UOCConc

Analitzador de Concordances en Python

Representació del Coneixement i Raonament

Memòria del Projecte de Fi de Carrera

Enginyeria en Informàtica

Alumne: **Pau Ubach Royo**
Consultor : **Ramon Masià Fornos**

17 de gener de 2011

Vull dedicar aquest projecte a la Sara, la meva dona, qui m'ha recolzat els darrers anys de la carrera i m'entén perquè sap quant costa seguir endavant quan s'allarga i sembla que no acabarà mai.

Vull dedicar-lo també als meus pares i germans, que sempre han estat amb mi i que ara estan tant lluny.

També vull agrair al consultor, en Ramon Masià, la positivitat a l'hora d'avaluar els lliuraments parcials. En anar-se complicant el projecte, anar requerint més temps, i en veure que no podia seguir el pla proposat, aquests comentaris van servir per animar-me a no deixar-ho córrer.

En darrer lloc, i sense restar-li importància, vull donar les gràcies a Déu, qui m'ha donat ànims durant la carrera i no m'ha permès defallir. Qui fa una història meravellosa i no es cansa de que sovint em queixi per que no ho se veure.

El present Projecte de Final de Carrera realitzat per Pau Ubach Royo està encabit dins de l'àrea de Representació del coneixement i raonament.

Consisteix en el disseny i creació d'un programa per a l'anàlisi de concordances.

L'aplicació resultant es pot utilitzar com a base de futurs projectes, pel que aquesta memòria, a més de documentar el projecte te un objectiu didàctic per tal de facilitar l'aprenentatge a qui l'amplii.

Per aquest motiu l'estructura i formes d'aquesta memòria poden resultar diferents als normals, ja que el que es cerca és introduir al lector en les eines que s'utilitzen per al desenvolupament, així com en l'aplicació generada.

S'adjunta a aquesta memòria l'aplicació resultant amb alguns fitxers de test i un manual de PyQt, per tal de que el lector, pugui arribar a un nivell de coneixement suficient per a poder continuar amb el desenvolupament de l'aplicació en el menor temps possible.

Índex de Contingut

Índex de Figures

1. Introducció

1.1. Justificació del PFC i context en el qual es desenvolupa

El present Projecte de Final de Carrera s'ubica dins de l'àrea de Representació del Coneixement.

El projecte consisteix en la realització d'una aplicació per a l'anàlisi de concordances basant-se en algunes de les funcionalitats que ofereixen programes similars. La base per a la aplicació ha estat, per tant, un programa d'aquest camp: La versió 3.2.1 d'AntConc.

Aquesta aplicació servirà com a base per altres projectes, pel que ha d'estar preparada per a sofrir modificacions i ampliacions.

1.2. Objectius del PFC

L'objectiu del projecte és crear una aplicació per a l'anàlisi de concordances, és a dir, que cerqui expressions en un text o conjunt de textos que compleixin certes condicions mostrant el context on apareixen aquestes. També tindrà altres funcions relacionades.

Amb aquest fi utilitzarem la versió 3.2.1 d'AntConc de Laurence Anthony com a model, i l'aplicació construïda haurà d'emular algunes de les funcionalitats bàsiques d'aquest programa.

Hi ha, però, un objectiu secundari. L'aplicació haurà de servir com a base de futurs projectes, per tant s'ha de tenir en compte futures ampliacions i modificacions. Això vol dir que el codi de l'aplicació ha d'estructurar-se correctament i la documentació ha de permetre una ràpida assimilació dels conceptes per tal de poder continuar amb la feina ràpidament.

Així, aquesta memòria, a més de servir com a documentació del projecte, te un objectiu principalment didàctic per tal de facilitar l'ampliació i modificació de l'aplicació així com la comprensió del funcionament d'aquesta.

Per al desenvolupament de l'aplicació s'utilitzarà Python, i per al disseny i desenvolupament de la part gràfica es farà servir PyQt.

Inicialment es va optar per eliminar les pestanyes Clusters i KeywordList. Posteriorment s'han deshabilitat altres.

1.3. Enfocament i mètode seguit

El fet de que les eines a utilitzar, així com el camp de l'anàlisi de concordances, fossin completament nous per mi, ha afectat en l'enfocament i el mètode a seguir.

Inicialment va ser necessari un estudi de l'eina a emular, així com de totes les funcionalitats que incorpora. Això inclou cerques al respecte i proves de l'eina.

Un cop estudiada aquesta va ser possible elaborar un pla de treball, on es va tenir en compte estructurar aquest conforme a la corba d'aprenentatge que el projecte requeria. Amb aquest motiu es van fer les següents consideracions:

1. Realitzar primer les funcions bàsiques implementades en Python per tal de familiaritzar-se amb el llenguatge.
2. Dissenyar la interfície gràfica amb Qt Designer.
3. Implementar les funcionalitats de la interfície gràfica, incorporar i fer canvis si era necessari en les funcions bàsiques ja definides.
4. Implementar i incorporar l'eina de dibuix per al plot.
5. Implementar altres funcions si fos necessari i incorporar-les a l'eina.

Per a la redefinició de funcions bàsiques en el punt 3, es va decidir generar noves funcions deixant les antigues disponibles per si més endavant eren necessàries. Les funcions que no s'han utilitzat no s'han eliminat de cara a que puguin ser útils en futures remodelacions i ampliacions de l'aplicació.

Es va decidir anar generant la documentació durant totes les fases del projecte per tal de no deixar-ho tot pel final i poder fer un seguiment òptim.

Degut a la simplicitat de l'aplicació i al desconeixement inicial de les eines no es va definir inicialment una arquitectura a seguir, però tot i així PyQt ha facilitat l'estructuració on es separa els models de dades de les funcionalitats. L'arquitectura es veurà més endavant.

1.4. Planificació del projecte

Inicialment es varen marcar les fites detallades a continuació:

- Redacció del pla de treball: Consisteix en la redacció del pla de treball que s'adjunta.
- Acceptació del pla de treball: Implica la revisió del pla de treball i la seva acceptació.
- Redefinició de fites: En cas de observacions o problemes en les fites marcades, s'ha reservat un temps per a redefinir aquestes.
- Redefinicions de disseny: Es reserva un temps per comprovar si un disseny similar al d'AntConc és la millor opció per les millores i canvis que s'haurà de fer al programa en un futur.
- Càrrega de fitxers: Programació i documentació del mòdul encarregat de la càrrega i lectura de fitxers.
- Funcions bàsiques de cerca: Programació i documentació de les funcions bàsiques que utilitzarà el programa per a les cerques.
- Interfície gràfica: Programació i documentació de la interfície gràfica.
- Plot: Programació i documentació del plot per a la pestanya Concordance Plot.
- Funcions avançades: Revisió de les funcions i desenvolupament i documentació, en cas de que sigui necessari de funcionalitats avançades.
- Tests: Disseny i execució dels tests necessaris per a avaluar el correcte funcionament del programa.
- Compilació i ampliació de la documentació: Recopilació de tota la documentació, unificació i ampliació.
- Tancament del projecte.

A més es van marcar les fites externes corresponents a la data de lliurament del projecte, i els lliuraments parcials per a portar un control de que es compleixen els requisits. Aquestes fites tenien data definida i inamovible:

PAC1 – Pla de treball: 25/10/10
PAC2 – Seguiment: 22/11/10
PAC3 – Seguiment: 21/12/10
PAC4 – Lliurament: 17/01/11

La resta de fites eren més flexibles. La temporalització inicial detallada s'inclou a continuació en el següent diagrama de Gantt, on no es van tenir en compte festius ni horaris degut a que no hi ha la possibilitat de treballar en el present projecte amb un horari fix:

	Nombre de tarea	Duración	Comienzo	Fin	Pred	Nombres de los recursos
1	Començament del projecte	0 días	lun 18/10/10	lun 18/10/10		
2	PAC1 - Pla de treball	0 días	lun 25/10/10	lun 25/10/10		
3	PAC2	0 días	lun 22/11/10	lun 22/11/10		
4	PAC3	0 días	mar 21/12/10	mar 21/12/10		
5	Lliurament final	0 días	lun 17/01/11	lun 17/01/11	31	
6	Redacció del pla de treball	8 días	lun 18/10/10	lun 25/10/10	1	Pau;Ramon
7	Acceptació del pla de treball	1 día?	mar 26/10/10	mar 26/10/10	6;2	Ramon
8	Redefinició de fites	3 días	mié 27/10/10	vie 29/10/10	7	Pau;Ramon
9	Redefinicions de disseny	2,5 días?	sáb 30/10/10	lun 01/11/10	8	Pau;Ramon
10	**Càrrega de fitxers**	**7 días**	**lun 01/11/10**	**lun 08/11/10**	**9**	**Pau**
11	Desenvolupament	5 días	lun 01/11/10	sáb 06/11/10		Pau
12	Documentació	2 días	sáb 06/11/10	lun 08/11/10	11	Pau
13	**Funcions bàsiques de cerca**	**14 días**	**lun 08/11/10**	**lun 22/11/10**	**10**	**Pau**
14	Definició	4 días	lun 08/11/10	vie 12/11/10		Pau
15	Desenvolupament	7 días	vie 12/11/10	vie 19/11/10	14	Pau
16	Documentació	3 días	vie 19/11/10	lun 22/11/10	15	Pau
17	**Interfície gràfica**	**24 días**	**lun 22/11/10**	**jue 16/12/10**	**13**	**Pau**
18	Desenvolupament	16 días	lun 22/11/10	mié 08/12/10		Pau
19	Inclusió funcionalitats	3 días	mié 08/12/10	sáb 11/12/10	18	Pau
20	Documentació	5 días	sáb 11/12/10	jue 16/12/10	19	Pau
21	**Plot**	**4 días?**	**jue 16/12/10**	**lun 20/12/10**	**17**	**Pau**
22	Desenvolupament	3 días	jue 16/12/10	dom 19/12/10		Pau
23	Documentació	1 día?	dom 19/12/10	lun 20/12/10	22	Pau
24	**Funcions avançades**	**16 días**	**lun 20/12/10**	**mié 05/01/11**	**21**	**Pau**
25	Desenvolupament	12 días	lun 20/12/10	sáb 01/01/11		Pau
26	Documentació	4 días	sáb 01/01/11	mié 05/01/11	25	Pau
27	**Tests**	**8 días**	**mié 05/01/11**	**jue 13/01/11**	**24**	**Pau**
28	Definició	4 días	mié 05/01/11	dom 09/01/11		Pau
29	Realització i documentació	4 días	dom 09/01/11	jue 13/01/11	28	Pau
30	Compilació i ampliació de la docume	4 días	jue 13/01/11	lun 17/01/11	27	Pau
31	Tancament del projecte	0 días	lun 17/01/11	lun 17/01/11	30	Pau

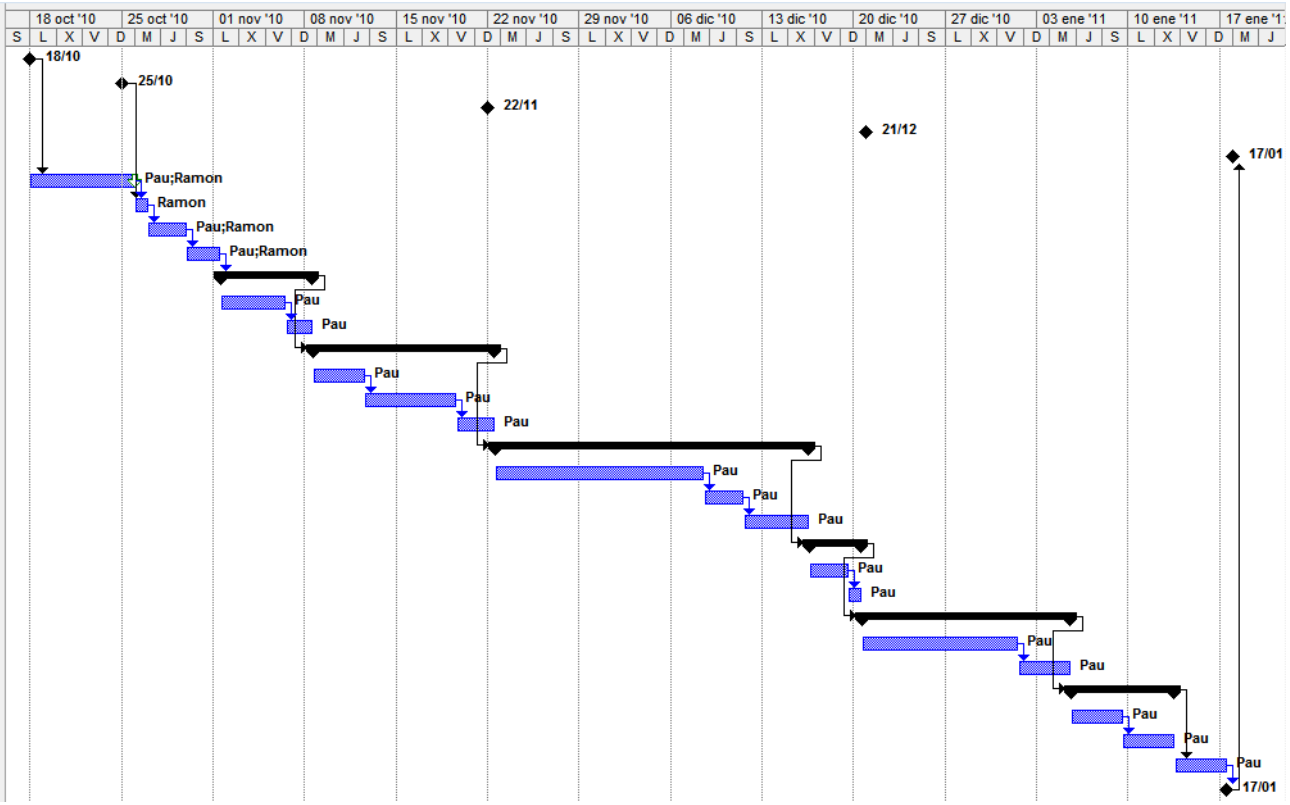

Fig. 1-A: Planificació inicial del Projecte

Malauradament no s'ha pogut completar tot el contingut que es volia desenvolupar pel que s'ha tingut que limitar aquest.

