulási nyersanyag lehet az írásos, de akár verbális gyakorlatok elkészítéséhez.[13]

[13] (Tribble 1990, 35-83) Részletesebben, konkrét alkalmazási módszerekkel.

3. Megjelenítési módszerek

Amikor sikerült a konkordanciaprogramban kiválasztani mely kulcsszavakra vagyunk kíváncsiak, akkor ezt a felhasználó számára előnyös módon és adott szempontok szerint prezentálni kell. A megjelenítési lehetőségek száma olyan sok, hogy itt csak egy-két a konkordanciaprogramokban gyakorta előforduló típust említek meg.

3.1. Statikus és dinamikus interaktív konkordanciák

Régebben csak a statikus, papír alapú, nyomtatott konkordanciák léteztek. Ahogy a 2.1 fejezetben említett Bibliai és egyéb konkordancia szótárak. Ezeknél nem is mindegyiknél másolták ki a környezetet, hanem csupán utaltak az egyes kulcsszavak előfordulási helyeire (könyv, fejezetcím). Előfordulnak még interneten elérhető HTML formátumú webes konkordanciák[14], ahol mindenegyes kulcsszóhoz egy előre legenerált statikus HTML fájl tartozik, amiben az összes előfordulása szerepel szövegkörnyezetével. Ezek néhány szempontból már jobbak, mint a nyomtatott társaik, mivel a keresés bennük jóval gyorsabb, viszont ugyancsak statikusnak tekinthetőek korlátozott megjelenítési lehetőségük miatt.

Véleményem szerint az interaktív konkordanciák sokkal több lehetőséggel rendelkeznek, mint a statikusak, és ezáltal sokkal hasznosabbak, több mindenre felhasználhatóak:

- egyéni szempontok szerint, dinamikusan kereshetünk bennük, akár több kulcsszó együttes előfordulására is (tágabb értelemben ilyenkor akár „többszavas konkordanciáról” is beszélhetünk);
- a keresés gyorsabb és rugalmasabb, hamarabb megtaláljuk az információt;
- a feldolgozott korpusz mérete gyakorlatilag korlátlan, dinamikusan változtatható, akár több korpuszban is kereshetünk egyszerre;
- egy konkordanciaprogram bármilyen (elektronikusan rendelkezésre álló) szövegre, illetve korpuszra használható;

[14] http://www.dundee.ac.uk/english/wics/wics.htm (2010.04.25)

- a találatokat (lényegében a kulcsszavakat szövegkörnyezetükkel együtt) különböző módon jeleníthetjük meg, és egyéb szóstatisztikai adatok is lekérdezhetőek;

- és természetesen a kapott konkordanciákat igény szerint bármikor ki is nyomtathatjuk.

Ez indokolta, hogy szakdolgozatom írása során egy interaktív konkordanciaprogram elkészítését és bemutatását választottam.

3.2. KWIC (Key Word In Context) index formátum

A KWIC (magyarul „Kontextusban lévő kulcsszó") a konkordancia sorok leggyakrabban használatos megjelenítési formátuma. Szinte minden konkordancia program alapértelmezetten támogatja. A találatokat egy listában, egymás alatt felsorolva találjuk meg.[15] [16]

Lényege:

- a keresett elem mindig a képernyő közepén, azonos helyre kerül,

- minden kulcsszó külön sorban, egymás alatt szerepel,

- általában szövegbeli előfordulásuk szerint rendezve,

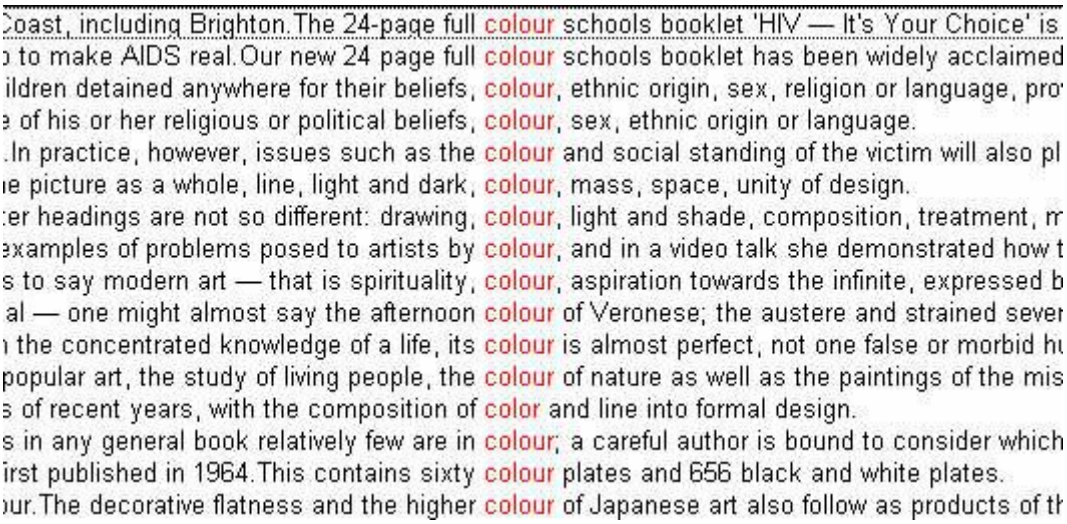
- a szövegkörnyezet többnyire nem teljes mondat, hanem csak annyi karakter amennyi egy sorban a képernyőre (kimenetre) a kulcsszó előtt és után kifér,

- rendszerint a sorok elején az előfordulás pontos helyét is meg szokták adni.

[15] http://en.wikipedia.org/wiki/Key_Word_in_Context (2010.04.25)
[16] (Barnbrook 1996, 69)

Ez a fajta vizuális elrendezés segíti a gyors felismerést. Tekintetünket végig középre irányítva, a szó környezete pillanatok alatt végigvizsgálható, így az esetleges törvényszerűségek is könnyebben észrevehetőek. Interaktív programoknál a szövegkörnyezet nagysága pedig lehet akár tetszőlegesen széles is, mivel az ablak görgethető.

Lássunk erre egy példát:



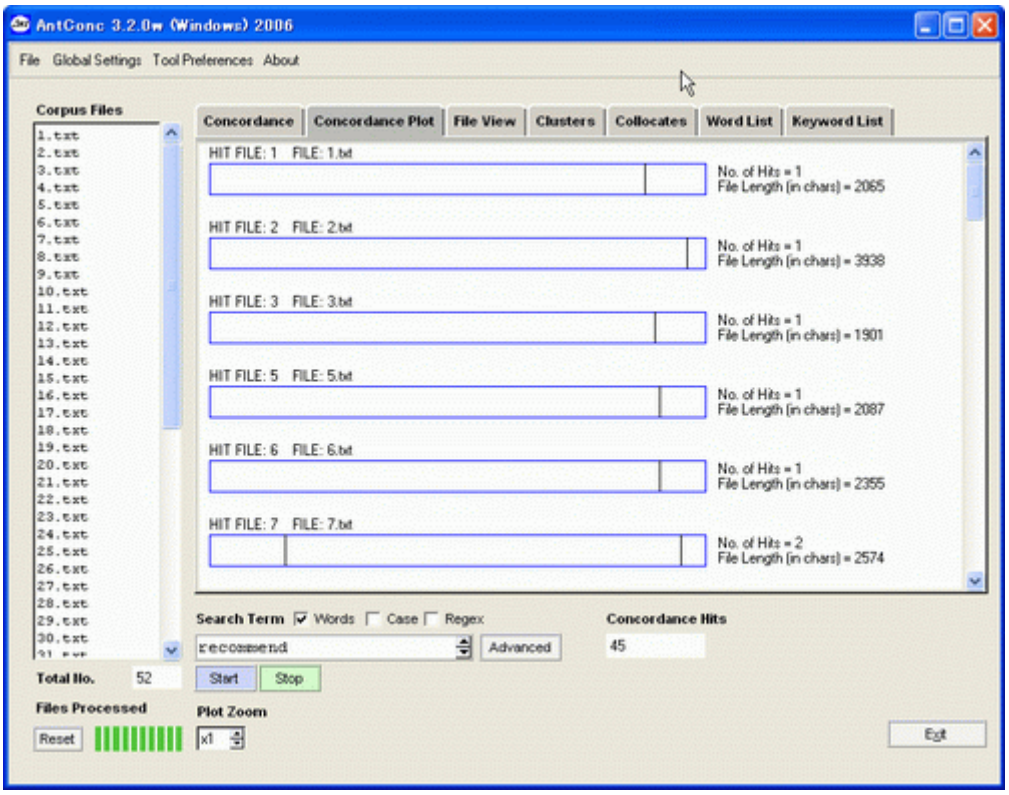
3. ábra KWIC index példa, a kulcsszó középen helyezkedik el

3.3. Konkordanciák rendezése, szó-statisztikai adatok

Ha egy kulcsszónak nagyon sok előfordulása van, nehézkes lehet a konkordanciahalmazban megtalálni az összefüggéseket. Ezért gyakran a programokban lehetőség van ezek valamilyen rendezésére. Leggyakoribb, amikor a kulcsszó konkordanciáit a szövegbeli előfordulásuk helye szerint rendezik. Ezt általában minden program megvalósítja, ez is szokott lenni alapértelmezett. Például akkor előnyös, ha egy szó használatának változásait szeretnénk tanulmányozni, ahogy az a szöveg egyes részeiben, fejezeteiben szerepel. Másik típus, amikor kulcsszó szerint abc sorrendbe rendezzük, jelentősége akkor van, ha nem egy konkrét kulcsszóra, hanem például helyettesítő karaktereket felhasználva egyszerre többre kerestünk. Ugyancsak hasznos lehet a bal vagy jobb oldali környezet szerinti rendezés, vagyis a kulcsszótól balra vagy jobbra eső, egy vagy több szótávolságra lévő szavak szerint. Megfelelően annotált (metaadatokkal ellátott) korpusszal dolgozva pedig rendezni tudunk aszerint, hogy a kulcsszónak éppen melyik

szófajú értelmezését használtuk az adott mondatban. Egyéb lehetőség a szóhossz vagy a szógyakoriság szerinti rendezés. Természetesen nem csak egy, hanem egyszerre több kritériumot is figyelembe véve, ezeket tetszőlegesen kombinálva is rendezhetünk. Tehát például először csoportosítjuk kulcsszó szerint, ezt tovább rendezzük jobbról az első, majd balról az első, jobbról a második… szó szerint. Ezt szokták „zig-zag" rendezésnek is hívni.[17] [18]

Egy interaktív konkordancia programmal általában nemcsak a keresett elem helyét és környezetét írhatjuk ki, hanem gyakran egyéb szóstatisztikai és elemzési adatokat is. A statisztikai adatok alapján pedig különféle *diagramok* és kimutatások rajzolhatók, mint a gyakorisági és a konkordancia diagram, vagy az előfordulások helye grafikusan is jelölhető.

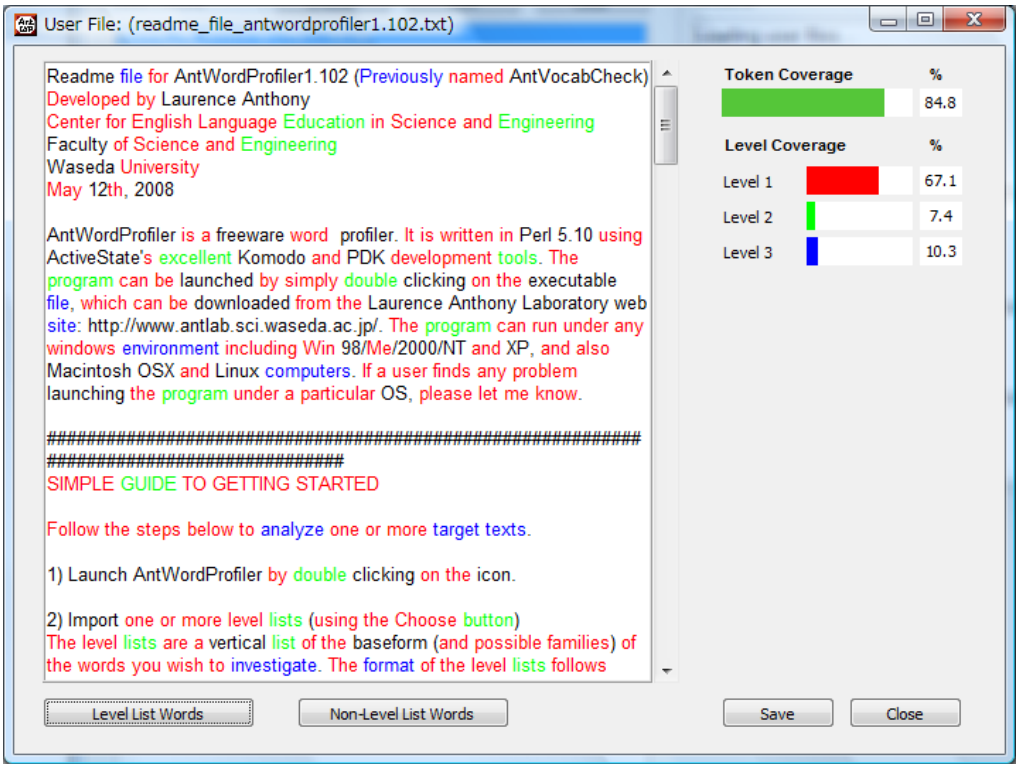


4. ábra: Előfordulások grafikusan jelölhetőek

Annotált korpusznál például különböző színnel jelölhetjük az eltérő szófajokat. Vagy keresést szűkíthetjük egy adott szófajra. Valamit ezen kívül, lemmatizálást és morfológiai

[17] (Oakes 1998, 155)
[18] (Barnbrook 1996, 72)

elemzési módszereket kihasználva, lehetőség nyílik *kollokációk*[19], *szinonimák* és *homonimák* keresésére, valamint az *eredeti szótő* meghatározására is. Ezen kívül több kulcsszó együttes keresésével lehetőség van *szó-szomszédsági* keresésre (proximity search) vagy *több szóból álló kifejezések* keresésére (phrase search) is. Az általam írt programban ezt a két utolsó keresési módszert én is szeretném megvalósítani.