La part dedicada a la implementació de les funcions de la interfície gràfica va requerir molt més temps del que s'esperava, el que va afectar negativament en l'acompliment de les fites.

A més el desconeixement de l'eina va obligar a moure algunes fites i eliminar-ne d'altres tenint en compte que el producte final tingués parts complertes i usables enlloc de parts començades però inutilitzables.

Les parts eliminades han estat les següents:

- En el disseny i desenvolupament de la Interfície Gràfica s'ha eliminat la inclusió de funcionalitats de les pestanyes següents:
    - **Concordance Plot**
    - **Collocates**

  Les pestanyes sense funcionalitats implementades apareixen, ja que estan dissenyades, però deshabilitades, per tal de que es puguin habilitar de forma ràpida si es vol ampliar l'aplicació.

  A més, la pestanya **Word List** no marca les coincidències, amb el que només retorna la llista ordenada de totes les paraules. Per tant Hit Location en aquesta pestanya tampoc serveix.
  Global Settings i ToolPreferences.

- En aquest punt tampoc s'han implementat les funcionalitats de **Global Settings** i **Tool Preferences**, obrint les finestres però amb un avís indicant que aquestes no funcionen.

- En File només s'han implementat les parts relatives a obrir i tancar fitxers, així com Exit que tanca l'aplicació.

- Plot: El no haver implementat l'eina de dibuix ha afectat directament a que la pestanya Concordance Plot estigués deshabilitada.

- S'ha reduït el Test a un sol dia, ja que els cada funcionalitat implementada ja ha inclòs tests unitaris.

1.5. Productes obtinguts

Els productes generats son els següents:

Arxius de Qt Designer: Arxius obtinguts en el disseny i desenvolupament de les finestres i/o widgets de l'aplicació utilitzant Qt Designer.

Arxius Python: Els podem separar en tres tipus:

- Arxius obtinguts de transformar els anteriors (de Qt Dessigner) a arxius Python
- Arxius d'implementació on s'han implementat les funcionalitats i les classes necessàries.
- Arxiu pfc.pyw que és on tenim el main i s'encarrega de llançar tota l'aplicació.

Arxius de test: Son arxius de text utilitzats per a provar el funcionament general de l'eina i els tests unitaris.

La relació complerta d'arxius ve detallada en el capítol 11.

1.6. Capítols de la memòria

La memòria s'ha estructurat de manera didàctica, per tal de que el lector, seguint-la, pugui entendre la corba d'aprenentatge necessària i que pugui aprofitar els meus errors, accelerant aquesta.

Per tant es suposa que el lector no te coneixements previs del llenguatge i eines utilitzades, pel que si no es així es pot saltar la primera part.

A continuació s'explica el funcionament de l'aplicació seguint la estructura que te, per tal de facilitar una visió general, i després més específica de l'eina afavorint l'assimilació del funcionament, i la comprensió del codi per tal de facilitar l'ampliació

Els capítols s'estructuren de la següent manera:

1. Introducció: Aquest capítol.
2. Introducció a les eines: Informació sobre Python i PyQt, manuals seguits, problemes trobats.
3. Visió general de l'aplicació: Explicació de l'aplicació a nivell d'usuari.
4. Arquitectura de l'aplicació: Visió general de l'aplicació i l'arquitectura d'aquesta. També detalla quin arxiu conté cada part de l'aplicació.
5. Funcions bàsiques: Funcions bàsiques per a la cerca, ordenació, etc.
6. Disseny de les vistes: Detalla com estan dissenyades les diferents finestres i parts de l'aplicació en Qt Designer.
7. Estructures de dades i Models: Descripció de les estructures on s'emmagatzemen les dades i dels models utilitzats i explicació de com els fa servir PyQt.
8. Controladors: Explicació de com es creen les finestres i com es declaren les funcionalitats d'aquestes.
9. Main: Explicació de la funció que crida a totes les altres, així com la relació entre aquestes.
10. Com continuar: Com actuar per continuar el projecte. Passos a seguir. Reactivació dels ítems desactivats.
11. Relació d'arxius lliurats.
12. Conclusions: Conclusions del projecte.
13. Bibliografia: Enllaços a pàgines útils per al desenvolupament del projecte així com l'ampliació d'aquest.

2. Introducció a les eines

2.1. Entorn i eines

Durant el desenvolupament de la pràctica ha canviat l'entorn de treball ja que s'han utilitzat dos ordinadors diferents. S'ha utilitzat Windows XP i Windows 7 de 32 bits sense problemes de compatibilitat.

S'ha utilitzat la versió 2.1 de Python i la 4.8.1 de PyQt.

L'editor utilitzat també ha variat. Vaig començar amb IDLE, que es un editor inclòs amb Python, però a mida que la complexitat dels arxius amb els que havia de treballar va anant-se incrementant vaig haver de recórrer a altres opcions, ja que IDLE no mostra el número de línia a l'esquerra com fan altres. En realitat ho mostra a sota, però s'ha d'anar clicant i seleccionant la línia per que la indiqui. A més no te barra de desplaçament horitzontal, el que fa una mica incòmode el desplaçament si tenim línies llargues.

En resum, no es que IDLE tingui cap problema que n'impedeixi la utilització, si no que és més un tema de comoditat. Així que vaig optar per UltraEdit, que a més permet obrir diversos arxius evitant-me tenir-los tots a la barra d'eines. Per tal de poder usar-lo amb Python amb comoditat i que marquès les keywords, comentaris i demés només vaig haver de buscar la sintaxi a Internet, la qual no és difícil trobar, i afegir-la al programa. De totes maneres qualsevol editor de textos decent servirà.

Finalment l'entorn queda amb les següents eines instal·lades:

- Windows 7 32 bits i posteriorment Windows XP
- Python 3.1
- PyQt GPL 4.8.1
- IDLE i posteriorment UltraEdit.

A continuació es donen uns apunts sobre Python i PyQt.

2.2. Python

Com ja s'ha comentat el llenguatge de programació que s'ha utilitzat per a desenvolupar l'aplicació ha estat Python, concretament la versió 3.1. És un llenguatge força dinàmic sense grans complicacions i força estès, això fa que la cerca de informació i manuals a Internet sigui fàcil.

2.2.1. Instal·lació

Python es pot descarregar de la pròpia web del llenguatge: http://www.python.org/download/

Només serà necessari descarregar la versió 3.1 corresponent al sistema operatiu a utilitzar, en el meu cas Windows X86, pel que era un arxiu MSI que s'instal·la sense complicacions.

2.2.2. Aprenentatge

En aquest apartat es donaran uns apunts sobre l'experiència amb la utilització de Python per tal d'estalviar a altres desenvolupadors de l'aplicació tenir els mateixos errors que he tingut jo i que així puguin estalviar temps.

La primera recomanació és buscar un manual per fer-se una idea del funcionament i la sintaxi del llenguatge. És important, però, assegurar-se de que el manual és per a Python 3, ja que hi ha algunes diferències amb Python 2 que pot fer que els exemples no funcionin. I personalment em va portar bastant temps descobrir per que aquests no em funcionaven.

Durant l'estudi de Python recomano parar força atenció en el tractament de cadenes i llistes.

Tenir cura amb la versió és extensible a l'hora de buscar informació o exemples de Python en Internet. Si no hi ha cap indicador de la versió a la pàgina, hi ha una manera bastant fàcil, només s'ha de buscar una crida a la funció print. En la versió 2 es crida amb la utilització d'un espai: *print x*, mentre que la versió 3 fa servir parèntesis: *print(x)*.

Una de les qüestions que m'ha causat més errors ha estat que Python no utilitza claudàtors o altres elements per indicar que cert codi està dins d'un bucle o una funció, simplement es basa en la indexació. Per tant s'ha d'anar amb compte, ja que un espai de més o de menys pot voler dir un error. De fet vaig haver de canviar la configuració d'UltraEdit per tal de que tractés la tabulació com quatre espais simples per que al principi em donava errors. Així que s'ha d'anar amb compte amb que l'editor no inclogui caràcters invisibles en el text.

Un cop comencem a desenvolupar recomano l'ús dels Python Manuals que inclou Python, ja que correspondran a la versió instal·lada. Tot i així, per a la cerca d'exemples concrets, de vegades és inevitable haver de recorre a Internet.

2.3. PyQt

PyQt és una adaptació de la biblioteca gràfica Qt per al llenguatge Python. La biblioteca Qt serveix per a desenvolupar interfícies gràfiques multiplataforma. És la biblioteca que utilitzen KDE i Google Earth entre altres.

A més de incloure les llibreries que utilitzarem, PyQt inclou una eina gràfica per al disseny d'interfícies, el que facilita la feina.

La versió que s'ha utilitzat ha estat la 4.8.1.

2.3.1. Instal·lació

Per a instal·lar PyQt necessitarem tenir Python ja instal·lat. Un cop tinguem aquesta premissa podem descarregar l'eina de la pàgina web dels desenvolupadors d'aquesta:

http://www.riverbankcomputing.co.uk/software/pyqt/download

S'haurà de tenir en compte, a més de que sigui la versió desitjada, que sigui pel nostre sistema operatiu i per la nostra versió de Python.

L'arxiu per Windows és un executable amb l'instal·lador, així que no tindrem major problema per finalitzar la instal·lació correctament.

2.3.2. Aprenentatge

A la web es poden trobar molts manuals sobre PyQt, tot i així per tal de familiaritzar-se ràpidament tant amb el funcionament de les llibreries com amb Qt Designer recomano seguir PyQt4.pdf que s'inclou amb la memòria. Recomano especial atenció, i fer els exemples, en especial el de la calculadora.

Tot i així, a l'hora de programar, recomano tenir a ma la documentació que proporciona Riverbank. En especial aquella sobre les classes, tot i que la major part d'exemples son en C++, pel que haurem d'adaptar el codi.

En darrer lloc, si posem el nom d'una classe a un cercador obtenim documentació d'aquesta en altres webs. El problema que això implica, es que aquesta classe te el mateix nom en PyQt que en Qt, pel que tindrem encara més exemples en C++ i funcions que potser no corresponguin a la nostra versió de PyQt. Per tant desaconsello, en general, la utilització de cercadors per accedir a les classes, utilitzant aquests només per a cerques molt concretes en trobar-se amb un problema.

No entraré a descriure com usar les eines que proporciona PyQt ja que els manuals ja contenen aquesta informació. Per tant aquesta part queda fora de l'abast de la documentació, quedant relegada als documents que s'adjunten.

3. Visió general de l'aplicació

L'aplicació a dissenyar consisteix en un analitzador de concordances la qual ha d'incloure diverses funcionalitats treballant sobre un o més fitxers de text.

L'eina te una aparença similar a AntConc i inclou en un costat el llistat d'arxius, on permet seleccionar-los per treballar sobre arxius concrets (com en el cas de la pestanya File View) o per eliminar-los. A la part dreta te uns panells que mostra amb pestanyes, és a dir, canviant la pestanya es selecciona un o altre panell amb diferents funcionalitats. També inclou en la part inferior diversos elements per interactuar amb els panells. Aquests elements canvien segons l'element que s'obri.

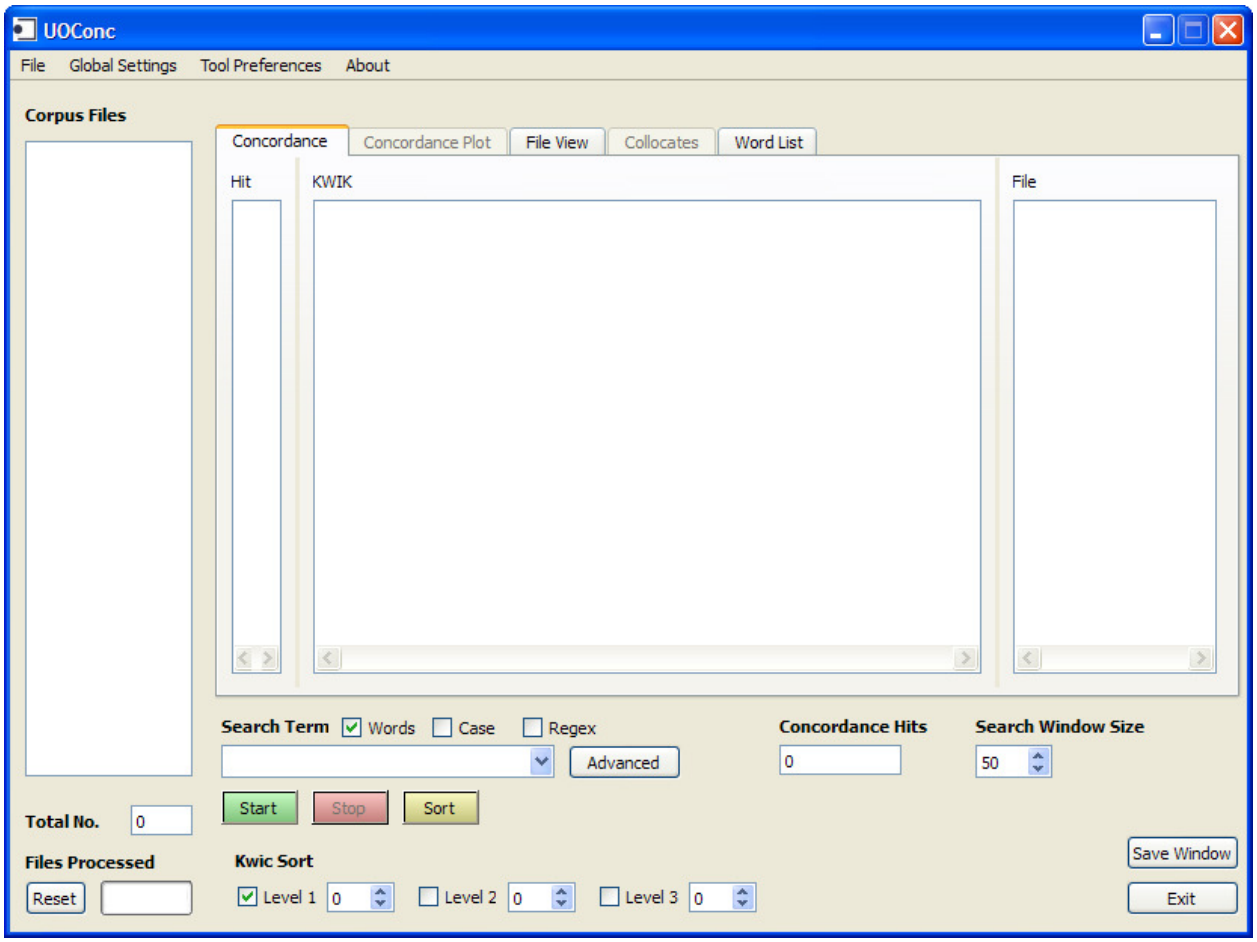

Fig. 3-A: Pantalla principal de l'aplicació

Podem afegir o treure arxius a la llista mitjançant les opcions del menú File a la barra de menú.

Es recomana utilitzar AntConc per a comprovar el funcionament del programa, en especial de les pestanyes que no estan implementades.

El Readme de AntConc a la web d'Antlab pot ajudar a comprendre el funcionament. A més cercant a Internet, podem trobar, explicacions, presentacions i informació en anglès, castellà o altres idiomes.

3.1. Execució de l'aplicació

Per a poder executar l'aplicació és necessari tenir Python instal·lat, així com la biblioteca PyQt.

Un cop instal·lats s'ha de descomprimir l'arxiu pubach_files.zip. Més endavant es detalla l'estructura d'arxius d'aquest zip.

Per a executar l'aplicació serà suficient amb executar l'arxiu pfc.pyw.

En cas de voler executar-la per a depuració s'haurà de canviar l'extensió de l'arxiu de pyw a py. Això s'explica amb major detall en el capítol que detalla com continuar amb el projecte.

3.2. Pestanyes

Aquest apartat defineix les diferents funcionalitats que proporciona l'eina, aquestes van directament relacionades amb les pestanyes.

3.2.1. Concordance

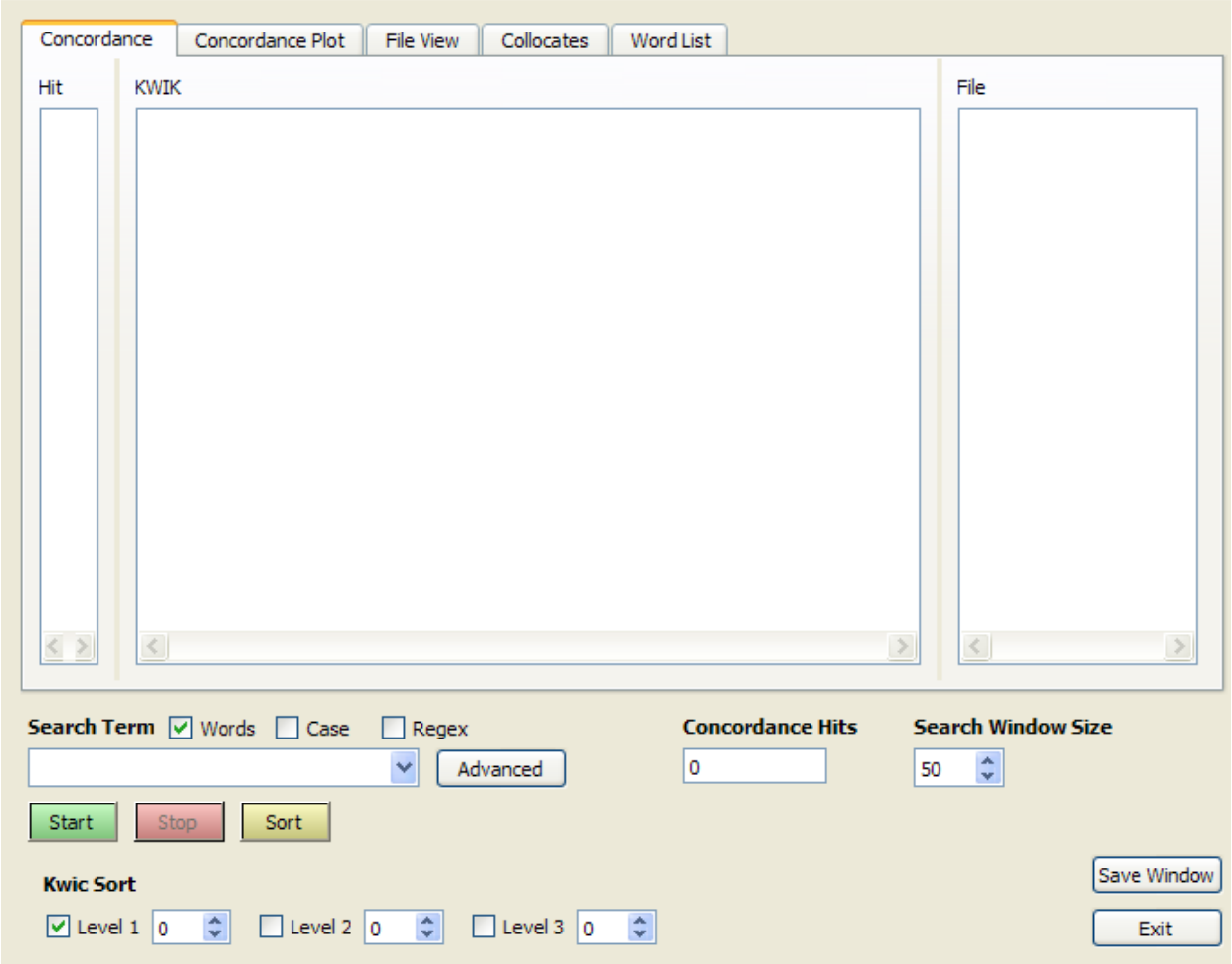

Fig. 3-B: Pestanya Concordance

Cerca una expressió o expressions en els arxius.

Podem introduir la expressió en Search Term o utilitzar llistats de paraules si obrim la finestra Advanced. Aquesta finestra, a més, permetrà definir un conjunt de paraules (Context Word) i un rang per indicar que les coincidències trobades, a més, han de tenir les paraules d'aquest conjunt dins del rang.

El rang es defineix com dos posicions, sent cada posició el nombre de paraules a la dreta o a l'esquerra de la coincidència. Per tant 0 correspon a la coincidència trobada, 1L és una paraula a l'esquerra de la coincidència i 2R dos paraules a la dreta.

És a dir que un rang de 1L 2R inclou la coincidència, la paraula de l'esquerra i dues a la dreta. En total 4 paraules. Aquestes 4 paraules haurien d'incloure les Context Word per tal de que la coincidència sigui vàlida.

En les paraules a buscar (Search Words) podem utilitzar comodins i expressions regulars, a més es permet diferenciar o no entre majúscules i minúscules i seleccionar si es tracta d'una paraula complerta o una cadena qualsevol.

La cerca retorna una llista de línies amb coincidències o concordances vàlides, és a dir línies que contenen alguna de les paraules de la cerca i compleixen amb les condicions establertes per les Context Words (si n'hi ha).

Aquesta línia conté tants caràcters al davant i al darrere de la coincidència com s'indiqui a Search Window Size.

A més es permet l'ordenació de la llista segons ordenació alfabètica de la paraula concordant o qualsevol paraula al voltant d'aquesta. Es permeten fins a tres nivells d'ordenació.

Podrem, a més, seleccionar cada línia, el que ens portarà a la vista de l'arxiu en la pestanya File View, remarcant la paraula, per tal de veure el context en que apareix.

3.2.2. Concordance Plot (deshabilitat)

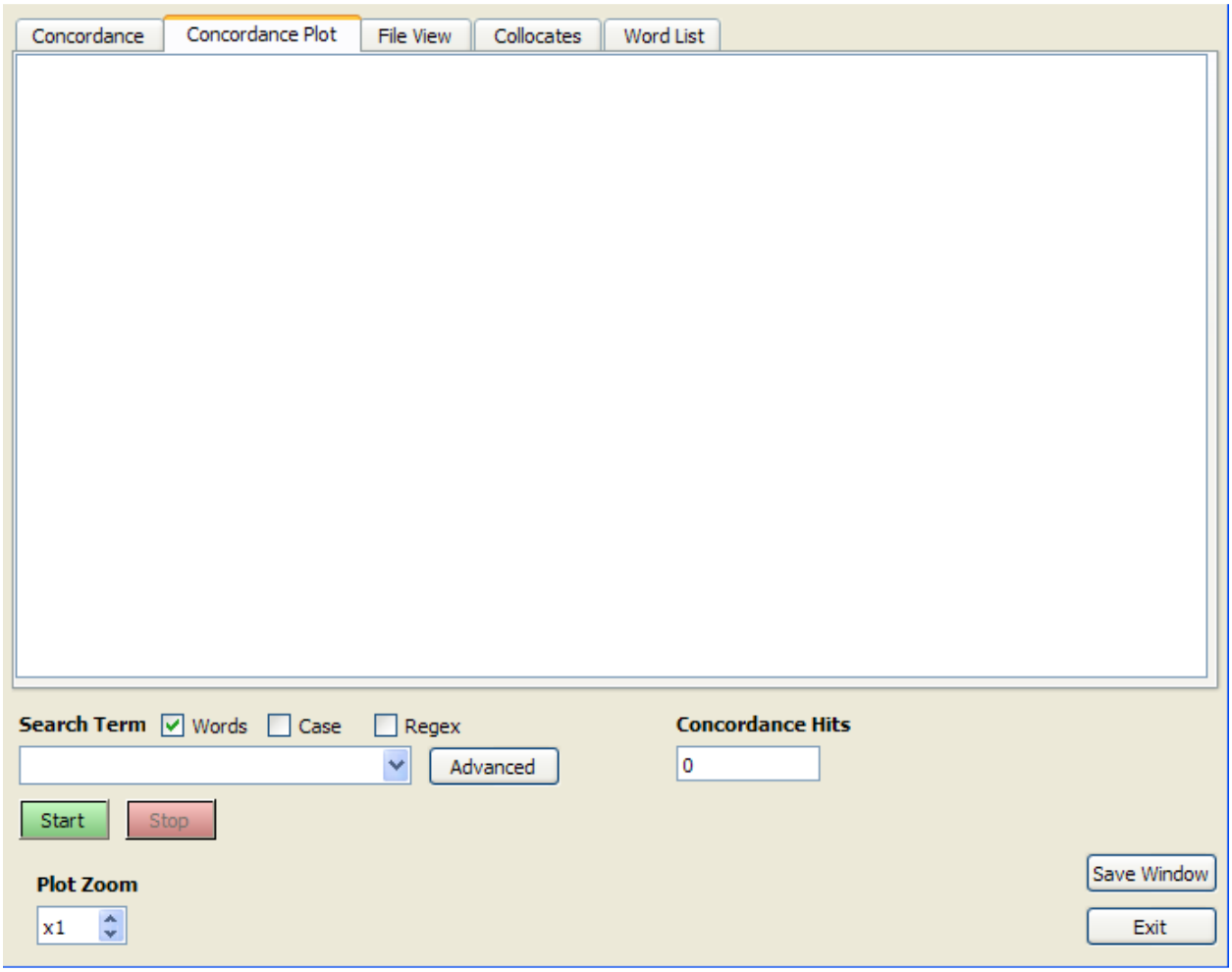

Fig. 3-C: Pestanya Concordance Plot

La cerca és igual que a la pestanya Concordance, amb la diferència que ara es genera un gràfic en el que cada arxiu s'identifica amb una barra horitzontal on s'indica amb una línia vertical la posició relativa en el text de cadascuna de les concordances.

Seleccionant una de les línies verticals es navega a la paraula en la pestanya File View, tal i com passa en el cas anterior.

La funcionalitat del Plot no s'ha implementat, pel que aquesta pestanya queda deshabilitada.

3.2.3. File View

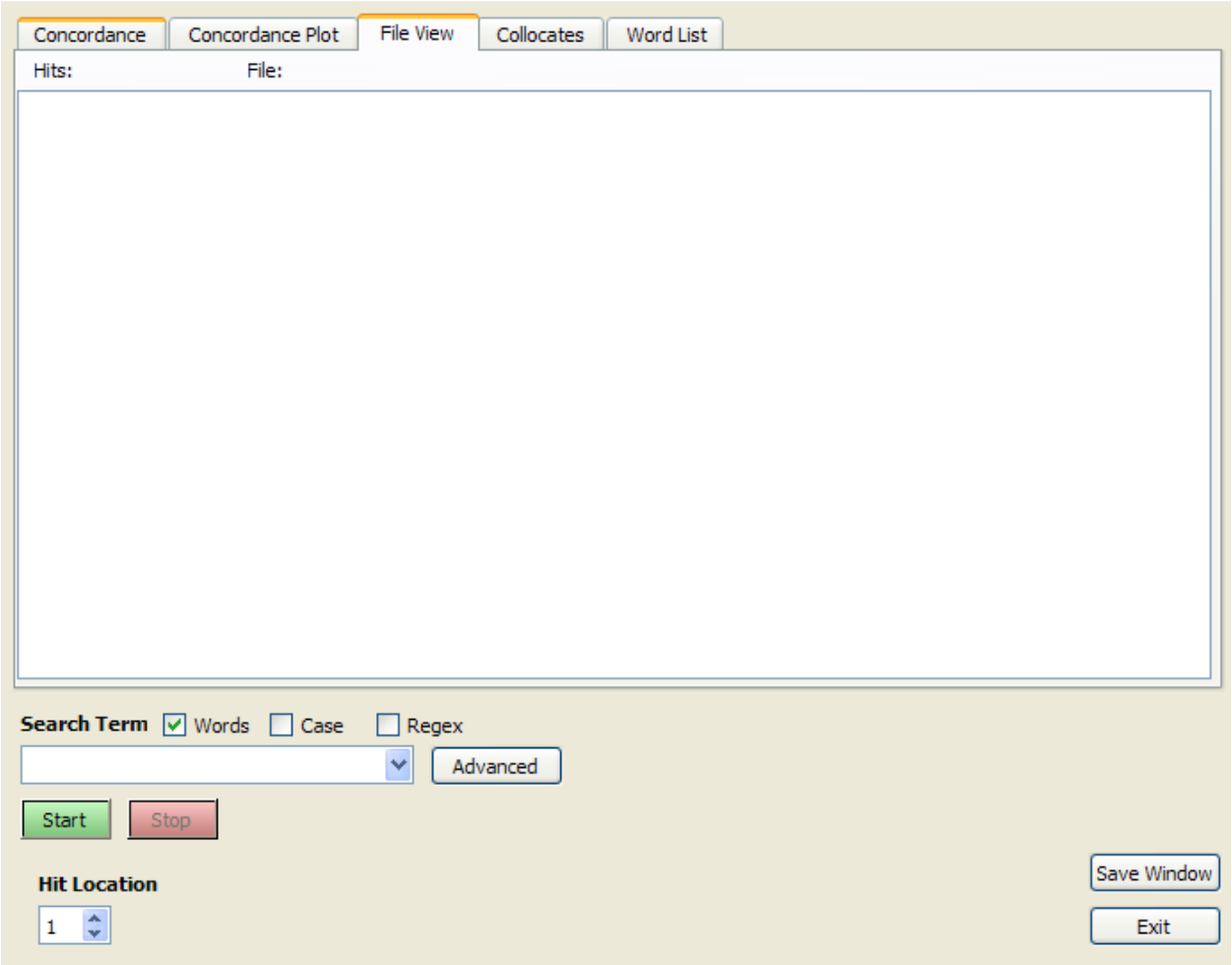

Fig. 3-D: Pestanya File View

Mostra el text original per cada arxiu. Permet cerques com en els casos anteriors, però en aquest cas les concordances apareixeran ressaltades.