5. ábra: Eltérő szókategóriákat is létrehozhatunk, melynél a ritkábban használt szavak elkülöníthetőek a gyakoriaktól, ahogy az AntWordprofiler[20] programban is látható.

[19] Kollokációk: Állandósult szószerkezetek. Melyek azok a szavak amik gyakran előfordulnak egymás mellett.
[20] http://www.antlab.sci.waseda.ac.jp/antwp_screenshots_win.html (2010.04.27)

4. A Program

Ahogy a bevezetésben már említettem, a szakdolgozatomban a témához egy saját konkordanciaprogram fejlesztésébe kezdtem bele, melynek célja, hogy természetes nyelvű szövegekhez, megadott kulcsszavak alapján konkordanciákat állítson elő és esetleg tegye lehetővé azt is, hogy ezek között különböző szempontok szerint egyéb kereséseket is végezzünk. A továbbiakban rátérnék a konkordanciaprogram részletes ismertetésére.

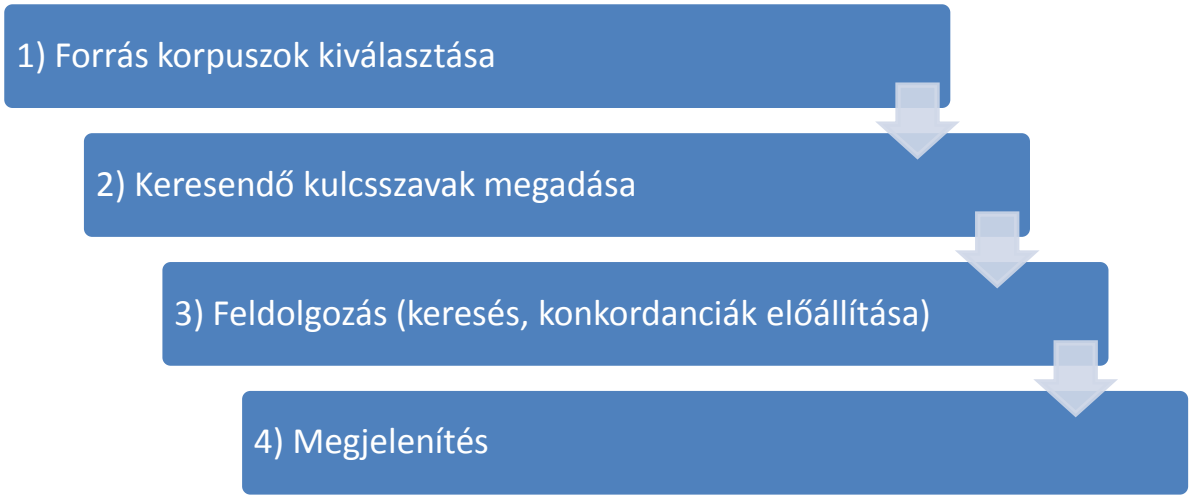
4.1. Feladat specifikáció

Nagyvonalakban a következő elvárásokat fogalmaztuk meg a program számára:

- Platform független legyen,
- Hordozható legyen,
- Szerzői jogok biztosítása érdekében lehetőleg ne lehessen teljes egészében az eredeti forrásszöveget vagy korpuszt letölteni, hanem a hozzáférés legyen korlátozva csak a konkordancia által megjelenített szövegrészletre,
- Különféle kulcsszó kiválasztási opciók választhatósága (például szó eleji egyezés vagy teljes szóra illeszkedés alapján),
- A konkordanciagenerálás viszonylag gyors legyen, vagyis nagyobb méretű korpusz esetén se kelljen sokat várni a felhasználónak, lehetőleg a korpusz kulcsszavai alapján egy külső indexfájl készítése javasolt,
- Az interaktív rész legyen felhasználóbarát, grafikus kezelőfelületű, webes, Java Appletes megvalósítással,
- Később teljesen online hozzáférés biztosítása, lokális korpuszon kívül egyéb internetes korpuszok használata.

4.2. Tervezés

A tervezésnél a program által megvalósítandó fő feladatot, vagyis a konkordanciamegjelenítést a következő négy kisebb részre bontottam:



6. ábra: A program feladatai.

1) Legelső lépésként a forrás korpuszt (vagy korpuszokat) választjuk ki, melyen a konkordanciavizsgálatot végezzük.

2) Ezután a keresendő kulcsszó beírása következik, esetlegesen beállíthatunk különböző keresési opciókat (teljes szó vagy szóeleji egyezés). Több kulcsszó keresésére is lehetőség van, erről később lesz szó.

3) Harmadik lépés a korpusz feldolgozása. Vagyis a forrásszövegben előforduló kulcsszó konkordanciáinak előállítása. Programozástechnikailag alapvetően háromféle módszerrel[21] szokták ezt megtenni:

a) A forrásszöveget tartalmazó fájl-t **soronként** olvassuk be a lemezről. Az egyes sorokat beolvasás után azonnal feldolgozzuk, és a kapott eredményt kimenetre írjuk. Ami lehet képernyő, nyomtató, merevlemez stb. Ezt a módszert alkalmazzák az **adatfolyam elvű** konkordancia szoftverek (streaming concordancers). Előnyei: nagyon kevés memóriaigény, egyszerű az implementálás, nincs átmeneti fájl, egy-egy alkalmi keresésnél hasznos. Hátrányai: mivel nincs átmeneti fájl, ezért mindenegyes

[21] (Tribble 1990, 12)

keresésnél újból át kell vizsgálni a teljes korpuszt, ezért terjedelmesebb korpuszoknál és gyakori keresésnél nem ajánlott.

b) Az egész szövegből egyetlenegy művelettel egy metaadatokkal ellátott **indexfájlt** generálunk, ez általában kisebb mint a vizsgálandó szöveg, de gyakran az adatok miatt nagyobb méret is előfordulhat. Ezzel az előfeldolgozó lépéssel lehetőségünk nyílik változatos szöveg-visszanyerési módszereket, valamint szó-statisztikai lekérdezéseket gyorsan és optimálisan végrehajtani. Általában gyorsasága miatt ezt az **indexfájl-alapú** (text-indexer) feldolgozást választják a leggyakrabban a nagyméretű, ritkán változó korpuszoknál.

c) A harmadik lehetőség, hogy indexfájl generálása helyett egy művelettel olvassuk be a **teljes** forrásszöveget a **memóriába**, és mivel itt az adathozzáférés sokszorosa a lemezének, ezért a feldolgozás is közel valósidőben történik. Ezt a módszert használják a **memória-alapú** konkordanciakészítő programok (in-memory concordancers). Ez a módszer értelemszerűen csak kis terjedelmű korpuszoknál alkalmazható, a nagy memóriaigénye miatt.[22]

A programom elvárásainak leginkább a b) változat, vagyis az indexfájl-alapú feldolgozás felel meg. Ezért ezt választottam. Magát az indexelést egy másik, önálló konzolos Java alkalmazással fogom megvalósítani.

4) A memóriában előálló konkordanciákat adott szempontok alapján meg kell jeleníteni. Itt is lehetőség legyen némi kinézeti beállításra, például szövegkörnyezet és maximális szótáv átállítására. Én a KWIC index formátum helyett inkább egy két-paneles megjelenítést választottam, melynél a felső panel tartalmazza a kulcsszó előfordulásokat az alsó pedig a hozzátartozó szövegkörnyezetet. Mely a szerint változik, hogy éppen melyik előfordulást választottuk ki. Egyéb megjelenítési lehetőségekről a 3. fejezetben esett szó bővebben.

[22] pl: Longman Mini-Concordancer

4.2.1. Fejlesztői környezet kiválasztása

A program Java nyelven íródott, web böngészős applet kisalkalmazásként. Így használatához elég csak egy web böngésző és hozzá a megfelelő verziójú Java futtatási környezet. (Java SE Runtime Environment, JRE).

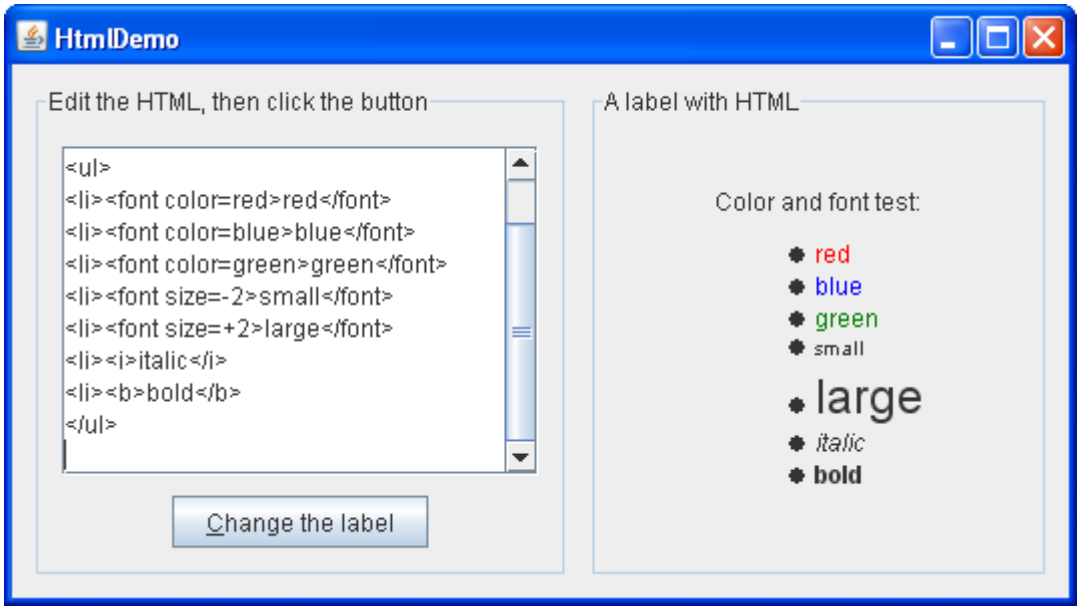
Rendszerkövetelmények:

- Java SE 6.0+ futtatási környezet (jelenlegi verzió JRE 6.0 Update 20),
- Tetszőleges Java által támogatott operációs rendszer (pl. Windows, Linux),
- Tetszőleges böngésző, a megfelelő java-pluginnal (JRE tartalmazza) (pl. Internet Explorer 8.0+, Mozilla Firefox 3.0+, Opera 10.0+, Google Chrome).
- 10MB szabad hely, VGA monitor, 128MB RAM, egér.

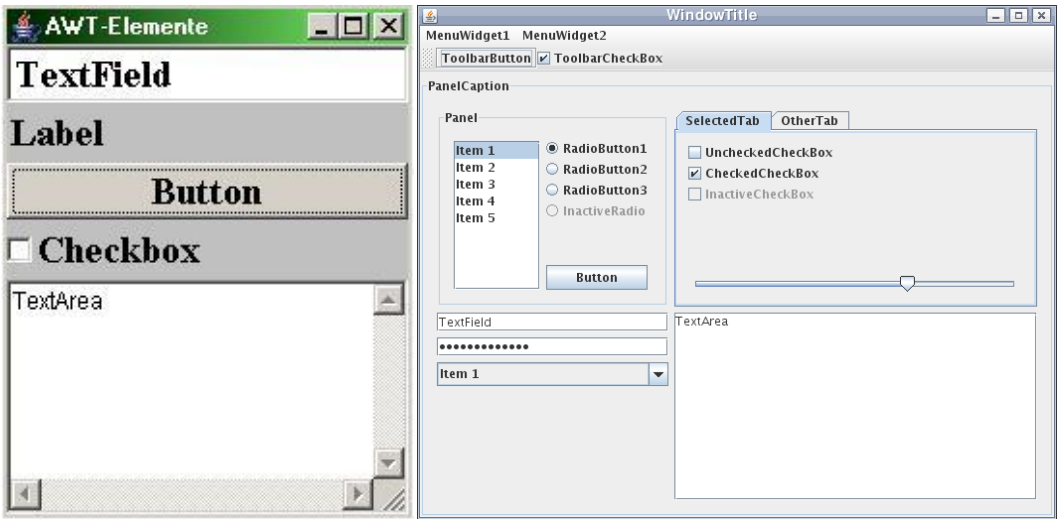
Az alacsony rendszerkövetelmény mellett platformfüggetlensége, hordozhatósága, nem túl bonyolult szintaktikája, tiszta objektum-orientált szemlélete, gazdag beépített osztálykönyvtára, kollekció keretrendszere, valamint webes appletek támogatása miatt is döntöttem a Java nyelv mellett.