Permet navegar per les concordances usant el control Hit Location.

3.2.4. Collocates (deshabilitat)

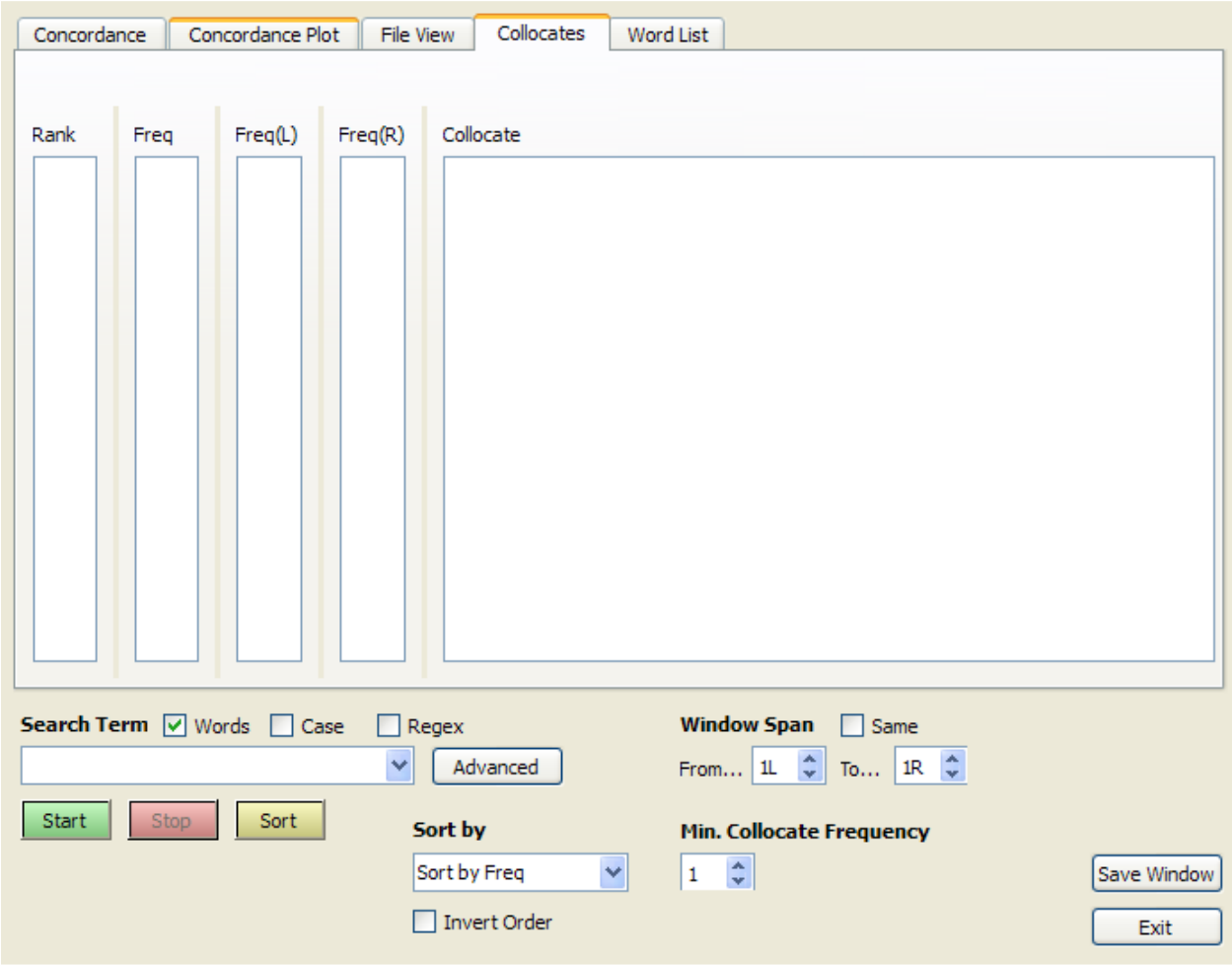

Fig. 3-E: Pestanya Collocates

Retorna un llistat amb les paraules que apareixen a la dreta i esquerra de les concordances incloent les mateixes concordances. Indica si aquestes paraules apareixen a la dreta o a l'esquerra de la concordança. Permet seleccionar en quin rang fer la cerca en nombre de paraules a la dreta i nombre de paraules a l'esquerra de manera independent. També permet limitar el mínim de cops que ha d'aparèixer una paraula per constar al llistat.

També permet fer ordenació seguint diferents criteris tant de manera ascendent com descendent, aquests criteris son la freqüència amb que apareixen en general, a la dreta o esquerra de la concordança, alfabèticament, etc.

La funcionalitat de Collocates no ha estat implementada, pel que la pestanya queda deshabilitada.

3.2.5. Word List

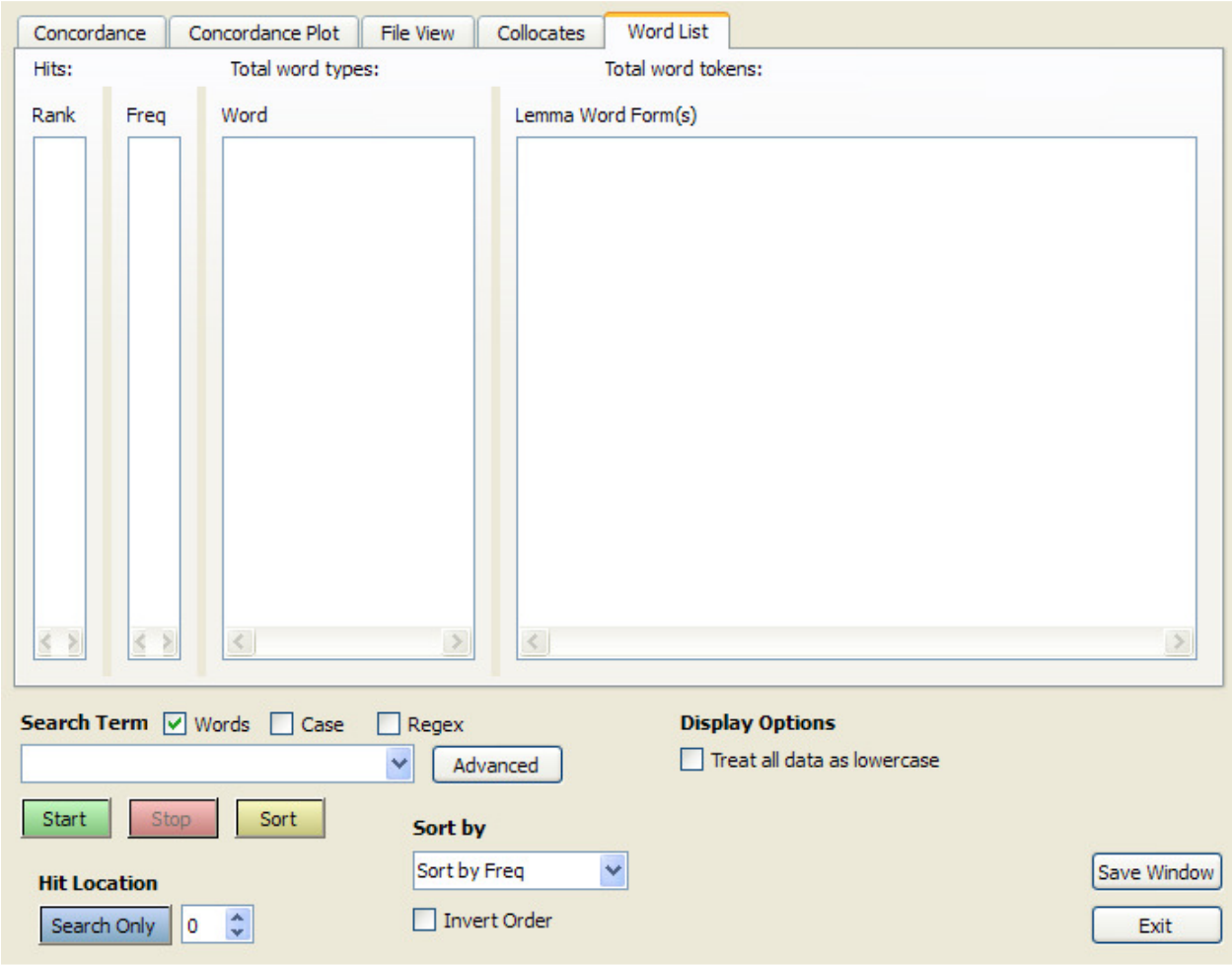

Fig. 3-F: Pestanya Word List

Mostra un llistat amb totes les paraules i la freqüència amb que apareixen. Permet seleccionar si es diferencia o no entre majúscules i minúscules.

Permet l'ordenació per freqüència, ordre alfabètic i ordre alfabètic per finalització de la paraula, en tots casos l'ordenació pot ser ascendent o descendent.

La funcionalitat per ressaltar les concordances no està implementada.

En fer clic en una paraula fa la cerca d'aquesta en la pestanya Concordance.

3.3. Barra de menú

La barra de menú de UOCConc conté quatre elements:

- File: És un menú desplegable amb opcions per obrir i tancar fitxers sobre els quals treballar i per tancar l'aplicació. Te altres opcions deshabilitades ja que no s'han implementat.

- Global Settings: Obre una finestra amb les opcions del programa, actualment cap està operativa, pel que abans d'obrir-la mostra un avís.

- Tool Preferences: Obre una finestra amb les opcions de cada pestanya, el que es mostra, etc. Actualment tampoc està operatiu, pel que abans d'obrir la finestra també mostra un avís.

- About: Mostra informació sobre el programa, la llicència, etc.

3.4. Altres finestres

A més de la finestra principal de UOCConc, hi ha quatre més, tres que responen a botons a la barra de menú i una altra per a la cerca avançada.

3.4.1. Advanced Search

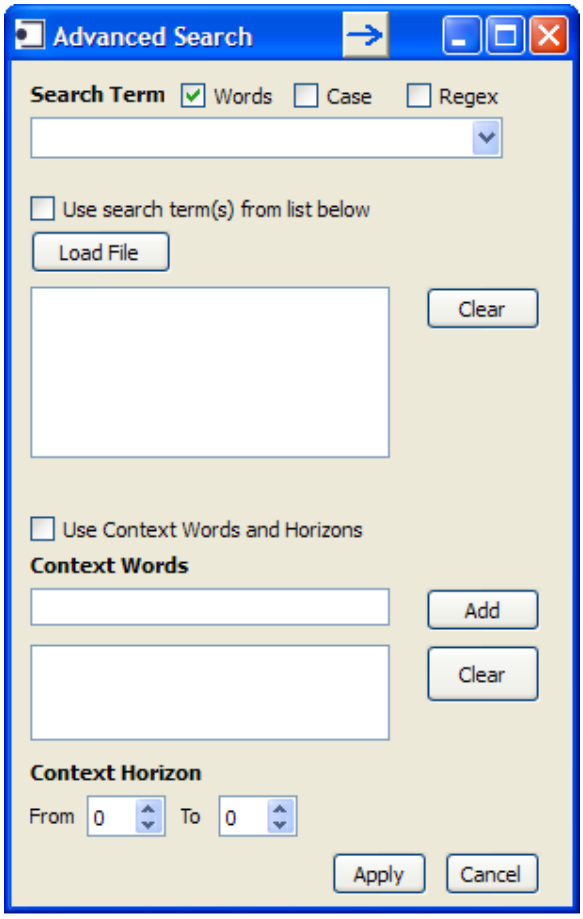

Fig. 3-G: Finestra Advanced Search

S'obre en fer clic al botó Advanced en la zona de cerca de la pantalla principal.

Permet seleccionar l'expressió a cercar, a més guarda els termes cercats anteriorment.

Dona l'opció d'escollir si l'expressió és una paraula sencera, si s'ha de fer diferenciació entre majúscules i minúscules o si és una expressió regular.

Fins aquí cap diferència amb el que tenim a la finestra principal. Ara veiem les opcions avançades:

Permet utilitzar un llistat de termes editats manualment o importats des d'un arxiu.

Permet l'ús de paraules de context. Aquestes paraules hauran d'estar en un rang determinat al voltant de la concordança. El rang també es defineix aquí.

Aquesta part s'explica amb major detall en l'explicació de la pestanya Concordance.

3.4.2. Global Settings (no implementat)

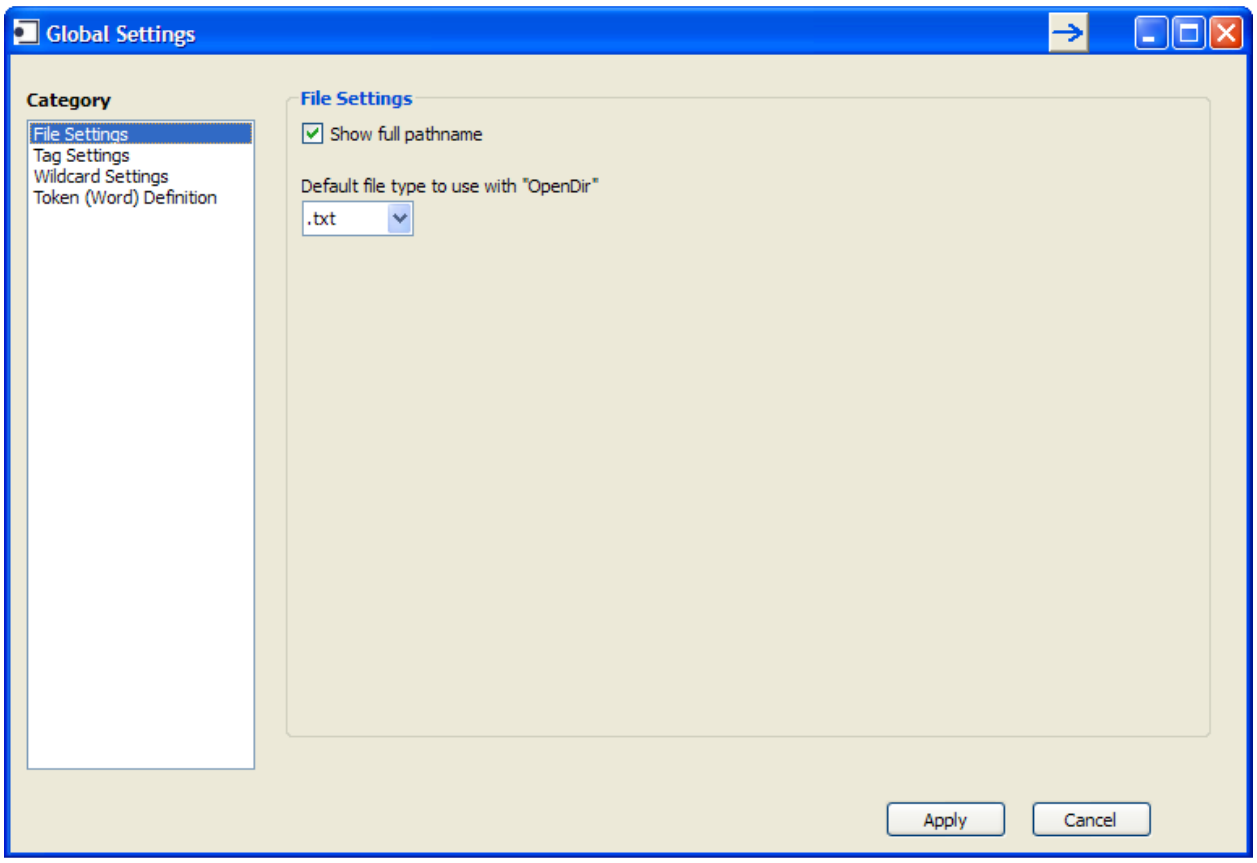

Fig. 3-H: Finestra Global Settings

S'accedeix des del botó amb el mateix nom a la barra de menú. Ofereix diverses opcions generals sobre el funcionament de l'eina.

La més interessant és la possibilitat de canviar la definició de paraula. Tot i que UOCConc, actualment no permet fer canvis en aquest punt, el codi està preparat per que una modificació sobre la definició sigui senzilla d'implementar.

Actualment abans d'obrir la finestra mostra un missatge informant de que aquesta no està operativa. Tot i així el disseny i la navegació estan implementats.

3.4.3. Tool Preferences (no implementat)

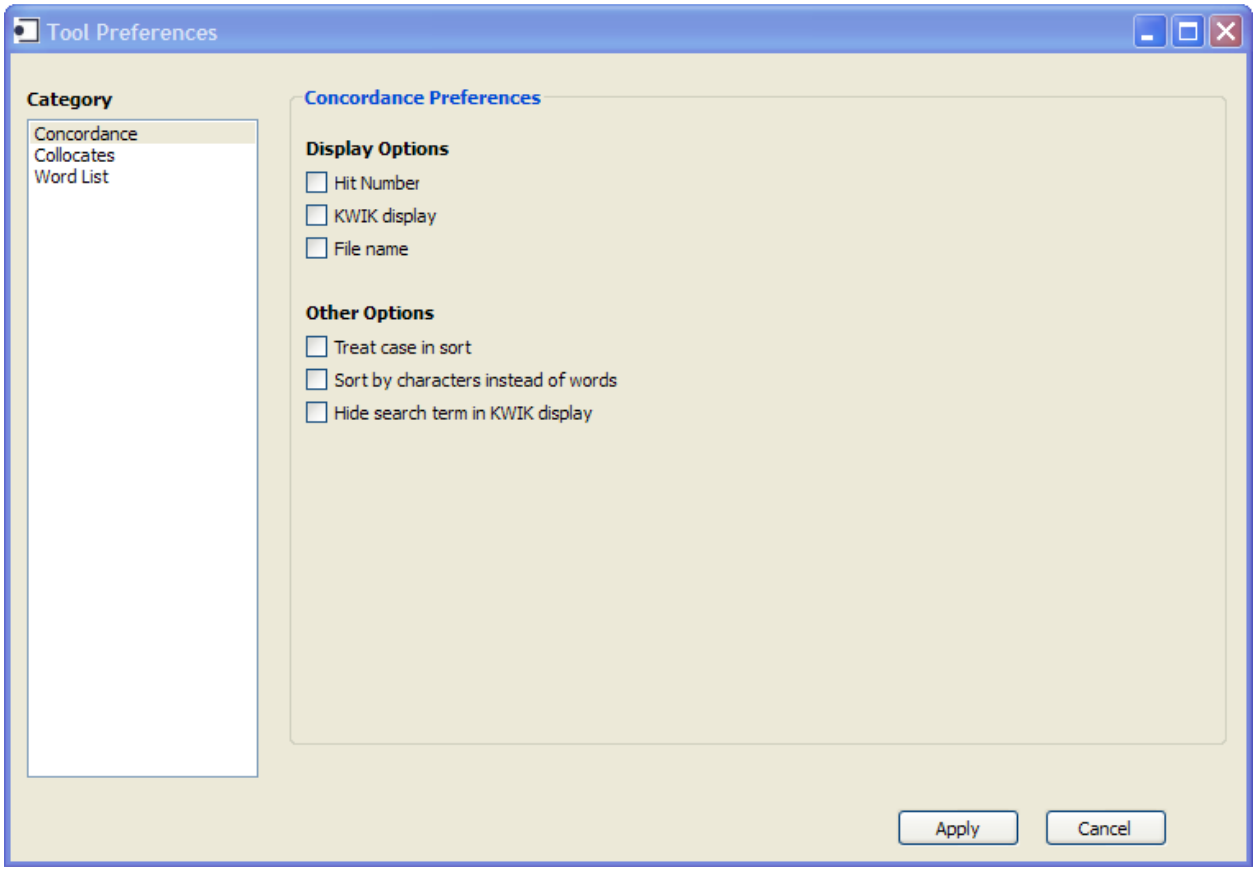

Fig. 3-I: Finestra Tool Preferences

S'accedeix des del botó amb el mateix nom a la barra de menú. Ofereix opcions concretes per a cada pestanya, com les columnes a mostrar o altres opcions.

Actualment abans d'obrir la finestra mostra un missatge informant de que aquesta no està operativa. Tot i així el disseny i la navegació estan implementats.

3.4.4. About

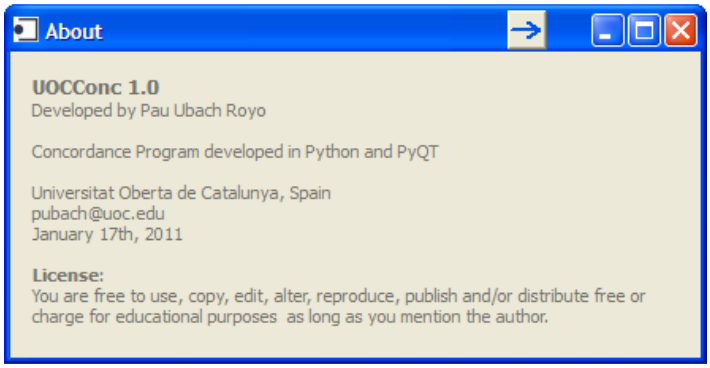

Fig. 3-J: Finestra About

S'accedeix des del botó amb el mateix nom a la barra de menú.

Ofereix informació sobre l'eina, l'autor i la llicència.

4. Arquitectura de l'aplicació

L'aplicació segueix el patró de tres capes MVC. És bastant fàcil fer distinció entre les tres parts.

La vista ve donada per els arxius creats a partir dels fitxers resultants del treball amb Qt Designer. Els fitxers son els següents:

- **MainWindow.py**: Conté la vista de la finestra principal.
- **AdvancedSearch.py**: Conté la vista de la finestra Advanced Search.
- **About.py**: Conté la vista de la finestra About.
- **GlobalSettings.py**: Conté la vista parcial de la finestra Global Settings. Les vistes dels forms que s'obren en la part dreta en seleccionar una o altra opció en el menú de l'esquerra estan definides en altres fitxers. Aquests es poden distingir per que el nom acaba amb la paraula Settings.
- **ToolPreferences.py**: Igual que en el cas anterior conté la vista parcial de la finestra Tool Preferences. Les vistes dels formularis que s'obren en la part dreta en seleccionar una o altra opció en el menú de l'esquerra estan definides en altres fitxers. Aquests es poden distingir per que el nom acaba amb la paraula Preferences.
- **FileSettings.py**: Conté el contingut que es mostra en seleccionar la categoria File Settings a la finestra Global Settings.
- **TagSettings.py**: Conté el contingut que es mostra en seleccionar la categoria Tag Settings a la finestra Global Settings.
- **WildcardSettings.py**: Conté el contingut que es mostra en seleccionar la categoria Wildcard Settings a la finestra Global Settings.
- **TokenDefinitionSettings.py**: Conté el contingut que es mostra en seleccionar la categoria Token (Word) Definition a la finestra Global Settings.
- **ConcordancePreferences.py**: Conté el contingut que es mostra en seleccionar la categoria Concordance a la finestra Tool Preferences.
- **CollocatesPreferences.py**: Conté el contingut que es mostra en seleccionar la categoria Collocates a la finestra Tool Preferences.
- **WordListPreferences.py**: Conté el contingut que es mostra en seleccionar la categoria Word List a la finestra Tool Preferences.

A més d'aquests fitxers es lliuren els originals de Qt Designer, amb els mateixos noms i extensió .ui.

La capa model defineix les estructures de dades i els models que utilitzarà l'aplicació. Les classes per a les estructures de dades venen definides en el fitxer **Dades.py**, mentre que els models estan a **Models.py**.

La capa controlador conté, a més de les funcions bàsiques, tota la part relativa a la interacció de l'usuari amb l'aplicació. Està formada per els arxius següents:

- **pfc_func.py**: Conté aquestes funcions bàsiques.
- **MainWindowFuncs.py**: Controla la part relativa a la finestra principal, a més d'incloure les funcions relacionades amb aquesta.
- **OtherWindowFuncs.py**: Controla totes les altres finestres. Amb excepció de les accions relacionades per la finestra principal que venen definides en MainWindowFuncs.
- **pfc.pyw**: Conté el *main* que obre la finestra principal.

Les relacions entre les diferents capes venen definides pel mode clàssic, tot i que s'ha de tenir en compte que el model inclou les estructures de dades.

Els capítols següents detallen cadascuna d'aquestes parts, tot i que l'ordre no correspon a

Hi ha informació sobre l'ús de MVC amb Qt a:

http://doc.qt.nokia.com/latest/model-view-programming.html

5. Funcions bàsiques

Les funcions bàsiques proveeixen les funcionalitats bàsiques necessàries per a executar la majoria d'accions que requereix l'aplicació. Aquestes haurien de ser vistes com a caixes negres.

El codi de les funcions està al fitxer **pfc_func.py**, però el que hauria d'interessar d'aquestes és les entrades, la funció que fan –i no com la fan– i la sortida que retornen.

- **wordsInRange(index, file, contextFrom, contextTo)**:
  - Funció: Retorna el llistat de paraules en un rang definit al voltant d'una paraula.
  - Paràmetres:
    - index: **int** – La posició de la paraula al voltant de la qual es buscarà.
    - file: **string** – Nom de la variable que conté el contingut de l'arxiu on volem buscar.
    - contextFrom: **string** – Posició de la paraula tal i com s'ha definit a l'explicació de la pestanya Concordance. Ha de ser un zero, o un nombre seguit de la lletra L o R. Indica el nombre de paraules a la dreta o esquerra on comença el rang per a cercar context words.
    - contextTo: **string** – Ha de tenir el mateix format que contextFrom. Indica el nombre de paraules a la dreta o esquerra on acaba el rang per a cercar context words.
  - Retorna: **list** – Retorna el llistat de paraules en el rang al voltant de la paraula donada.

- **wordPos(index, file, contextFrom, contextTo)**:
  - Funció: Similar a l'anterior, però en aquest cas retorna tuples amb les posicions inicial i final de cada paraula.
  - Paràmetres:
    - index: **int** – La posició de la paraula al voltant de la qual es buscarà.
    - file: **string** – Nom de la variable que conté el contingut de l'arxiu on volem buscar.
    - contextFrom: **string** – Posició de la paraula tal i com s'ha definit a l'explicació de la pestanya Concordance. Ha de ser un zero, o un nombre seguit de la lletra L o R. Indica el nombre de paraules a la dreta o esquerra on comença el rang per a cercar context words.
    - contextTo: **string** – Ha de tenir el mateix format que contextFrom. Indica el nombre de paraules a la dreta o esquerra on acaba el rang per a cercar context words.
  - Retorna: **Llistat de tuples** – Retorna un llistat de tuples amb la posició inicial i fina de cara paraula en el rang al voltant de la paraula donada.

- **checkContext(index, file, contextw, contextFrom, contextTo)**:
    - Funció: Comprova si existeix una paraula o llistat d'aquestes en un rang definit al voltant d'una expressió.
    - Paràmetres
        - index: **int** – La posició de la paraula al voltant de la qual es buscarà.
        - file: **string** – Nom de la variable que conté el contingut de l'arxiu on volem buscar.
        - contextw: **string** o **list** – Paraula o paraules que han d'estar en un rang definit al voltant de la paraula cercada (Context Words).
        - contextFrom: **string** – Ha de ser un zero, o un nombre seguit de la lletra L o R. Indica el nombre de paraules a la dreta o esquerra on comença el rang per a cercar context words.
        - contextTo: **string** – Ha de tenir el mateix format que contextFrom. Indica el nombre de paraules a la dreta o esquerra on acaba el rang per a cercar context words.
    - Retorna: **bool** – Retorna True si existeixen totes les paraules en la entrada contextw i False si no es troben totes.