Az applet grafikus felületének megírásakor a kezdetlegesebb AWT (Abstract Window Toolkit) helyett a JFC/Swing (Java Foundation Classes/Swing) gazdagabb komponenseit használtam fel. Egyik esztétikai szempont például, hogy a Swing komponenseknél – melyek valamilyen szöveget jelenítenek meg (JLabel, JButton, JMenuItem, stb.) – lehetőség van HTML formázás használatára is. Így egy címkén belül is eltérő színeket, betűtípust használhatunk.[23]

[23] http://java.sun.com/docs/books/tutorial/uiswing/components/html.html (2010.04.22)

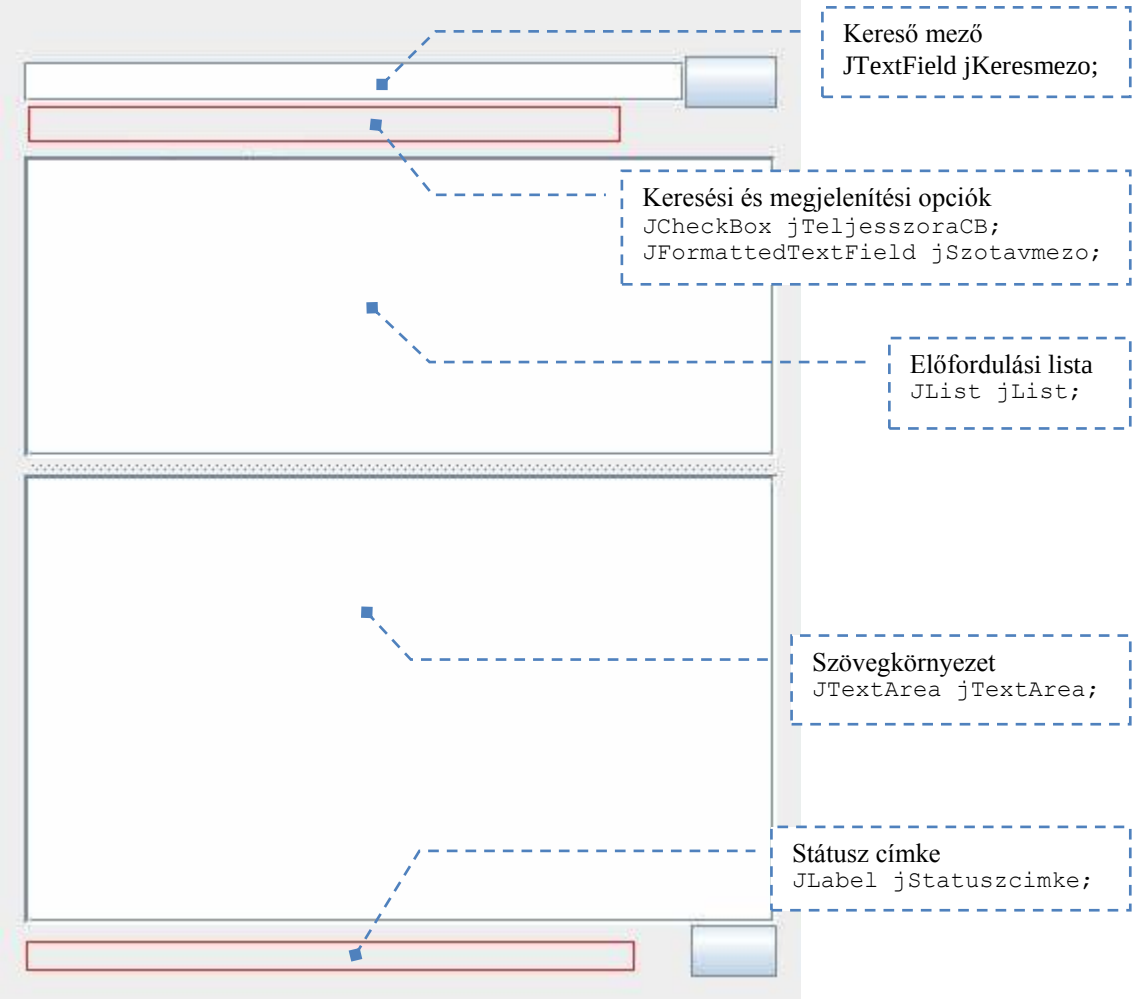


7. ábra: Swing appleteknél pl. JLabel komponens támogatja a HTML formátumot.



8. ábra: AWT és Swing komponensek.

4.2.2. Grafikus felület megtervezése



9. ábra A kezelőfelületet fontosabb komponenseinek elhelyezkedése.

4.3. A program fejlesztésének és működésének bemutatása

Ahogy a tervezésnél már említettem a program két részből áll: egy offline indexelőből (Indexelo.java) és egy online interaktív, konkordanciákat megjelenítő appletből (KonkApplet.java).

Előkészületek: A programot alkotó fájlokat másoljuk fel gépünkre azonos könyvtárba, vagy töltsük fel web-szerverünkre. Tehát a Java osztályokat (KonkApplet*.class, Indexelo.class), az appletet beágyazó HTML fájlt (KonkApplet.html), a korpusz fájlt (arany.txt) és esetleg ha már ezelőtt létrehoztuk, a korpusz indexét (index_arany.txt).

4.3.1. Indexelo konzolos felületű Java alkalmazás

Az indexelő program indítása után a következő képernyőképet kapjuk:

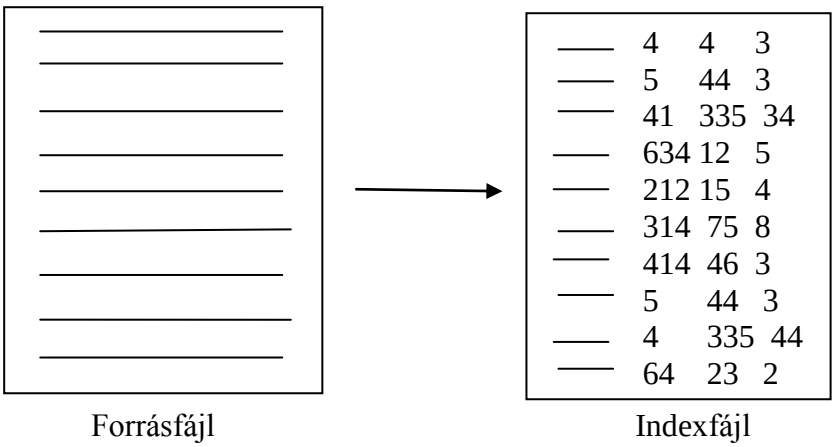
10. ábra: *Indexelo* program kezdőképernyője

Az indexelő egy konzolos felületen futó, Java nyelven írt alkalmazás. Egyetlen argumentumaként egy UTF-8 karakterkódolású forrásszöveget vár. (például arany.txt). Melyből előállítja az indexét, különféle metaadatok hozzáadásával. Kimenete egy hasonlóan UTF-8 karakterkészletű szövegfájl, az index. (az adott esetben index_arany.txt). Az indexelést egy korpuszon csak egyszer kell elvégezni. Segítségével az információ-visszakeresés nagyságrendekkel gyorsabb és kevésbé memóriaigényes művelet. A továbbiakban az interaktív *KonkApplet* már ezzel a legenerált index-fájlal fog dolgozni.

Az indexfájl pontos formátuma a következő:

- A korpuszban található összes szó összes előfordulása külön bejegyzést foglal el az indexfájlban, abc szerint növekvő sorrendbe rendezve, kisbetűvel írva,
- az egyes bejegyzéseket újsor karakter választja el egymástól,
- egy soron belül, a bejegyzéshez tartozó adatokat pedig 1 vagy több szóköz határolja,
- egymás után rendre sorrendben egy bejegyzés a következő adatokat tartalmazza: 1. címszó, 2. ez hányadik szó (a teljes szöveghez viszonyítva), 3. hányadik sor (a teljes szövegben). 4. hányadik karakter (a saját sorában).

A kulcsszó szerint abc sorrendbe rendezett indexfájl azért is hasznos, hogy feldolgozásnál ne kelljen a teljes indexfájlt átvizsgálni, hanem elegendő legyen a lineáris keresés (teljes keresés helyett).



11. ábra: Az indexfájl közelítő vázlata.

Indexfájl létrehozásra példa: java Indexelo arany.txt.

Implementálás: Az *Indexelo* java osztály implementálásánál az indexfájl bejegyzéseit a memóriában egy *String*-eket tartalmazó *ArrayList* kollekcióba gyűjtöm, amit a végén rendezek majd fájlba írok. A korpuszfájlból való olvasást illetve az indexfájlba írást *Scanner* és *PrintWriter* osztályokkal oldottam meg.

```
public class Indexelo {
  private static Scanner forrasfile;
  private static PrintWriter indexfile;
  private static ArrayList<String> index;
  ...
```

12. kódrészlet Felhasznált osztályok.

Az indexelő program *main* metódusában legelőször a konzolos programoknál megszokott argumentum ellenőrzésre kerül sor, ezután az esetleges kezdő BOM (Byte Order Mark) jelző bájtok detektálása és törlése következik.

```
  public static void main(String[] args) {
    try {
      utf8bomtorles(args[0]);
      forrasfile = new Scanner(new File(args[0]),"UTF-8");
    } catch(ArrayIndexOutOfBoundsException e) {
      System.out.println(HASZNALAT);
      return;
    } catch(IOException e) {
      System.out.println("File megnyitasi hiba!");
      System.out.println(HASZNALAT);
      return;
    }
    ...
```

13. kódrészlet: *main()* metódus 1. részlet, ellenőrzés

Melyet az *utf8bomtorles(..)* statikus metódus valósít meg. Erre azért volt szükség, mivel egyes Windowsos programok a fájl elejére írt (0xEF, 0xBB, 0xBF) bájtsorozattal jelzik, hogy a fájl UTF-8 kódolású.[24] De én úgy vettem észre, hogy a Java fájlbeolvasó metódusai nem hagyják ki ezeket, hanem a szöveg részeként tekintik, így ez gondot okozhat a későbbi feldolgozás során.

```
  private static void utf8bomtorles(String filenev) throws IOException {
    byte[] bom = new byte[3];
    File f1 = new File(filenev);
    BufferedInputStream inp = new BufferedInputStream(new
                                        FileInputStream(f1));
    inp.read(bom);
    if (bom[0] == (byte)0xEF && bom[1] == (byte)0xBB
     && bom[2] == (byte)0xBF) {
      File f2 = new File(filenev + "_bak");
      BufferedOutputStream outp = new BufferedOutputStream(new
                                       FileOutputStream(f2));
      int ch;
      while ((ch = inp.read())!=-1) outp.write(ch);
      inp.close();
      outp.close();
      f1.delete();
      f2.renameTo(new File(filenev));
      System.out.println("A bemeneti (UTF-8) file kezdeti 3byteos BOM
jelzoje torolve.");
    }
    else inp.close();
  }
```

14. kódrészlet *utf8bomtorles()* metódus

[24] http://hu.wikipedia.org/wiki/UTF-8 (2010.04.28)

A forrásfájlban a kulcsszavak egymástól való elkülönítését és az indexnek az előfordulási helyekkel kiegészített feltöltését két, egymásba ágyázott for ciklus oldja meg.

```
    ...
    for(int hanyadiksor=1,hanyadikszo=1; forrasfile.hasNextLine();
hanyadiksor++) {
      Scanner aktsor = new Scanner(forrasfile.nextLine());
      aktsor.useDelimiter("[\\p{Blank}\\p{Punct}]+");
      for(; aktsor.hasNext(); hanyadikszo++)
        index.add(String.format("%s %5d %5d %2d",
aktsor.next().toLowerCase(), hanyadikszo, hanyadiksor,
aktsor.match().start()));
    }
    ...
```

15. kódrészlet: *main()* metódus 2. részlet, kulcsszavak elkülönítése

Soronként majd a soron belül szavanként haladva a forrásszövegben, a kulcsszavakat – a Scanner osztály metódusait felhasználva – szóköz karakterek (szóköz, tab, enter) vagy egyéb írásjelek legalább egyszeri előfordulása alapján különítem el. Majd a már fent említett metaadatokkal kiegészítve, megfelelő formátumban bővítem az *index* kollekciót.

Ezután, a forrásfájl végéhez érve, a *java.util.Collections* segédosztály statikus *sort* metódusával az indexet a kulcsszavak alapján abc sorrendbe rendezem. Majd UTF-8 karakterkódolást használva fájlba írom.