- **searchTerms (terms, file, start, end, case, words, regex, contextw, contextFrom, contextTo)**:
    - Funció: és la funció de cerca principal. Retorna el llistat de coincidències.
    - Paràmetres:
        - terms: **string** o **list** – Expressió o llistat d'expressions a cercar.
        - file: **string** – Nom de la variable que conté el text de l'arxiu on volem buscar.
        - start: **int –** Posició inicial des d'on començar a buscar.
        - end: **int** – posició final on acabar la cerca. -1 per buscar fins EOF.
        - case: **bool** – Cerca sensible a majúscules i minúscules.
        - words: **bool** – Indica si l'expressió ha de ser una paraula sencera o, per el contrari, pot ser part d'una paraula.
        - regex: **bool** – Activa la cerca d'expressions regulars.
        - contextw: **string** o **list** – Paraula o paraules que han d'estar en un rang definit al voltant de la expressió cercada (Context Words).
        - contextFrom: **string** – Ha de ser un zero, o un nombre seguit de la lletra L o R. Indica el nombre de paraules a la dreta o esquerra on comença el rang per a cercar context words.
        - contextTo: **string** – Ha de tenir el mateix format que contextFrom. Indica el nombre de paraules a la dreta o esquerra on acaba el rang per a cercar context words.
    - Retorna: **Llistat de tuples** amb l'índex on s'ha trobat una coincidència i el text. La segona part és necessària per saber la longitud de la cadena trobada, en cas de expressions regulars o de cerques de múltiples mots, i així poder obtenir les línies a les que corresponen amb x caràcters davant i al darrere. (Sense la longitud d'aquests mots no podríem limitar pel darrere).

- **sortWList (wordList, file, l1, l2, l3)**:
    - Funció: Retorna el llistat de paraules ordenades donat un fitxer i passant la llista de coincidències.
    - Paràmetres:
        - wordList: **llista de tuples** – Expressió i índex de cadascuna de les coincidències..
        - file: **string** – Nom de la variable que conté el text de l'arxiu on volem buscar.
        - l1: **int** – Posició de la primera paraula per a la cerca amb respecte a la expressió coincident.
        - l2: **int** – Posició de la segona paraula per a la cerca amb respecte a la expressió coincident.
        - l3: **int** – Posició de la tercera paraula per a la cerca amb respecte a la expressió coincident.
    - Retorna: **Llistat de tuples** que entren per wordList ordenades segons el criteri que es passa a l1, l2 i l3.

- **sortWList (wordList, l1, l2, l3)**:
    - Funció: Modificació de la funció anterior. Endreça el llistat de concordances en més d'un arxiu i retorna l'ordre en forma de llistat de enters.
    - Paràmetres:
        - wordList: **llista de tuples** – Expressió, índex i nom de l'arxiu de cadascuna de les coincidències. La tupla pot contenir més informació, però aquests tres valors son els únics requerits per aquesta funció.
        - l1: **int** – Posició de la primera paraula per a la cerca amb respecte a la expressió coincident.
        - l2: **int** – Posició de la segona paraula per a la cerca amb respecte a la expressió coincident.
        - l3: **int** – Posició de la tercera paraula per a la cerca amb respecte a la expressió coincident.
    - Retorna: **List** de **int** amb l'ordre que ocuparien en una llista segons el criteri que es passa amb l1, l2 i l3.

- **wordList(file, sortF, sortD, case)**:
    - Funció: Retorna un llistat ordenat de les paraules que té un arxiu.
    - Paràmetres:
        - file: **string** – Nom de la variable que conté el text de l'arxiu on volem buscar.
        - sortF: **string** – Tipus d'ordenació. Accepta tres valors:
            - Freq: Ordenació per freqüència en la que apareixen les paraules.
            - Word: Ordenació alfabètica.
            - WordEnd: Ordenació alfabètica per el final de la paraula.
        - sortD: Ordenació ascendent o descendent. Ha de ser 'desc' per a fer la ordenació descendent, en qualsevol altra cas aquesta serà ascendent.
        - case: **bool** – Llistat sensible a majúscules i minúscules.
    - Retorna: **Llistat** de **tuples** on cada tupla inclou la paraula, la paraula en ordre invers (per a la ordenació) i el nombre de cops que apareix.

- **wordList(sortF, sortD, case)**:
    - Funció: Modificació de l'anterior funció. La cerca s'estén a tots els fitxers oberts.
    - Paràmetres:
        - sortF: **string** – Tipus d'ordenació. Accepta tres valors:
            - Freq: Ordenació per freqüència en la que apareixen les paraules.
            - Word: Ordenació alfabètica.
            - WordEnd: Ordenació alfabètica per el final de la paraula.
        - sortD: Ordenació ascendent o descendent. Ha de ser 'desc' per a fer la ordenació descendent, en qualsevol altra cas aquesta serà ascendent.
        - case: **bool** – Llistat sensible a majúscules i minúscules.
    - Retorna: **Llistat** de **tuples** on cada tupla inclou la paraula, la paraula en ordre invers (per a la ordenació) i el nombre de cops que apareix.

- **isWord(text)**
    - Funció: Retorna True o False indicant si el text que es passa és una paraula. No és una funció realment necessària en aquest punt del projecte, ja que és suficient amb usar la funció per strings que incorpora Python isalpha(). Aquesta funció es va crear amb vista a la configuració del significat de paraula que es podria canviar a Global Settings. D'aquesta manera només s'haurà de tocar aquesta funció i no totes les altres on necessitem fer una comprovació.
    - Paràmetres:
        - text: **string** – text a analitzar.
    - Retorna: **bool** – True si el text és una paraula.

- **returnLine(word, file, schWinSz)**
    - Funció: Retorna la línia on apareix una expressió. Substitueix els canvis de línia per espais per tal de donar línies més complertes i així retornar un context més ric.
    - Paràmetres:
        - word: **tupla** – Inclou la posició (índex) de la paraula en l'arxiu i la paraula.
        - file: **string** – Nom de la variable que conté el text de l'arxiu.
        - schWinSz: **int** – Nombre de caràcters a cada costat de la paraula que han d'aparèixer.
    - Retorna: **tupla** – Retorna la línia i la posició de la expressió en aquesta.

- **paintLine(line,wordPos,word,l1,l2,l3)**
    - Funció: Retorna la línia que es passa com a paràmetre afegint codi HTML per tal de mostrar-la pintada.
    - Paràmetres:
        - line: **string** – Línia que s'ha de pintar.
        - wordPos: **int** – Índex on comença l'expressió
        - word: **string** – Expressió
        - l1: **tupla** – Conté dos enters amb la posició inicial i final de la paraula d'ordenació de primer nivell.
        - L2: **tupla** – Conté dos enters amb la posició inicial i final de la paraula d'ordenació de segon nivell.
        - L3: **tupla** – Conté dos enters amb la posició inicial i final de la paraula d'ordenació de tercer nivell.
    - Retorna: **string** – Retorna la línia amb el codi HTML inserit corresponent als colors que es mostraran en la pestanya Concordance.

Insisteixo en que és més important comprendre la funció, de cara als següents capítols, que entendre el funcionament intern de cadascuna d'aquestes. Tot i així pot ser interessant seguir el codi per comprendre com funciona Python o com s'ha utilitzat per aconseguir la funcionalitat bàsica necessària.

6. Disseny de les vistes

Per al disseny de les vistes s'ha utilitzat Qt Designer. Si mai s'ha utilitzat aquesta eina recomano no passar d'aquest punt sense seguir el tutorial proporcionat junt amb aquesta memòria en el document PyQT4.pdf. El seguiment i comprensió dels exercicis d'aquest ajudarà a comprendre aquest i els següents capítols.

Assumint, doncs, que s'ha seguit el tutorial i que el lector ja te una base de PyQt i Qt Designer podem obviar la introducció i explicacions de l'eina, fora de l'abast d'aquesta memòria, i passar a explicar com s'ha dissenyat cadascuna de les parts de l'aplicació.

6.1. Finestra principal

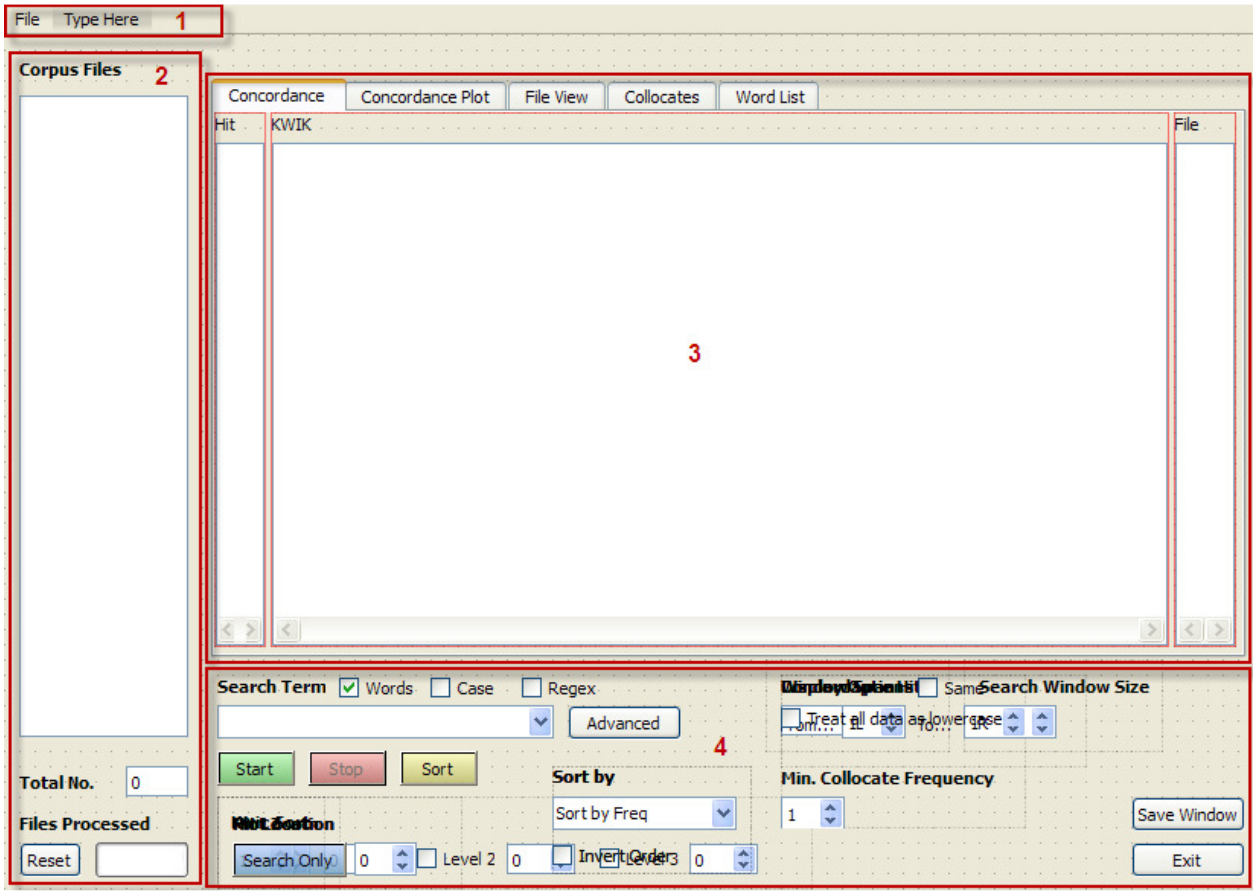

Fig. 6-A: Finestra principal des del punt de vista del disseny

La finestra principal es construeix a partir del template Main Window, que a diferència dels Widgets inclou barres de menú i altres opcions com la barra d'estat que he eliminat.

Podem diferenciar quatre parts:

1. Barra de menú
2. Zona de fitxers
3. Panell de pestanyes
4. Zona del formulari

A més dedicarem un darrer apartat a les senyals programades en aquest punt.

Per últim, és molt important que en crear fitxers Python partint de fitxers de Qt Designer no manipular aquests, ja que si fem cap canvi en el disseny i tornem a convertir el fitxer de Qt Designer a Python perdrem aquests canvis. Enlloc d'això s'ha d'utilitzar els fitxers que criden a aquestes vistes.

6.1.1. Barra de menú

Construir un menú amb Qt Dessigner és fàcil, és suficient amb anar escrivint directament els noms de menús i les accions, que és com Qt anomena als botons dins d'un menú. També podem afegir separadors. Després, en el property editor podem configurar una selecció de tecles per a executar aquesta acció i deshabilitar alguna de les opcions.

En aquest punt només s'ha definit el menú File i les accions d'aquest, ja que els altres botons d'aquesta barra es volen definir com a accions o botons i no com a menús, cosa que no permet fer Qt Designer.

6.1.2. Zona de fitxers

La zona de fitxers és senzilla, inclou una llista utilitzant la classe QListView per mostrar els fitxers. Per ara només ens ocupem de la vista, pel que no ens preocuparem per com introduirem o eliminarem fitxers d'aquesta fins més endavant.

A més s'inclou una línia on mostrar el nombre total de fitxers, una barra de progrés, i un botó Reset, emulant el disseny d'AntConc.

Fins ara l'únic que s'ha fet ha estat agafar elements de Widget Box i posar-los a la finestra. En cas de voler canviar alguna propietat es fa des de el Property Editor.

6.1.3. Panell de pestanyes

Per a fer el panell de pestanyes utilitzem la classe QTabWidget. Fent clic amb el botó dret permet afegir i eliminar pestanyes. A més el mateix submenú permet moure'ls, tot i que també podem fer-ho arrossegant la pestanya a la posició desitjada.

Per canviar el nom de les pestanyes haurem d'accedir a la propietat currentTabName que permet canviar el nom de la pestanya seleccionada. Pel que només haurem d'anar recorrent les pestanyes per poder canviar el nom de cadascuna.

Per a explicar el contingut de les pestanyes sense caure en la repetició es poden agrupar **Concordance**, **Collocates** i **Word List**, ja que totes elles tenen una distribució similar. Es tracta principalment d'una taula on podem redimensionar l'amplada de les columnes. Cada columna te una barra de desplaçament horitzontal independent de les altres columnes. Aquesta barra és un problema per a utilitzar les classes QTableView o QTableWidget – la diferència es pot trobar al glossari buscant View i Widget – que proporciona Qt, ja que aquestes inclouen una barra de desplaçament per tot l'ample de la taula i no independentment per columna.

Per tant s'ha d'emular el funcionament de taula amb altres ítems, permetent redimensionar l'ample de les columnes, amb barres de desplaçament horitzontal independents per cada columna i una única barra vertical a la dreta.

La solució ha estat utilitzar una classe QListView per a cada columna, eliminant la barra de desplaçament vertical de totes excepte la darrera columna i fent que el desplaçament de totes les columnes funcionés amb la barra de la darrera. Aquesta part, però, queda fora de la vista i s'explica en el capítol dedicat al controlador de la finestra principal.

En crear les llistes s'ha afegit el nom a sobre i s'han unit utilitzant la classe QVBoxLayout. Després per a poder redimensionar les columnes s'ha utilitzat la classe QSplitter. Es seleccionen tots els QVBoxLayout corresponents a la combinació de títol i columna (o llista) i amb el botó dret obrim el submenú Lay out. Apliquem *Lay out Horizontally in Splitter*.

Per tal de que la modificació de l'ample de les columnes mitjançant resizing sigui correcta hem de definir Horizontal Policy al Property Editor de totes les QListView com Ignored, exceptuant una que definirem com Expanding, d'aquesta manera la columna s'adaptarà a la mida de les altres. A més definirem per totes elles que mostrin sempre la barra de desplaçament horitzontal, i només les columnes que queden a la dreta mostraran la barra de desplaçament vertical si és necessari.

La pestanya **Word List**, a més de la taula, inclou diversos labels i zones per a text que s'afegeixen sense major complicació.

La pestanya **Concordance Plot** conté simplement un ítem de la classe QGraphicsView, però com que no s'ha implementat la funcionalitat de la pestanya no he tingut temps de comprovar que sigui la classe que més convingui. Per tant no descarto que aquest element hagi de canviar en continuar amb el desenvolupament de l'aplicació.

La pestanya **File View** consta d'un quadre de text de la classe QTextBrowser que ocupa gairebé tot l'espai disponible, exceptuant la part superior del panell que conté dos labels i zones per a text com passa amb Word List.

6.1.4. Zona de formulari

En la zona de formulari s'ha inclòs tots els elements que es mostren per cadascuna de les pestanyes.

Aquests elements apareixen es mostren o s'amaguen depenent de la pestanya que hi ha oberta, per tant en veure'ls tots en Qt Designer ens trobem en que alguns es superposen. Cosa que no passa en obrir l'aplicació.

Per al disseny s'han fet grups depenent del contingut, zones i pestanyes per a les que havien d'aparèixer i s'ha ficat cada grup dintre d'un element de la classe QFrame, cosa que facilita moure els grups en Qt Designer i al mateix temps facilita la feina del controlador ja que pot treballar i fer aparèixer o desaparèixer grups sencers enlloc de treballar amb elements independents.

Alguns camps dels formularis tindran comportaments especials, això es definirà amb el controlador d'aquesta finestra.

El botó Stop s'ha deshabilitat ja que no s'ha implementat la funcionalitat per a parar una cerca o ordenació. Aquest tema es tracta amb major detall en el capítol dedicat al controlador de la finestra principal.

Tampoc funciona el botó Save Window ja que no te cap funcionalitat assignada.

6.1.5. Senyals

Qt Designer permet connectar senyals enviades per un element amb accions predefinides d'un altre element. Així doncs s'han connectat els següents elements:

Sender	Signal	Receiver	Slot
actionExit	triggered()	MainWindow	close()
exitButton	clicked()	MainWindow	close()

La primera correspon a tancar la finestra quan el botó o acció Exit en el menú File és clicat. La segona la tanca en clicar el botó Exit a la part inferior dreta de la finestra.

Tal i com es pot comprovar corresponen a funcionalitats molt simples i directes.

6.2. Advanced Search

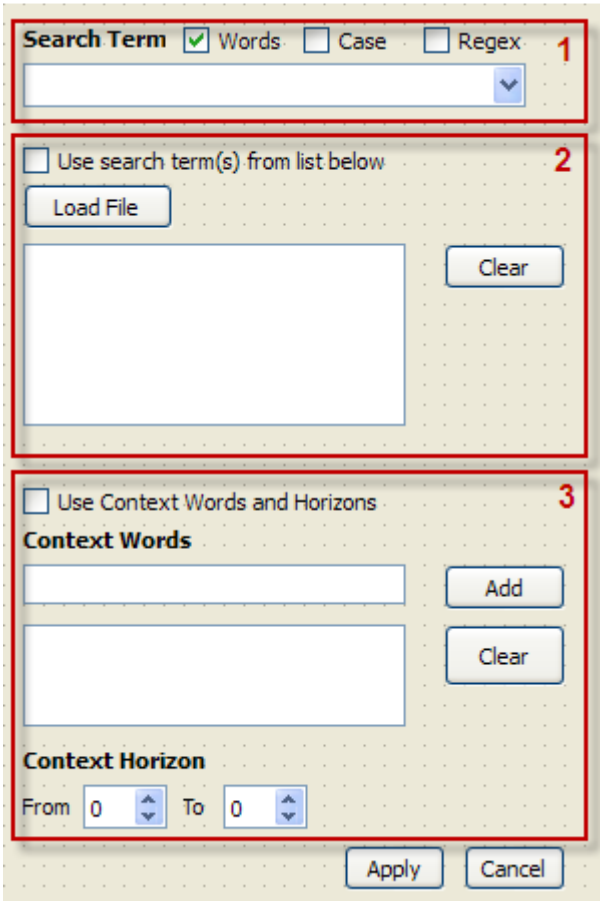

Fig. 6-B: Finestra Advanced Search des del punt de vista del disseny

La finestra Advanced Search es construeix a partir d'un template de tipus Widget. Podem diferenciar tres parts en el formulari que mostra, a més dels botons Apply i Cancel.

En primer lloc tenim un formulari similar al que apareix a la finestra anterior, el qual no te major complicació per al disseny, però que serà necessari més endavant que funcioni sincronitzat amb el de la finestra principal de l'aplicació.

En segon lloc tenim la part que permet utilitzar un llistat de paraules per a la cerca. Consta de dos botons i un quadre de text.

La tercera part permet definir un llistat de paraules i el rang, per tal de que els resultats de la cerca hagin de tenir les paraules del llistat dins del rang definit. Consta de la línia de dalt, editable. El botó Add que permet afegir la paraula a la llista de sota, i la llista. En aquest cas es tracta d'un element de la classe QListWidget i no QListView. La diferència és que un Widget permet introduir directament la informació i no depèn d'un model. L'única peculiaritat que resta destacar és la connexió que es definirà sobre la senyal clicked() del botó clear.

Les connexions entre elements són les següents:

Sender	Signal	Receiver	Slot
clearContextButton	clicked()	contextWordList	clear()
cancelButton	clicked()	AdvancedSearch	close()

La primera connexió fa que el botó clear de la tercera zona buidi el quadre de text del costat. Això no es pot fer amb el botó clear de la segona zona ja que al costat te un simple quadre de text enlloc d'un llistat.

La segona connexió tanca la finestra.

6.3. Global Settings

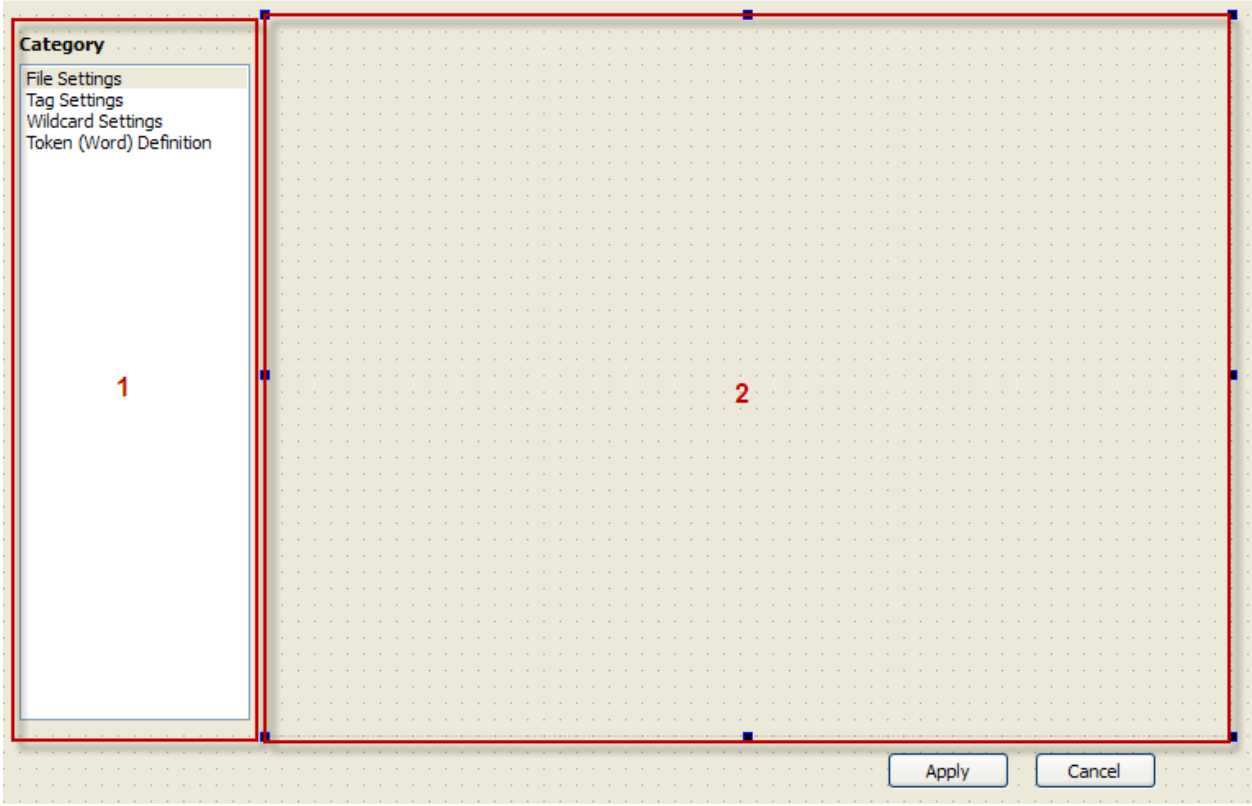

Fig. 6-C: Finestra Global Settings des del punt de vista del disseny

La finestra Global Settings és, a diferència de les anteriors, composta. Mentre que les altres finestres es definien amb una vista, aquesta te una vista principal tal i com es veu a la imatge, i una més per cadascuna de les opcions seleccionables en la llista Category.

Tant la vista corresponent a aquesta finestra com les que corresponen a les opcions es creen a partir del template Widget.

Podem diferenciar dues parts. La primera part conté el label Category i un QListWidget que farem servir com a menú. Obrint el menú contextual de la llista creada podrem editar els elements que conté.

La segona part és un quadre que utilitza la classe QFrame. Aquest servirà per que el controlador carregui la vista corresponent a la opció seleccionada.

A més tenim un parell de botons en la part inferior.

A continuació es llisten les altres vistes i les senyals.

6.3.1. File Settings

Fig. 6-D: Vista de File Settings

Es crea a partir del template Widget.

És una vista senzilla composta per una caixa (QGroupBox) que conté tots els altres elements. Aquesta caixa permet posar un títol al conjunt.

Dins de la caixa tenim una casella i un desplegable.

No s'han definit senyals per a aquesta vista.

6.3.2. *Tag Settings*

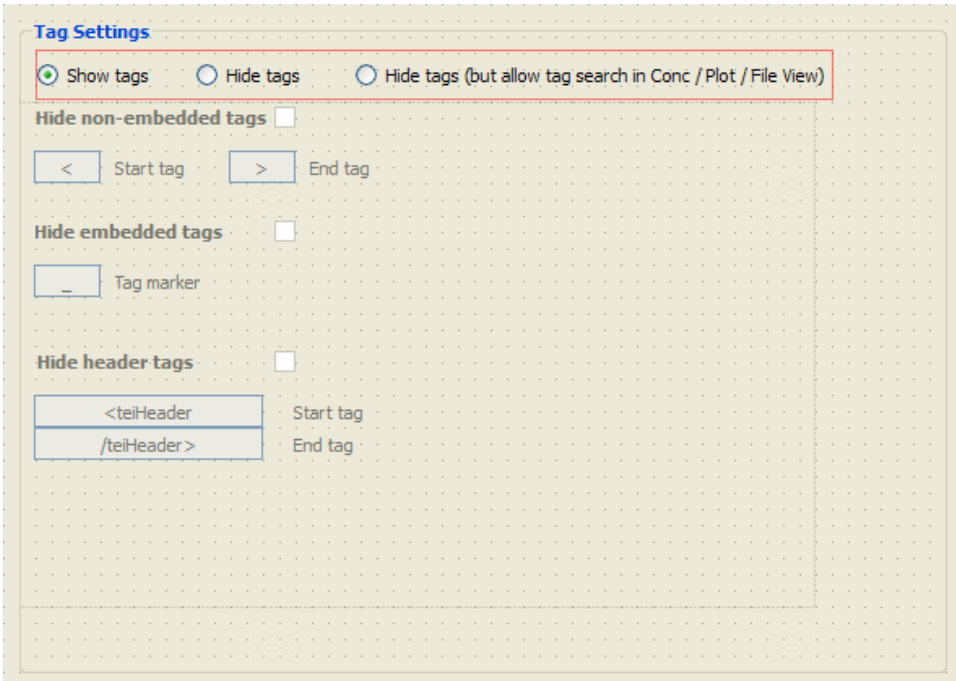

Fig. 6-E: Vista de Tag Settings

Es crea a partir del template Widget.