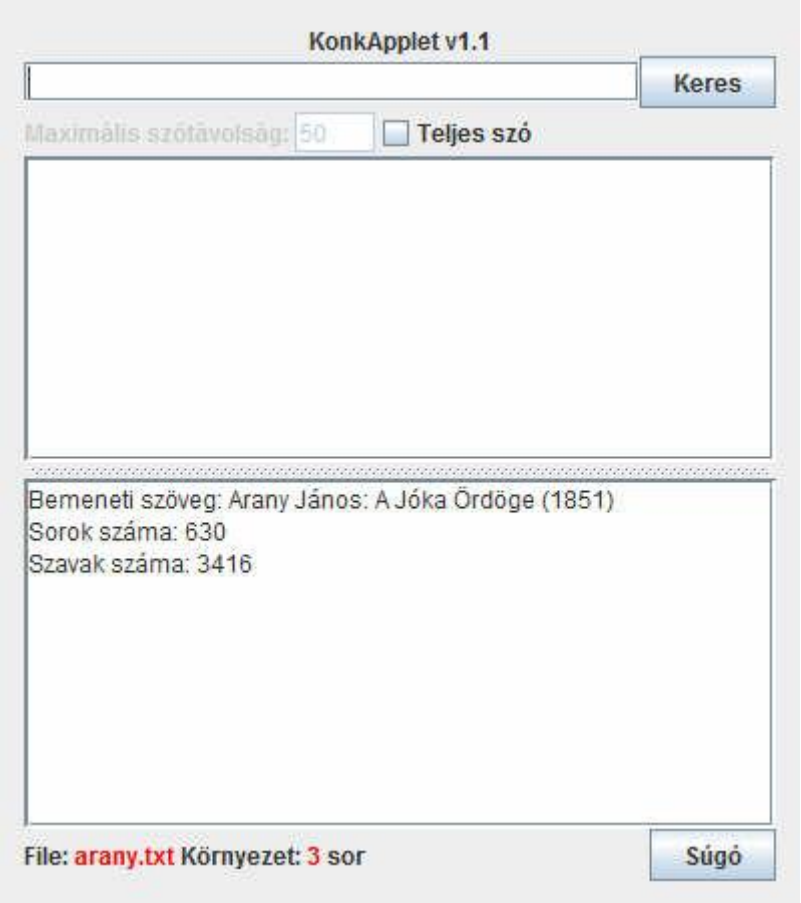
```
    ...
    java.util.Collections.sort(index);

    try {
      indexfile = new PrintWriter("index_" + args[0],"UTF-8");
    } catch(IOException e) {
      System.out.println("File irasi hiba!(index)");
      return;
    }
    for (int i=0; i<index.size()-1; i++)
      indexfile.println(index.get(i));
    indexfile.print(index.get(index.size()-1
    indexfile.close();
    System.out.println("Kesz.");
  }
```

16. kódrészlet: *main()* metódus 3.részlet, Az index fájlba írása.

4.3.2. *KonkApplet* grafikus felületű Java applet

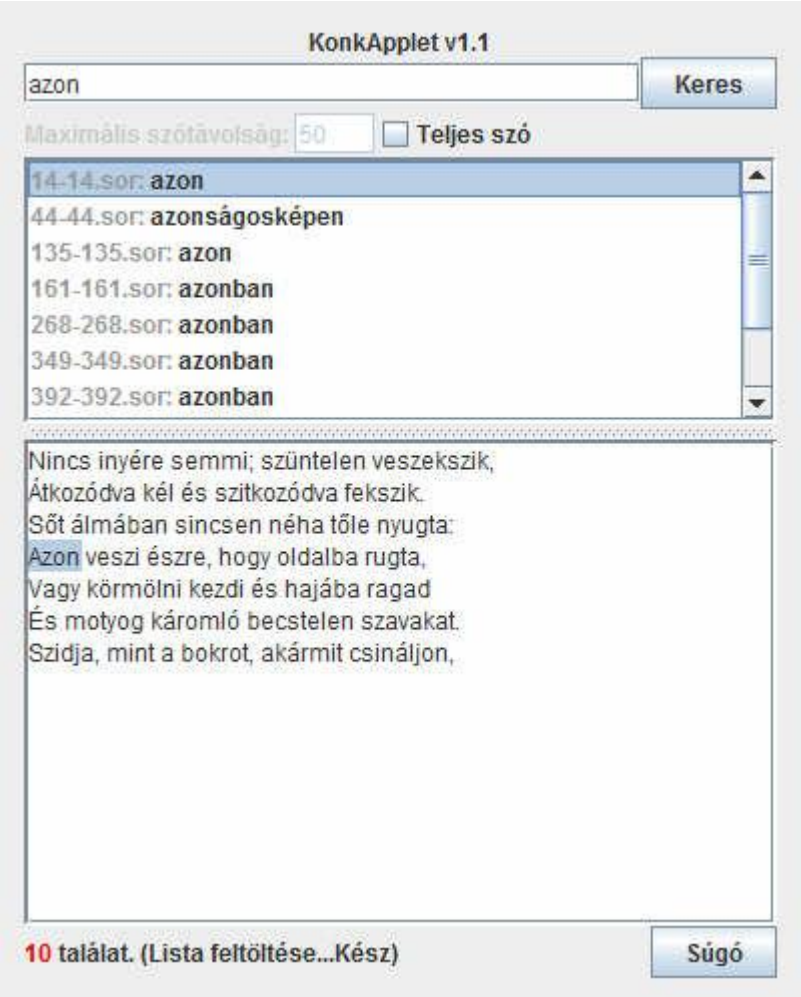
A *KonkApplet* egy Java nyelven írt webes grafikus felületű kisalkalmazás (Java applet). Célja, hogy a forráskorpuszon a hozzá tartozó előzőleg felépített indexfájl felhasználásával, különféle szempontok szerint konkordanciakereséseket végezzen és ezeket a konkordanciákat lehetőleg a felhasználó számára minél egyszerűbb, átláthatóbb formában, interaktívan jelenítse meg.

A konkordancia applet indítása úgy történik, mint bármely más appleté. Létrehozzuk a megfelelő applet-beágyazó kóddal ellátott HTML fájlt, majd böngészőnkkel megnyitjuk azt. Egy minta HTML-fájlt az applettel együtt mellékeltem. (*KonkApplet.html*) Szükséges még az ugyanebben a könyvtárban elhelyezett korpusz és indexfájl megléte is, a program jelenleg alapértelmezetten az *arany.txt*, és *index_arany.txt* fájlokkal dolgozik. Az appletet elindítva a következő képernyőképet kapjuk:



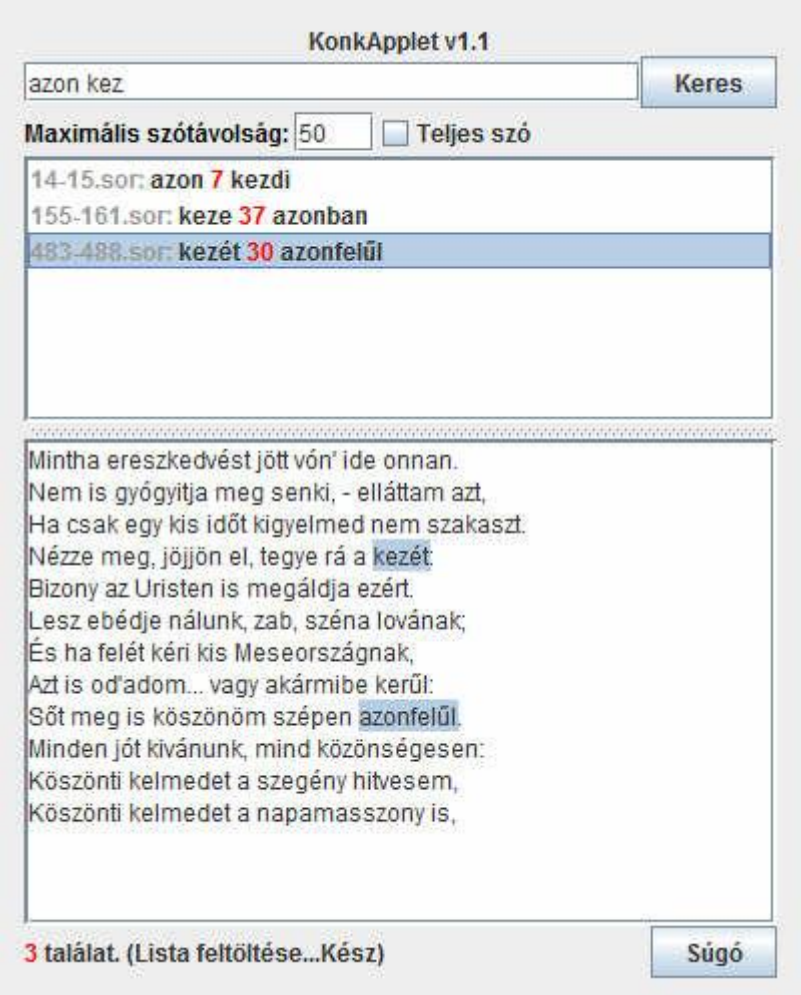
17. példa: *KonkApplet* kezdőképernyő

Az applet betöltődésekor feldolgozza az indexfájlt, külön tárolva a címszóhoz tartozó szövegpozíció információkat, és a memóriába tölti. Így a keresést ezzel sem lassítva. A felső keresőmezőbe írjuk be a keresendő kulcsszót, aminek a konkordanciáit meg szeretnénk jeleníteni, majd nyomjunk a „Keres” gombra. Ekkor az applet az indexben megkezdi a kulcsszóhoz tartozó előfordulások megkeresését. Lineáris keresést használva, alapértelmezetten szóeleji egyezést vizsgál. Kis és nagybetűket figyelmen kívül hagyva. A fenti „Teljes szó” opció bekapcsolásával pedig csak teljes egyezést vesz találatnak. Ezután a kulcsszó összes előfordulását, a sorpozíciót megjelölve és e szerint rendezetten, a középső görgetősávval ellátott listadobozban felsorolja és alul a státusz mezőben jelzi a találatok számát. Egy elemre rákattintva pedig az adott előfordulást szövegkörnyezetével együtt kimásolja a forráskorpuszból, és az alsó szövegdobozban jeleníti meg. Kék színnel kiemelve a kulcsszót, alapértelmezetten előtte és utána 3-3 sor környezettel.



18. példa: Kulcsszó konkordanciái, alapértelmezetten szóeleji egyezéssel.

A 3.3 fejezet végén említett *szó-szomszédsági* keresésre (proximity search) is lehetőség van.[25] Ennek lényege, hogy több kulcsszót adunk meg egyszerre, szóközzel elválasztva egymástól, a sorrend tetszőleges. Ekkor egy-egy találat olyan elfordulást jelöl, melynél az összes kulcsszó szerepel a megadott maximális szótávolságon belül (ami alapértelmezés szerint 50). Vagyis a kulcsszavak között legfeljebb 50 egyéb szó lehet. Így egyfajta tágabb értelemben vett konkordanciát vizsgálhatunk, ahol az egyes kulcsszavak között annál szorosabb kapcsolat feltételezhető, minél kevesebb szó választja őket el egymástól. Ilyenkor a megjelenítés is némileg változik, egy-egy előfordulás már több sor hosszú is lehet, ha a kulcsszavak külön sorban helyezkednek el. Valamint egy piros szám mutatja az adott két kulcsszó közötti tényleges szótávolságot.

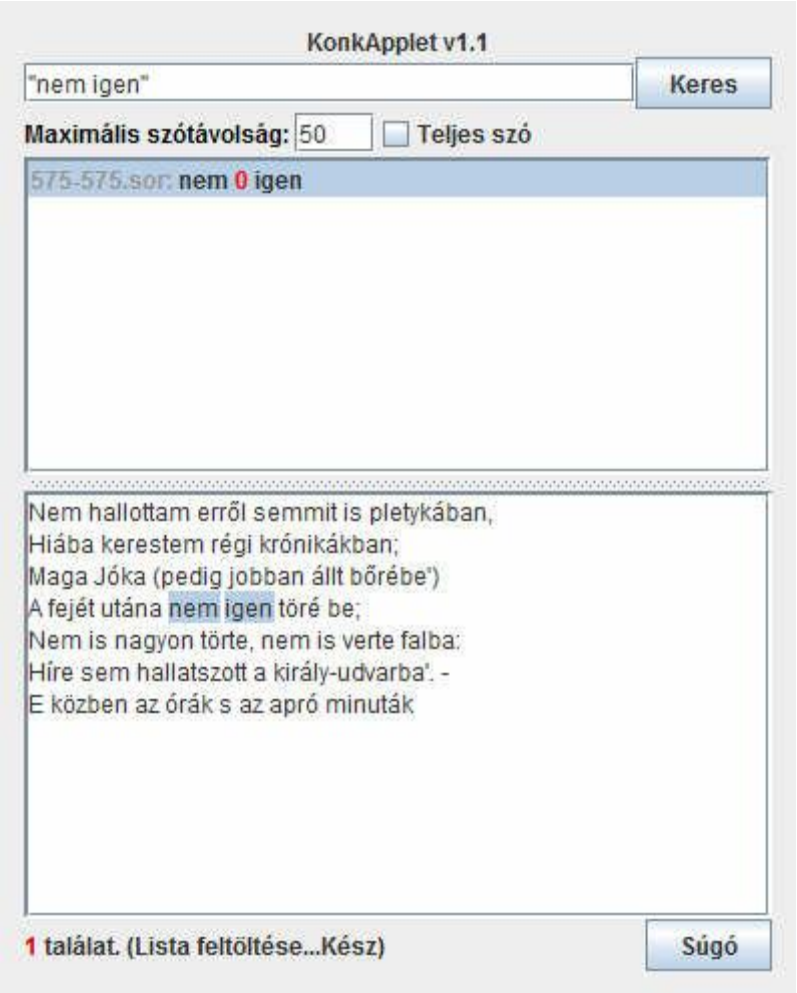


19. példa: Szó-szomszédsági keresés.

[25] http://en.wikipedia.org/wiki/Proximity_search_(text) (2010.04.28)

Egy előfordulásra rákattintva a többsoros előfordulás előtt és után még 3-3 sor szövegkörnyezetet jelenít meg hasonlóan az egy-kulcsszavas kereséshez.

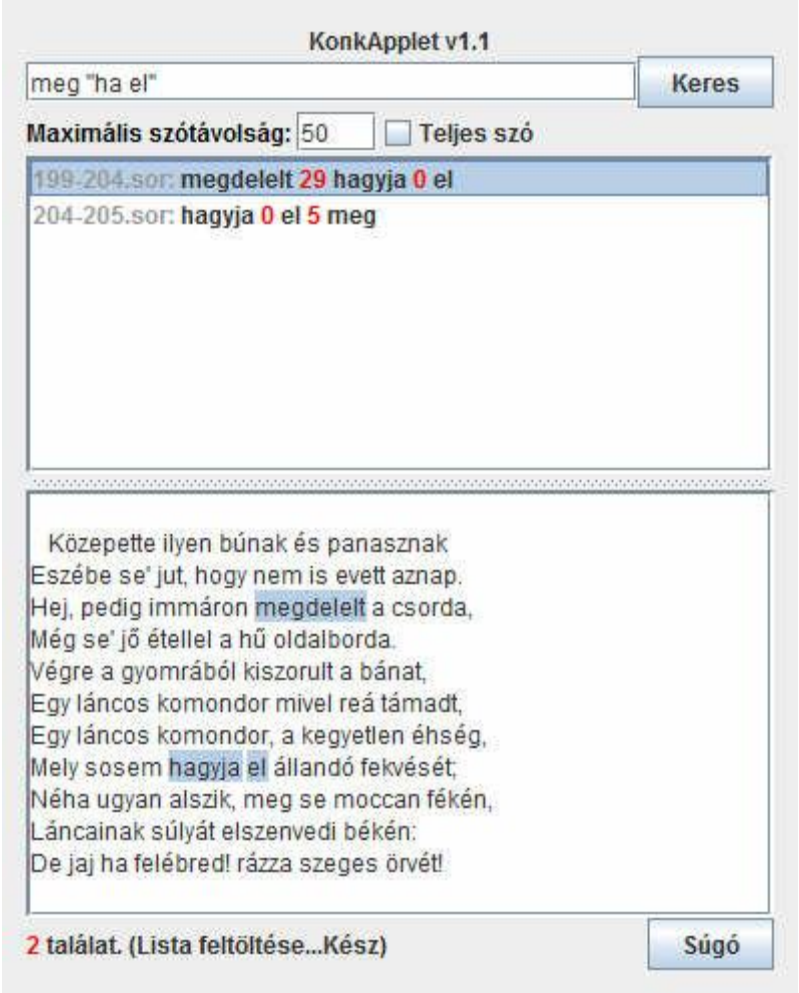
Ha közvetlenül egymás után előforduló szavakra, vagy más néven *több szóból álló kifejezésre* (phrase search) szeretnénk keresni, akkor tegyük a kulcsszavakat idézőjelek közé. Ezt gyakorlatilag úgy valósítja meg a program, mint egy olyan szószomszédsági keresést, amelynél a beírt kulcsszavak sorrendje kötött és köztük a szótávolság nulla.



20. ábra: Közvetlenül egymás után előforduló szavakra történő keresés.

Több kulcsszavas kereséseknél is ugyanúgy működik a szóeleji és teljes egyezés beállítás, mint egyszavasnál, csak itt minden egyes szóra külön-külön értelmezve. Például ha a keresőmezőbe „ha el” szavakra keresünk idézőjelben, szóeleji egyezést állítva, akkor ez illeszkedni fog a „hagyja el” szövegrészre is. Ami elsőre megtévesztő lehet.

Lehetőség van a különböző keresések tetszőleges számú kombinálására is, tehát kereshetünk több szóból álló és egyszavas kifejezés szomszédságára is, természetesen konkordancia megjelenítést használva:



21. ábra: Többszavas keresések kombinálása

Implementálás: A KonkApplet osztály implementálásánál először a 4.2.2. terv szerint megpróbáltam a grafikus felületet kialakítani. Az ablakot, paneleket, gombokat, címkéket és egyéb elemeket kezdetben kizárólag a régebbi AWT eszközrendszerrel valósítottam meg. De később egyre több akadályba ütköztem, melyeket nem tudtam ezekkel a komponensekkel megoldani, és egyéb funkcionalitásbeli hiányosságai miatt is áttértem teljes egészében a – az általam még nem próbált – Swing Appletre (*JApplet*), ahogy a 4.2.1. fejezetben is említettem. Főbb okok, hogy a szövegkörnyezetben a kulcsszó kiemelését, az előfordulás és a státusz sorban a színek használatát, valamint a komponensek automatikus méretezését és rendezését

jóval egyszerűbben meg tudtam oldani. Az áttérés sem volt annyira nehéz, mint gondoltam, mivel a Swing komponensei is az AWT-re épülnek, a *java.awt.Component* leszármazottjai. A komponens nevek is hasonlóak, például az AWT-s *TextArea*, *TextField*, *Button*, *Label*, stb. komponensek megfelelői Swingben rendre *JTextArea*, *JTextField*, *JButton*, *JLabel*.

A KonkApplet grafikus felületét alkotó komponensek, valamint az adatokat tároló adattípusok és saját osztályok:

```
...
private JLabel jPrognevcimke;
private JTextField jKeresmezo;
private JButton jKeresgomb;
private JCheckBox jTeljesszoraCB;
private JFormattedTextField jSzotavmezo;
private JLabel jSzotavcimke;
private JSplitPane jSplitPane;
private JScrollPane jScrollPane1;
private JList jList;
private JScrollPane jScrollPane2;
private JTextArea jTextArea;
private JLabel jStatuszcimke;
private JButton jSugogomb;
...
```

22. kódrészlet: Használt komponensek.

```
...
private ArrayList<IndexElem> index;
private ArrayList<String> forrasfile;
private KeresettSzo[] keresettek;
private Vector<Talalat> talalatok;
...
```

23. kódrészlet: Használt adatszerkezetek

```
...
private class IndexElem implements Comparable<IndexElem>{..}
private class KeresettSzo {..}
private class Talalat {..}
...
```

24. kódrészlet: Létrehozott saját osztályok

Az applet betöltődésekor először az inicializáló rutin, az Applet.init() metódus fut le, életciklusában pontosan egyszer. Ebben szokás a használt változóknak, komponenseknek kezdőértéket adni. Az appletem a következőképpen definiálja ezt felül:

```
public void init() {
    try {
      java.awt.EventQueue.invokeAndWait(new Runnable() {
        public void run() {
          initComponents();
          feltoltindex();
          feltoltforras();
        }
      });
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
```

25. kódrészlet: *init()* metódus

Először az *invokeAndWait()* metódussal az applet fő esemény-küldő szálához (event dispatcher thread) hozzáadom a végrehajtásra szánt saját futtatandó kódomat.

Az inicializálást 3 résztevékenységre lehet osztani, melyet a következő 3 metódus valósít meg:

1) *initCompoments()*: Ebben a metódusban történik az összes megjelenítendő komponens megfelelő kezdőértékekkel való példányosítása és az egyes komponensekhez az eseménykezelők regisztrálása. Ezután felfűzöm a komponenseket a panelre, *GroupLayout* rendezési vezérlőt alkalmazva:

```
  private void initComponents() {
    jPrognevcimke = new JLabel("KonkApplet v1.1");
    jKeresmezo = new JTextField();
    jKeresmezo.addActionListener(this);
    jKeresmezo.addKeyListener(new KeyListener() {..});
    jKeresgomb = new JButton("Keres");
    jKeresgomb.addActionListener(this);
    jTeljesszoraCB = new JCheckBox("Teljes szó");
    jTeljesszoraCB.setFocusable(false);
    jSzotavcimke = new JLabel("Maximális szótávolság: ");
    jSzotavcimke.setForeground(new java.awt.Color(0xCCCCCC));
    jSzotavmezo = new
JFormattedTextField(java.text.NumberFormat.getNumberInstance());
    jSzotavmezo.setValue(new Long(50));
    jSzotavmezo.setEnabled(false);
    jSzotavmezo.addActionListener(this);
    jSzotavmezo.addPropertyChangeListener("value", new
PropertyChangeListener() {..});
    ...
```

26. kódrészlet: *initComponents()* metódus 1. része

```
    ..
    jList = new JList();
    jList.addListSelectionListener(this);
    jList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    jScrollPane1 = new JScrollPane(jList);
    jTextArea = new JTextArea(..);
    jTextArea.setEditable(false);
    jTextArea.setFocusable(false);
    jTextArea.setWrapStyleWord(true);
    jTextArea.setLineWrap(true);
    hilit = jTextArea.getHighlighter();
    painter = new
javax.swing.text.DefaultHighlighter.DefaultHighlightPainter(new
java.awt.Color(0xB8CFE5)); // világoskék
    jScrollPane2 = new JScrollPane(jTextArea);
    jSplitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT, true,
jScrollPane1, jScrollPane2);
    jStatuszcimke = new JLabel(..);
    jSugogomb = new JButton("Súgó");
    jSugogomb.setFocusable(false);
    jSugogomb.addActionListener(new ActionListener() {..});

    GroupLayout layout = new GroupLayout(getContentPane());
    setLayout(layout);
    layout.setAutoCreateContainerGaps(true);
    layout.setHorizontalGroup(layout.createParallelGroup()..);
    layout.setVerticalGroup(layout.createSequentialGroup()..);
  }
```

27. kódrészlet: *initComponents()* metódus 2. része

2) *feltoltindex()*: Itt történik az indexfájl beolvasása az *ArrayList* típusú *index* kollekcióba, melynek elemei *IndexElem* objektumokból állnak.

```
  private void feltoltindex() {
    index = new ArrayList<IndexElem>(5000);
    Scanner file = new Scanner(getClass().getResourceAsStream("index_" +
FILENEV),"UTF-8");
    while (file.hasNextLine()) {
      Scanner aktsor = new Scanner(file.nextLine()).useDelimiter(" +");
      index.add(new IndexElem(aktsor.next(),
                              aktsor.nextInt(),
                              aktsor.nextInt(),
                              aktsor.nextInt()));
    }
    file.close();
  }
```

28. kódrészlet: *feltoltindex()* metódus

Az *IndexElem* osztállyal implementálom a *Comparable* interfészt, és megadom, hogy a rendezés alapja a szövegbeli pozíció legyen, vagyis „hanyadikszo”.

```
  private ArrayList<IndexElem> index;
  private class IndexElem implements Comparable<IndexElem>{
    String str;       // címszó
    int hanyadikszo;  // a teljes szövegben, rendezés alapja
    int hanyadiksor;  // a teljes szövegben
    int hanyadikchar_soronbelul; // soron belül
    IndexElem(String str, int hanyadikszo, int hanyadiksor, int
hanyadikchar_soronbelul) {
      this.str = new String(str);
      this.hanyadikszo = hanyadikszo;
      this.hanyadiksor = hanyadiksor;
      this.hanyadikchar_soronbelul = hanyadikchar_soronbelul;
    }
    public int compareTo(IndexElem o) {
      return hanyadikszo-o.hanyadikszo; // szótávolság számolásnál -1 et
ki kell vonni (talalatok feltöltésénél és Talalat kiírásnál)
    }
    public String toString() {  // indexelem kiírásakor (listadobozba
íráskor)
      return str;
    }
  }
```

29. kódrészlet: *IndexElem* osztály

3) *feltoltforras()*: Az UTF8 kódolású forrásszöveg beolvasása soronként történik, egy *String*-eket tartalmazó *ArrayList* típusú *forrasfile* kollekcióba.

```
  private void feltoltforras() {
    forrasfile = new ArrayList<String>(1000);
    Scanner file = new
Scanner(getClass().getResourceAsStream(FILENEV),"UTF-8");
    while (file.hasNextLine()) forrasfile.add(file.nextLine());
    file.close();
  }
```

30. kódrészlet: *feltoltforras()* metódus

Az *init*() metódus végrehajtása után az applet vár a felhasználó által végrehajtott eseményekre. Az egyes komponensekhez rendelt eseménykezelők ilyenkor futnak le. Lényeges például a „Keresés” gomb megnyomásánál lefutó eseménykezelő:

```
  public void actionPerformed(ActionEvent e) {
    if (jKeresmezo.getText().trim().equals("")) return;
    jStatuszcimke.setText("Keresés...");
    update(getGraphics());
    feltoltkeresettek(jKeresmezo.getText().toLowerCase());
    feltolttalalatok();
    jStatuszcimke.setText("<html><font color=red>" + talalatok.size() +
"</font> találat. (Lista feltöltése...)</html>");
    jList.setListData(talalatok);
    jList.setSelectedIndex(0);
    jStatuszcimke.setText("<html><font color=red>" + talalatok.size() +
"</font> találat. (Lista feltöltése...Kész)</html>");
  }
```

31. kódrészlet: *jKeresGomb*–hoz rendelt *actionPerformed()* eseménykezelő, mely a megnyomásakor hajtódik végre.

A kereséssgomb megnyomásakor lefutó eseménykezelő legfőbb 3 tevékenysége:

1.) A *feltoltkeresettek()* metódus első feladata, hogy feldolgozza a keresésmezőbe írt szöveget, szétválassza egymástól a szavakat, és a megfelelő adattagokba (*KeresettSzo.str*) helyezze el őket. Második feladata pedig, hogy mindenegyes beírt szóhoz, keresse meg az összes előfordulását az indexben (figyelembe véve teljes és szóeleji egyezés opciókat), és ezekkel az adatokkal bővítse ki az adattagokat (*KeresettSzo.elofordulasok*).

```
  private void feltoltkeresettek(String input) {
    initkeresettek(input);              // keresettek.str
feltöltése
    if (jTeljesszoraCB.isSelected())    // keresettek.elofordulasok
feltöltése (indexbeli összes előfordulásával)
      for (KeresettSzo szo : keresettek) {
        szo.elofordulasok = new ArrayList<IndexElem>(50);
        for (IndexElem indexszo : index) // a szó/kifejezés összevetése
a teljes index-el
          if (indexszo.str.equals(szo.str[0])) {
            if (egymasutankovetkeznek(indexszo,szo)) // többszavas
kifejezésnél csak akkor adjuk hozzá a szót az előforduláshoz ha a többi
szavai is egyeznek
              szo.elofordulasok.add(indexszo);
          }
          else if (indexszo.str.compareTo(szo.str[0])>0) break; //
Lineáris keresés
        Collections.sort(szo.elofordulasok); // végül az előfordulásokat
a szövegbeli helyük alapján növekvő sorrendben rendezem
      }
    ...
  }
```

32. kódrészlet: *feltoltkeresettek() metódus*

A keresett szavak szétválasztásáért és *keresettek* adattagba helyezésért az *initkeresettek()* metódus felelős:

```
private void initkeresettek(String input) {
    // A keresésmező szavakká bontása és ezzel a keresettek tömb str
elemeinek feltöltése.
    // Először a többszavas idézőjeles kifejezések átalakítása: "egy
kettő három" -> egy_kettő_három
    StringBuffer sb = new StringBuffer(20);
    int kezd, veg=-1;
    while ((kezd = input.indexOf('"', veg+1)) != -1) {
      sb.append(input.substring(veg+1,kezd));
      veg = input.indexOf('"', kezd+1);
      sb.append(input.substring(kezd,veg+1).replace(' ',
'_').replace('"', ' '));
    }
    String[] inputszavak =
sb.append(input.substring(veg+1)).toString().trim().split(" +");
    // keresettek[] tömb str elemeinek feltöltése az inputszavak[] -al,
    // de a kifejezéseket sztringtömbként hozzáadva: egy_ketto_harom ->
{egy,ketto,harom}
    keresettek = new KeresettSzo[inputszavak.length];
    for (int i=0; i<inputszavak.length; i++)
      keresettek[i] = new KeresettSzo(inputszavak[i].split("_"));
  }
```

33. kódrészlet: *initkeresettek() metódus*

```
  private KeresettSzo[] keresettek;
  private class KeresettSzo {
    String[] str;                        // a keresett szó(vagy
kifejezésnél több szó) (szókezdet is lehet)
    ArrayList<IndexElem> elofordulasok; // indexbeli összes előfordulása
    KeresettSzo(String[] str) {
      this.str = str;
    }
  }
```

34. kódrészlet: *KeresettSzo* osztály

A *keresettek* tömb tartalmazza a keresésnél beírt szavakat, egy-egy *KeresettSzo* objektumban. Többszavas kifejezést is egy *KeresettSzo* objektum tárol, melynél az *str* tömb több elemű, tartalmazza az idézőjelen belüli többi szót is. Egy *KeresettSzo* objektum

elofordulasok ArrayList kollekciója tartalmazza az adott szó indexbeli előfordulásait, egy-egy *IndexElem* objektumban.

2.) Ha a *feltoltkeresettek()* metódus sikeresen beállította a *keresettek* tömböt, vagyis kitöltötte a *KeresettSzo* objektumokat a hozzájuk tartozó előfordulásokkal, akkor következhet az előfordulásokból a tényleges találatnak számító esetek kiválasztása. Ezt a műveletet a *feltolttalalatok()* metódus végzi. Ha csak egyetlen kulcsszót adtunk meg, akkor a találatok egy az egyben a szó előfordulásai lesznek. Viszont több szónál – szószomszédsági keresésnél – a szótávolság számításhoz össze kell hasonlítani egymással az egyes szavak előfordulásának szövegbeli pozícióját is, melyet a *bejar()* metódus fog megvalósítani.

```
  private void feltolttalalatok() {
    talalatok = new Vector<Talalat>(200);
    IndexElem[] akteloforduLassor = new IndexElem[keresettek.length];
    bejar(0,akteloforduLassor);
  }
```

35. kódrészlet: *feltolttalalatok()* metódus

```
  private Vector<Talalat> talalatok;
  private class Talalat {
    IndexElem[] elemek;
    int sormin;
    int sormax;
    Talalat(IndexElem[] elemek) {
      ..
      Arrays.sort(this.elemek); // a találatszavak rendezése szövegbeli
előfordulás szerint
      ..
    }
    public String toString() {  // talalat kiírásánál (listadobozba)
      StringBuffer sb = new StringBuffer(String.format(
          "<html><font color=#999999>%d-%d.sor:</font> %s", sormin,
sormax, elemek[0])); // szürke
      for (int i=1; i<elemek.length; i++)
        sb.append(String.format(
          " <font color=red>%d</font> %s ",elemek[i].compareTo(elemek[i-
1])-1,elemek[i]));
      return sb.append("</html>").toString();
    }
  }
```

36. kódrészlet: *Talalat* osztály

A megjelenítendő találatokat a *talalatok Vector*-ban tárolom, melynek elemei a ***Talalat*** objektumok. A *Talalat* objektum *elemek* tömbje tartalmazza a találat előfordulás-helyét.

Az *elemek* tömb annyi elemű ahány szóra kerestünk. Ugyanúgy, mint ahogy a *keresettek.str* adattag is. Tehát több szavas, szomszédsági keresés esetén ez többelemű, minden szónak egy előfordulásával. Pontosabban mindenegyes keresett szónak egy olyan előfordulását tartalmazza, melynél a közöttük lévő szótávolság a megadott határon belül van. Megjelenítésnél az egyes találatokhoz tartozó szavakat szövegbeli előfordulásuk szerint rendezetten jelenítem meg. Ezért már a inicializáláskor rendezem az *Arrays.sort(this.elemek)* segéd osztálylyal.

A ***bejar()*** metódus dönti el, hogy mely előfordulások számítanak találatnak:

```
  private void bejar(int i, IndexElem[] vizsgelofordulassor) {
    // i: aktuális vizsgált inputszó (oszlop)
    // aktelofordulássor: a vizsgált előfordulás-sor
    if (i == keresettek.length)
      talalatok.add(new Talalat(aktelofordulássor));
    else
      for (IndexElem elofordultelem : keresettek[i].elofordulasok) {
        aktelofordulássor[i] = elofordultelem;
        if (i==0
         || (aktelofordulássor[i-1].compareTo(aktelofordulássor[i]) != 0
 && Math.abs(aktelofordulássor[i-1].compareTo(aktelofordulássor[i])) <=
(Long)jSzotavmezo.getValue()+1)) // Maximum szótáv: beállítható
          bejar(i+1, aktelofordulássor);
      }
  }
```

37. kódrészlet: *bejar()* rekurzív metódus

Egyszavas keresésnél a kulcsszó összes előfordulását hozzáadja a találatokhoz. Többszavasnál pedig rekurzívan végigjárja az inputszavak előfordulásait az összes lehetséges módon, és azokat az előfordulás-sorokat (minden inputszóból egy-egy előfordulást választva), melyeknél mind a megadott szótáv határon belül esik, hozzáadja a találatokhoz.

A következő (38.) példa táblázatosan szemlélteti, hogy a szomszédsági keresésnél mely előfordulások kerülnek kiválasztásra. A keresett szavak: *egy ketto harom*, alatta pedig az előfordulásaik egy-egy számmal jelölve. Ezek közül most az 50 szótávon belüli előfordulásokat tekintettem találatnak.

```
egy  ketto  harom    ;keresett kulcsszavak (sorrend tetszőleges)
5    1      4        ;előfordulásaik(hányadik szó a szövegben)
51   33     123
     99     301
     512
találatok       rendezve
5,1,4       -> 1,4,5     (ketto harom egy)
5,33,4          4,5,33    (harom egy ketto)
51,1,4          1,4,51    (ketto harom egy)
51,33,4         4,33,51   (harom ketto egy)
51,99,123       51,99,123 (egy ketto harom)
```

38. példa: Az előfőrduláslista táblázatosan, legfeljebb 50 szótávra lévő találatok kiválasztása

3.) Végül a megjelenítés következik. A *jList.setListData(talalatok)* metódus feltölti keresőmező alatti *jList* előfordulás-listadobozt a találatokkal. Ezután a *jList.setSelectedIndex(0)* hatására kijelöli az első listaelemet (ha legalább egy találat volt), így lefut a listaelem-választás regisztrált eseménykezelője:

```
  public void valueChanged(ListSelectionEvent e) {
    Talalat jelolt = (Talalat)jList.getSelectedValue();
    jTextArea.setText("");
    if (jelolt==null) return;

    int sormin = (jelolt.sormin-SORELOTTEUTANA-1 < 0
                ? 0 : jelolt.sormin-SORELOTTEUTANA-1);
    int sormax = (jelolt.sormax+SORELOTTEUTANA-1 > forrasfile.size()-1
                ? forrasfile.size()-1 : jelolt.sormax+SORELOTTEUTANA-1);
    int[] hilit_kezd = new int[jelolt.elemek.length];
    int[] hilit_veg =  new int[jelolt.elemek.length];

    for (int i=sormin; i<=sormax; i++) {
      for (int j=0; j<jelolt.elemek.length; j++)
        if (jelolt.elemek[j].hanyadiksor-1 == i) {
          hilit_kezd[j] = jTextArea.getText().length()
                          +jelolt.elemek[j].hanyadikchar_soronbelul;
          hilit_veg[j] = hilit_kezd[j]+jelolt.elemek[j].str.length()-1;
        }
      jTextArea.append(forrasfile.get(i) + "\n");
    }

    hilit.removeAllHighlights();
    try {
      for (int i=0; i<hilit_kezd.length; i++)
        hilit.addHighlight(hilit_kezd[i], hilit_veg[i]+1, painter);
    } catch (javax.swing.text.BadLocationException ex) {
      ex.printStackTrace();
    }
  }
```

39. kódrészlet: *valueChanged()*, listaelem kiválasztásakor lefutó eseménykezelő, megjelenítve a kiválasztott találat konkordanciáját.

A metódus először kiszámolja a megjelenítendő konkordancia kezdő és végsorának pozícióját(alapértelmezetten 3sor a találat előtt és után), majd soronként a forrásfájlból kiírja a megfelelő sorokat. Ha a kiírás valamely kulcsszó sorához ér, akkor megjegyzi ennek pontos kezdő és végkaraktereit. Végül ezeket a pozíciókat felhasználva, az összes találatba tartozó kulcsszót kék színnel kiemeli a konkordanciából.

4.4. Továbbfejlesztési lehetőségek, ötletek

Forrásszöveg:

- További korpuszok közüli választás, akár felhasználó által feltöltött korpuszokon történő elemzés,
- Annotált korpusz formátumok támogatása.

Kulcsszó kiválasztás:

- Egy szólista listadoboz mely a forrásszöveg összes előforduló szavát tartalmazza, és ebből is történhet a kulcsszókiválasztás a kézi beírás helyett,
- Kézi beíráskor gyorskiegészítés felajánlása a teljes szólista alapján,
- Különféle helyettesítő karakterek használatának lehetősége (*,?,+ wildcard karakterek),
- Reguláris kifejezésekkel (programozás során gyakran használt mintaillesztési formátum) történő keresési opció.

Lemmatizálás: (szóalakok összevonása, hasonlóan a helyesírás ellenőrző programokhoz):

- Szótövező modul,
- Morfológiai elemző modul.

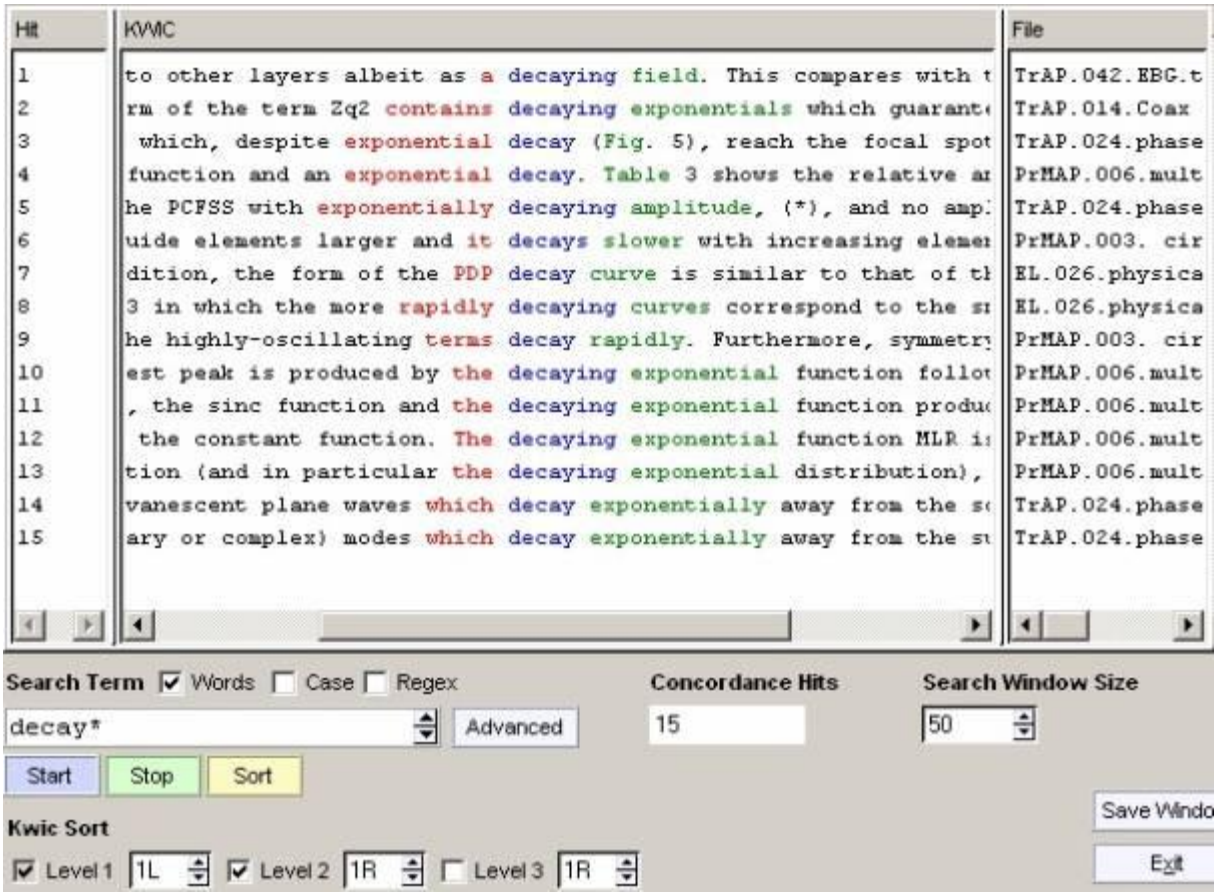
További megjelenítési formák:

- A képernyő kimeneten kívüli egyéb kimenetek választásának lehetősége (nyomtatóra, fájlba),
- KWIC index generálás,
- Webes HTML formátumba való exportálás,
- Kollokációk[26] kilistázása (állandósult szószerkezetek, tehát olyan szavak keresése melyek gyakran előfordulnak egymás mellett),
- Találatok többféle rendezése (forrásbeli előfordulás, abc, bal/jobb kontextus, szógyakoriság, kulcsszóhossz),
- Teljes szöveg statisztikák (karakter, mondat, bekezdések száma, különböző arányszámok (szófajok, szóhosszok)),
- Egyéb ötletek itt találhatóak.[27]

[26] http://www.angoltanszek.hu/TUT/tut.php?tid=1055 (2010.04.25)
[27] http://www.cch.kcl.ac.uk/legacy/teaching/av1000/textanalysis/concord.html (2010.04.25)

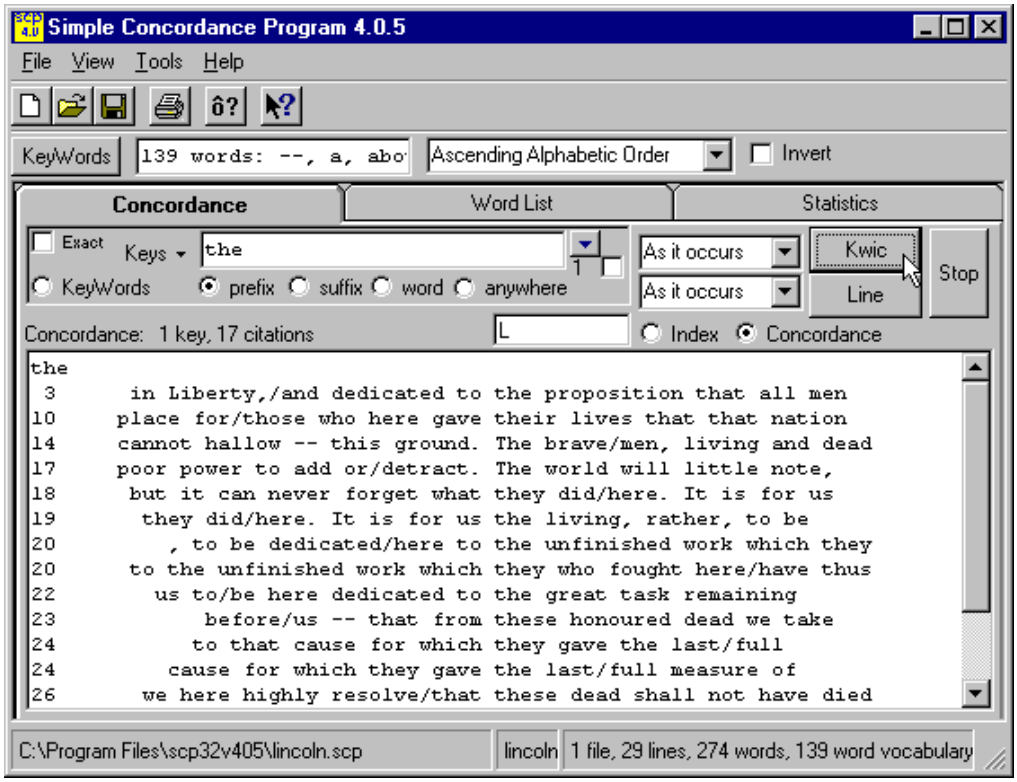
5. Egyéb konkordancia szoftverek és online alkalmazások

A dolgozat írása során több létező konkordancia programmal találkoztam, melyek közül megemlítenék néhányat, lényeges képességeivel.

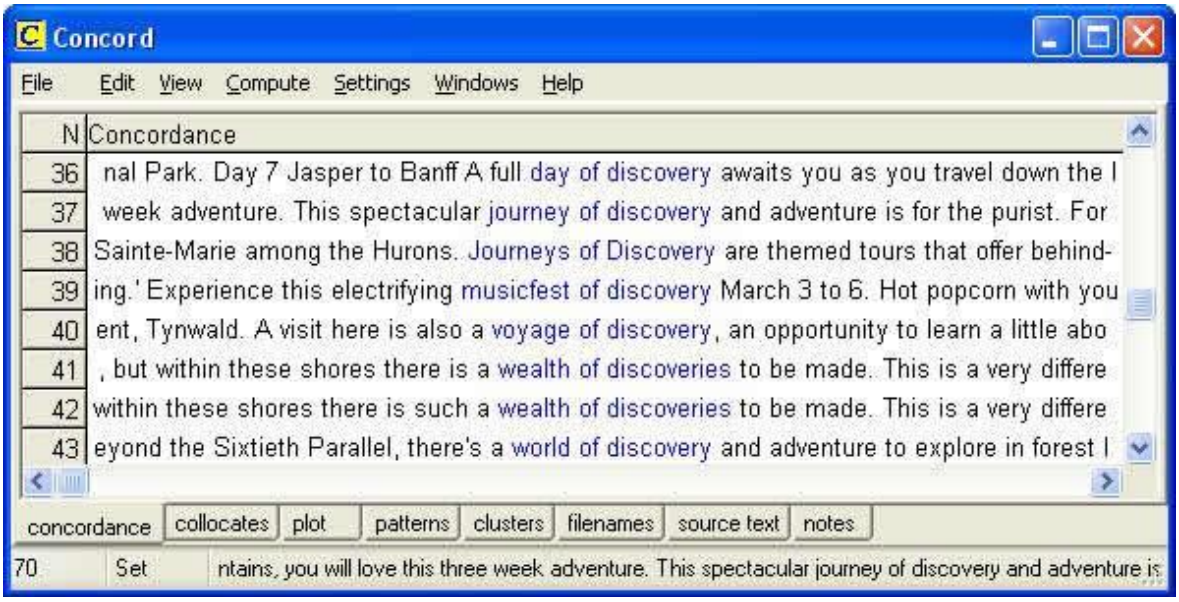


40. ábra: *AntConc 3.2.1* (2007.03.10), ingyenes(freeware) konkordancia program, multiplatform (Win/Mac/Linux), KWIC index formátum, szócsoportok, konordancia-térkép, kollokációk, szógyakorisági lista.[28]

[28] http://www.antlab.sci.waseda.ac.jp/software.html (2010.04.25)

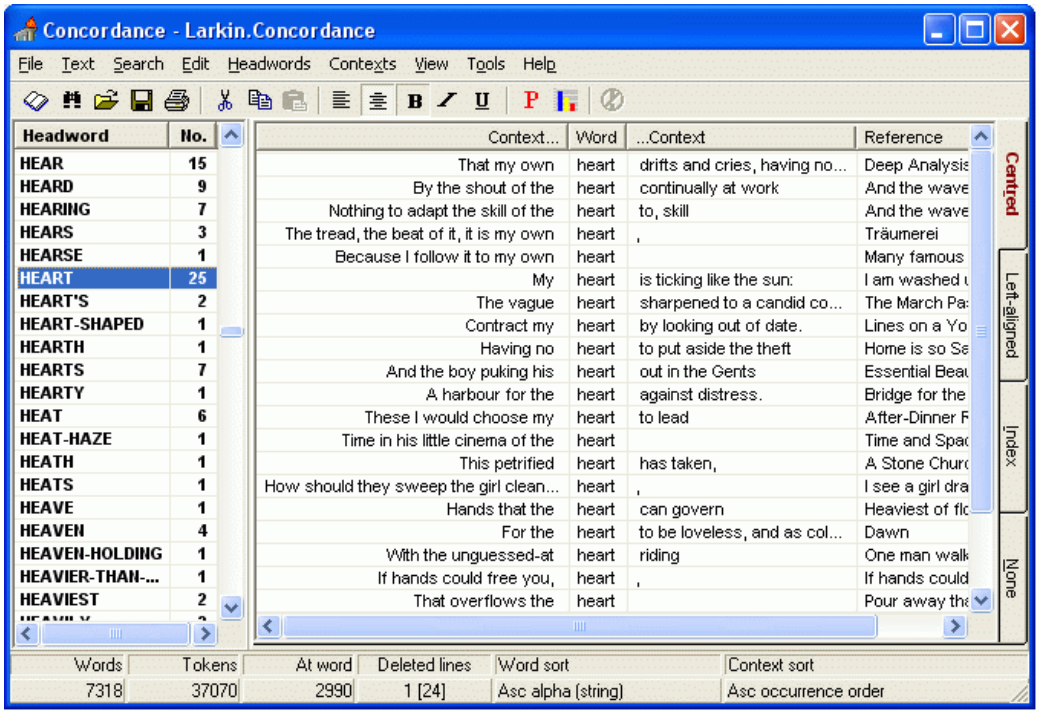


41. ábra: *Simple Concordance Program 4.0.9* (2010.03.26), ingyenes, Win/MAC, többféle rendezés.[29]

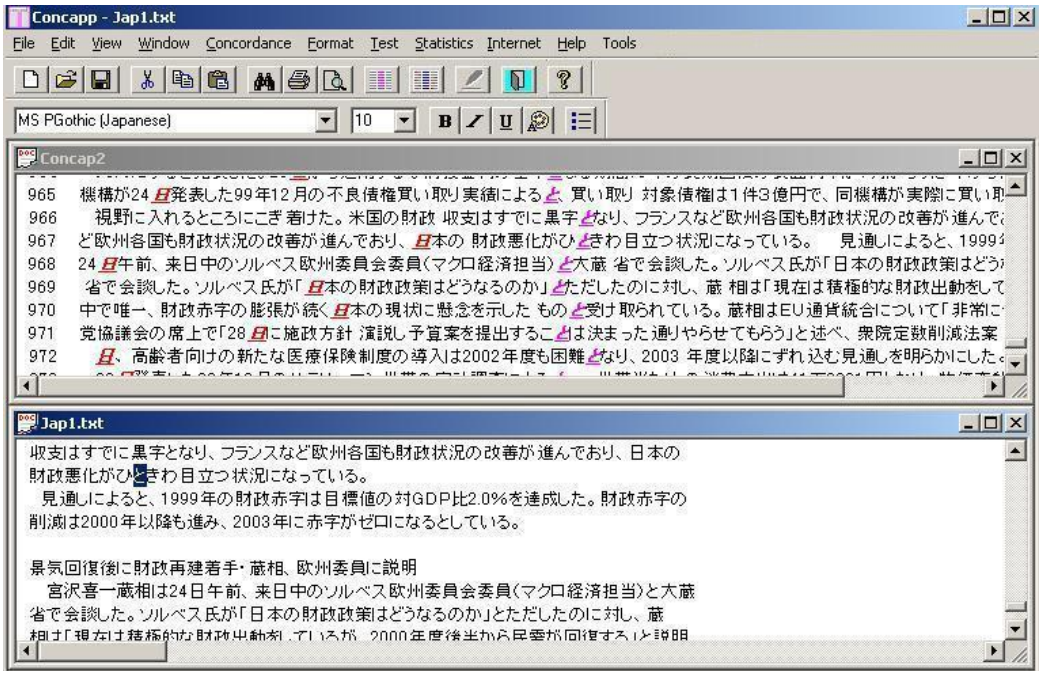


42. ábra *WordSmith Tools Concord 5.0* (2010.04.10), £50, különféle megjelenítési funkciókkal.[30]

[29] http://www.textworld.com/scp/ (2010.04.25)
[30] http://www.lexically.net/wordsmith/ (2010.04.25)

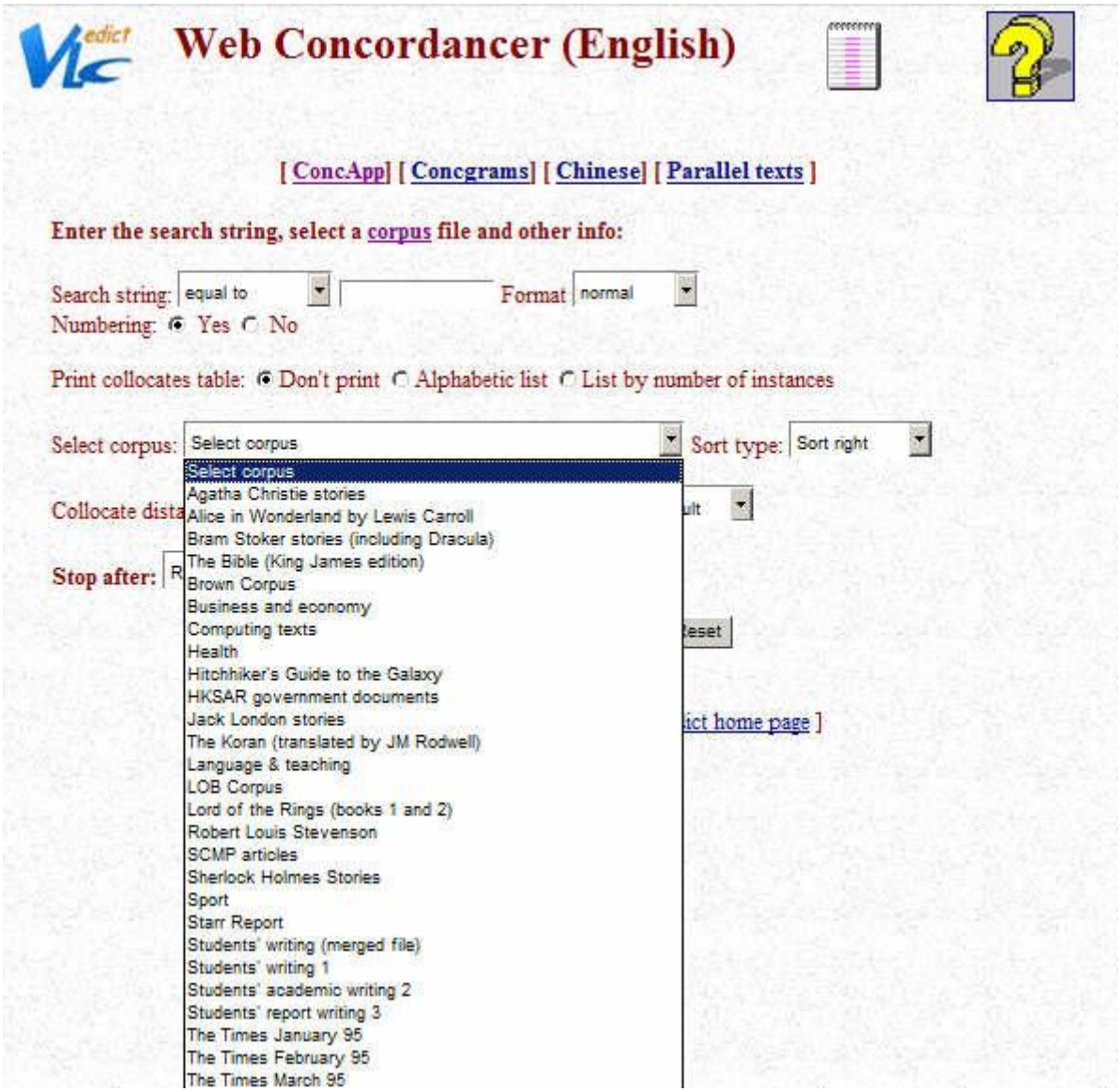


43. ábra: *R.J.C. Watt Concordance 3.30* (2009.07), Windows, $87. Tetszőleges méretű forrás fájl, HTML kimenet, lemmatizálás, statisztikák.[31]



44. ábra A *ConcApp 5 (2008.11.20)*, Windows, $20, megbirkózik a távol-keleti karakterkészletekkel is (japán, kínai, thai, orosz).[32]

[31] http://www.concordancesoftware.co.uk/ (2010.04.25)
[32] http://www.edict.com.hk/PUB/concapp/ (2010.04.25)



45. ábra: *Web Concordancer*, online konkordancia keresés, több beépített korpusszal, KWIC webes megjelenítés.[33]

[33] http://www.edict.com.hk/concordance/ (2010.04.25)

6. Összefoglalás

A dolgozatom fő témája az interaktív konkordanciák és bennük rejlő még máig kihasználatlan lehetőségek feltárása volt. Valamint másik fő célkitűzés az volt, hogy a témában magam is készítsek egy interaktív konkordancia programot, melynek tervezését, fejlesztését és használatát mutatom be.

Az első két fejezetben megpróbáltam az Olvasót közelebb hozni a konkordanciák világához. A történetének rövid áttekintése után azt hiszem kellő számú példát is sikerült felhozni a különféle felhasználási és alkalmazási területéről, mind régi, mai és lehetséges jövőbeli tekintetben is.

Összességében elmondható, hogy ha az olvasó nem is mindig tudta, de ha már használt valamilyen internet kereső oldalt, vagy szövegszerkesztőben speciális keresést akkor szinte biztos, hogy találkozott már konkordanciával. Az internetes keresők a találati listájukban a kulcsszavakat mindig környezetével listázzák ki, és egyre több szövegszerkesztő is felajánlja ezt a lehetőséget. Viszont ezek természetesen nem helyettesítik a kifejezetten erre a feladatra kifejlesztett, célirányú konkordanciaprogramokat. Melyek az egyszerű kulcsszó környezetének megjelenítésén kívül, számos egyéb nagyon hasznos funkciókat nyújtanak. Például az interneten ingyenesen hozzáférhető nagy terjedelmű korpuszokban azonnali keresés, grafikus megjelenítés, számos nyelvészeti funkció és statisztikai kimutatás.

Végül dolgozatban bemutattam az általam fejlesztett interaktív konkordancia program működését és fejlesztését. Munkám során egyre jobban meggyőződtem a konkordanciák hasznosságáról és gyakorlati felhasználási lehetőségeikről. Ezért munkámat a jövőben mindenképpen szeretném tovább folytatni, és a programot a weben bárki számára hozzáférhetővé tenni.

7. Köszönetnyilvánítás

Szeretném megköszönni szakdolgozatom témavezetőjének Dr. Boda István Tanár Úrnak, hogy ötleteivel, tanácsaival, szakmai tudásával támogatta szakdolgozatom elkészülését.

8. Irodalomjegyzék

8.1. Könyvek

Baranyi, József. *Konkordancia a Károli Bibliához.* Budapest: Veritas Kiadó, 2001.

Barnbrook, Geoff. *Language and Computers.* Edinburgh: Edinburgh University Press, 1996.

Kennedy, Graeme. *Introduction to Corpus Linguistics.* New York: Longman, 1998.

Kugler, Nóra, és Nagy Gábor Tolcsvai. *Nyelvi fogalmak kisszótára.* Budapest: Korona Kiadó, 2000.

Oakes, Michael P. *Statistics for Corpus Linguistics.* Edinburgh: Edinburgh University Press, 1998.

Opálény, Mihály. *Újszövetségi szövegmutató szótár (konkordancia).* Budapest: Pázmány Péter Római Katolikus Hittudományi Akadémia, 1987.

Sinclair, John. *Corpus, concordance, collocation.* Oxford: Oxford University press, 1995.

Szirmai, Mónika. *Bevezetés a korpusznyelvészetbe.* Budapest: Tinta könyvkiadó, 2005.

Tribble, Christopher. *Concordances in the classroom.* Harlow: Longman, 1990.

8.2. Internetes források

AntConc

http://www.antlab.sci.waseda.ac.jp/software.html

Letöltve: 2010.04.27.

A Table Alphabeticall of Hard Usual English Words (R. Cawdrey, 1604)

http://www.library.utoronto.ca/utel/ret/cawdrey/cawdrey0.html

Letöltve: 2010.04.25

ConcApp Concordancer

http://www.edict.com.hk/PUB/concapp/

Letöltve: 2010.04.27.

Concordance - Definition of concordance

http://www.thefreedictionary.com/concordance

Letöltve: 2010.03.05

Fundamentals of the digital humanities - Keywords and context

http://www.cch.kcl.ac.uk/legacy/teaching/av1000/textanalysis/gaskin/index.html

Letöltve: 2010.04.20.

Java Foundation Classes (JFC)

http://java.sun.com/products/jfc/reference/faqs/

Letöltve: 2010.04.20.

Key Word in Context - Wikipedia

http://en.wikipedia.org/wiki/Key_Word_in_Context

Letöltve: 2010.04.25

Konkordancia – Wikipédia

http://hu.wikipedia.org/wiki/Konkordancia

Letöltve: 2010.03.05.

Konkordancia – Magyar Katolikus Lexikon

http://lexikon.katolikus.hu/K/konkordancia.html

Letöltve: 2010.03.05.

Magyar Nemzeti Szövegtár

http://corpus.nytud.hu/mnsz/bevezeto_hun.html

Letöltve: 2010.04.20.

Mi mindennel jár a szótárkészítés – Magyar Szó

http://www.magyarszo.org/fex.page:c72da289-aafe-8f0f-7d8f-288b8dbcf881.settoxhtml

Letöltve: 2010.04.25.

R.J.C.Watt Concordance

http://www.concordancesoftware.co.uk/

Letöltve: 2010.04.27.

Searching and Concordancing

http://www.pala.ac.uk/resources/sigs/corpus-style/searching/handbook.html

Letöltve: 2010.04.28.

Simple Concordance Program

http://www.textworld.com/scp/

Letöltve: 2010.04.27.

Szövegbányászat linkek

http://szovegbanyaszat.lap.hu/

Letöltve: 2010.04.25.

The basics of concording

http://www.cch.kcl.ac.uk/legacy/teaching/av1000/textanalysis/concord.html

Letöltve: 2010.04.25.

The Web Concordancer

http://www.edict.com.hk/concordance/

Letöltve: 2010.04.27.

Typology of Books, MMM

http://web.ceu.hu/medstud/manual/MMMhu/frame14.html

Letöltve: 2010.04.20.

Vázsonyi INC. Szövegbányászat cikksorozat

http://www.vazsonyi.hu/szovegbanyaszat/

Letöltve: 2010.04.25.

Web Concordances

http://www.dundee.ac.uk/english/wics/wics.htm

Letöltve: 2010.04.27.

WordSmith Tools

http://www.lexically.net/wordsmith/

Letöltve: 2010.04.27.