També conté una caixa amb el títol. Conté diversos ítems deshabilitats ja que s'activen depenent de la selecció. Actualment aquesta funcionalitat no està implementada, però les seleccions fetes en la part superior haurien de permetre habilitar i deshabilitar les diferents parts.

No hi ha res més del que es veu i no s'han definit senyals per a aquesta vista.

6.3.3. *Wildcard Settings*

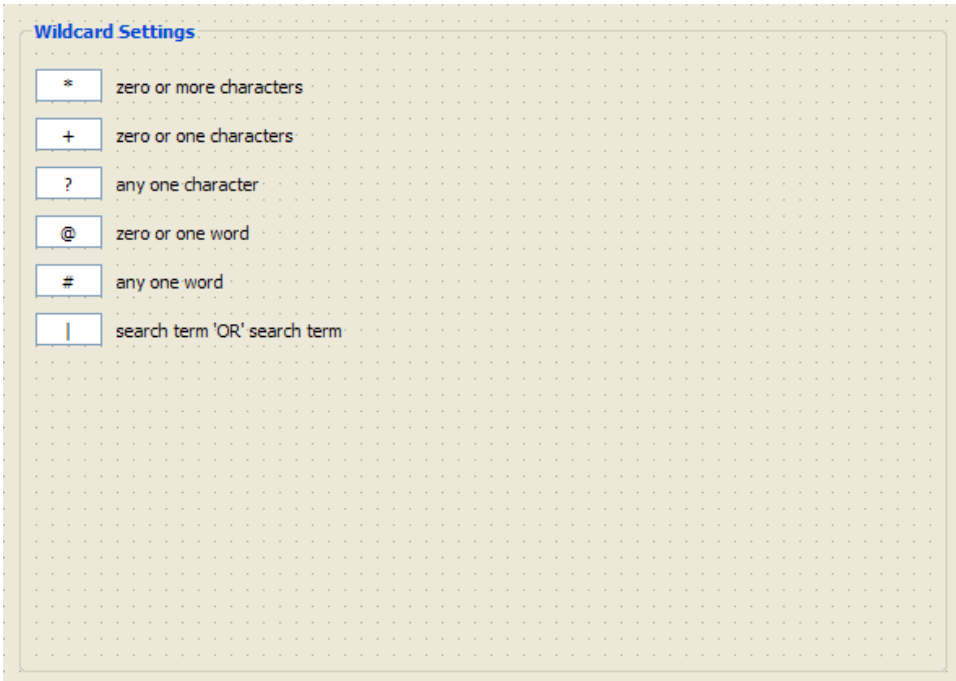

Fig. 6-F: Vista de Wildcard Settings

Es crea a partir del template Widget.

També conté una caixa amb el títol on s'allotgen caixes per text amb labels. Tampoc s'han definit senyals per a aquesta vista.

6.3.4. Token Definition Settings

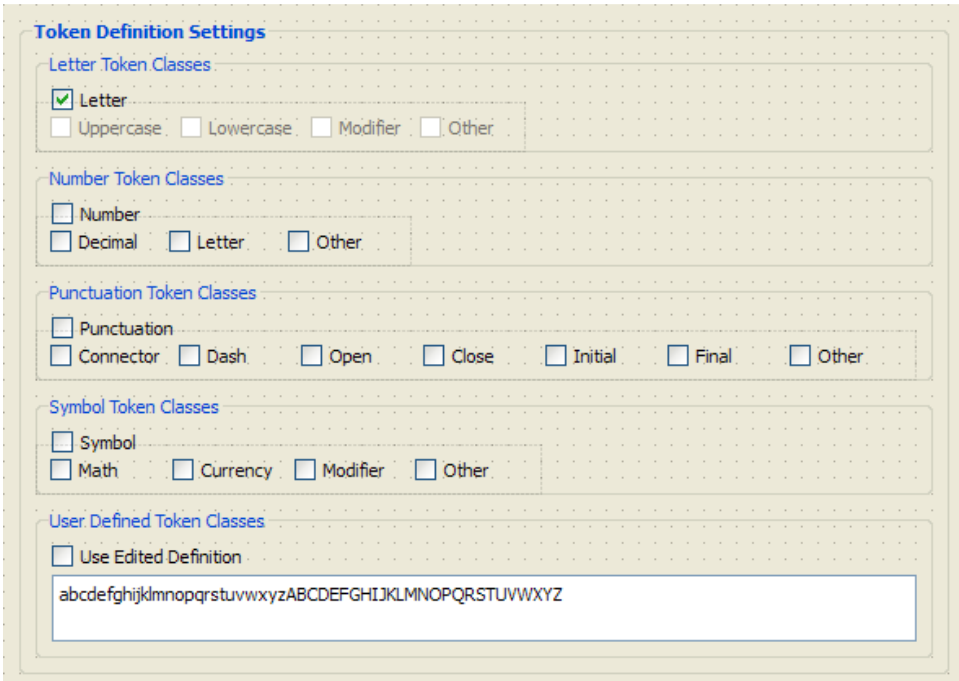

Fig. 6-G: Vista de Token Definition Settings

Com els anteriors es crea a partir del template Widget.

Dins la caixa amb el títol tenim altres caixes similars (QGroupBox) on dins apareixen diverses caselles. Tot i que la funcionalitat no està implementada, marcar la casella superior d'un grup desactiva el conjunt de caselles de sota. Tenint en compte això, s'han agrupat les caselles en grups (QFrame) per tal de permetre deshabilitar o habilitar grups sencers i no haver de fer la feina casella per casella.

En aquest cas tampoc hi ha senyals definides.

6.3.5. Senyals

Tal i com s'ha comentat només la finestra Global Settings te senyals definides.

Sender	Signal	Receiver	Slot
pushButton_2	clicked()	GlobalSettings	close()

pushButton_2 correspon al botó Cancel que tanca la finestra en ser clicat.

6.4. Tool Preferences

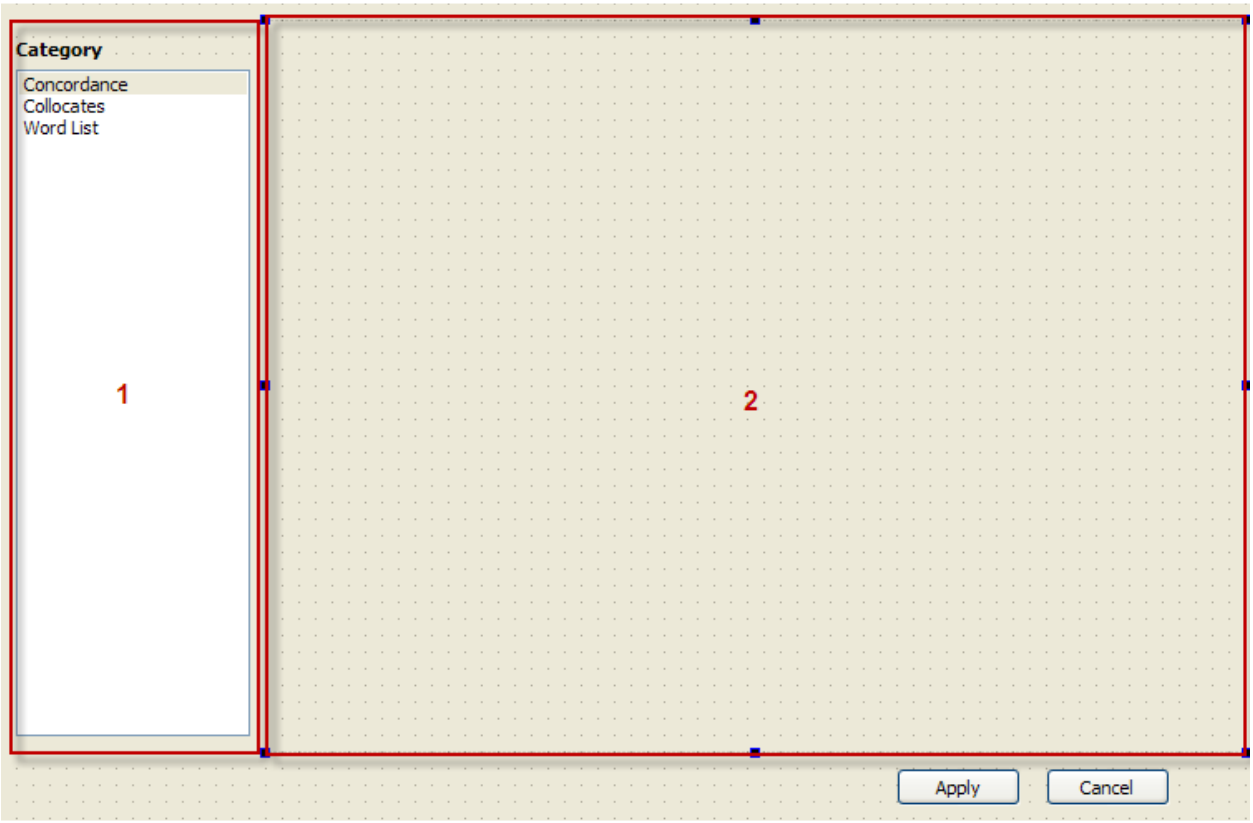

Fig. 6-H: Finestra Tool Preferences des del punt de vista del disseny

La finestra Tool Preferences funciona de manera similar a l'anterior. L'única diferència són les opcions que es llisten sota Category. Per tant les vistes "filles" canvien, pel que es detallen a continuació. Obviarem les senyals ja que també son iguals al cas anterior.

6.4.1. Concordance Preferences

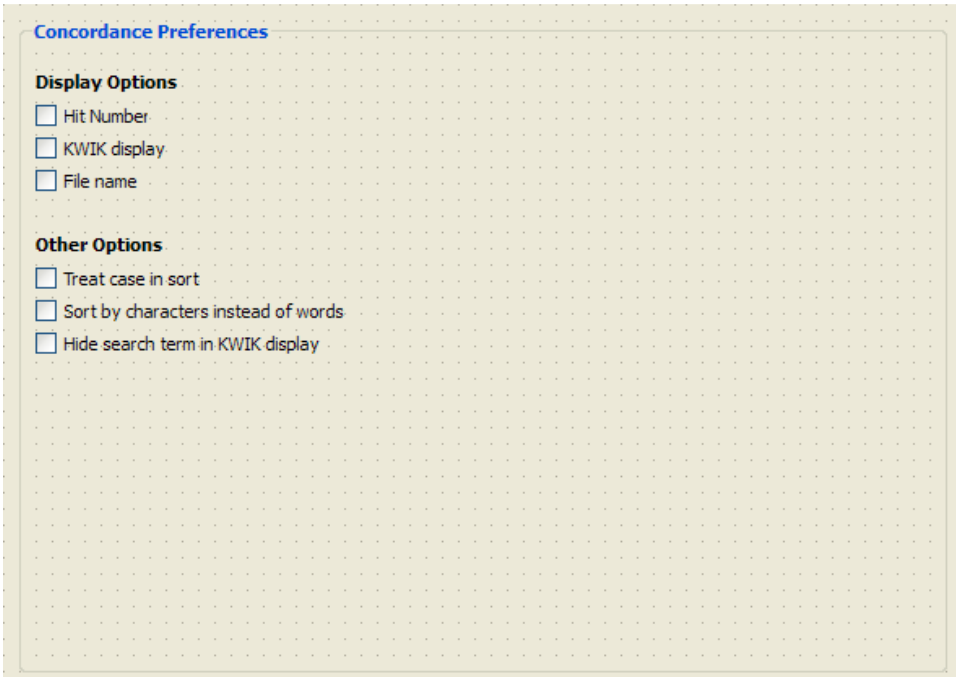

Fig. 6-I: Vista de Concordance Preferences

Es crea a partir del template Widget. Conté la caixa amb el títol i diverses caselles amb labels.

No s'han definit senyals per a aquesta vista.

6.4.2. Collocates Preferences

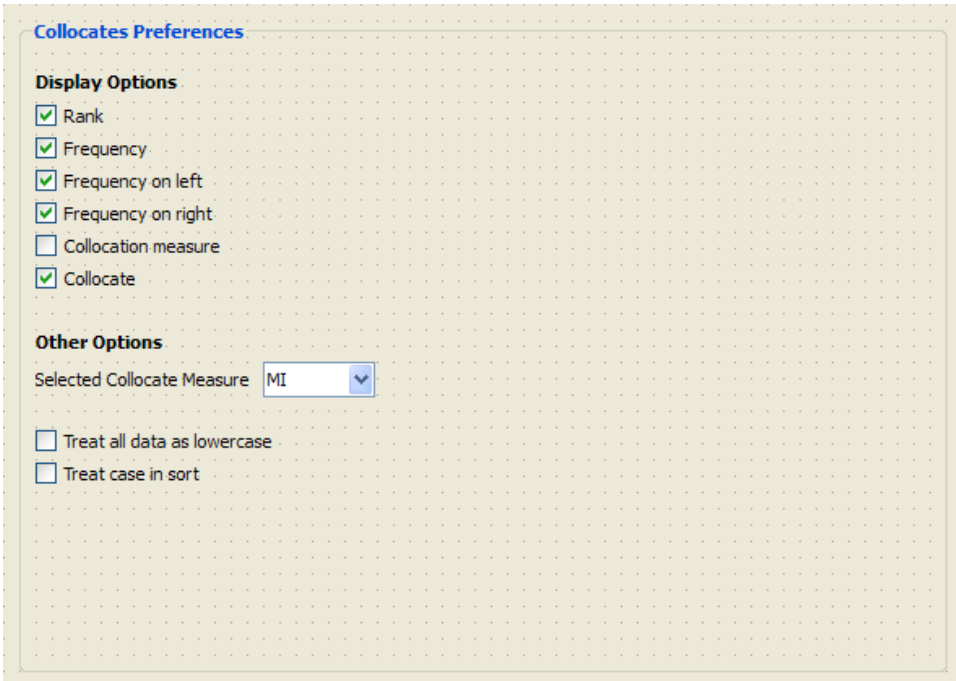

Fig. 6-J: Vista de Collocates Preferences

Es crea a partir del template Widget.

També composta per la caixa amb el títol que en aquest cas conté a més de diverses caselles un desplegable. Per a editar les opcions d'aquest s'ha d'obrir el menú contextual fent clic amb el botó dret amb el que apareix la opció d'editar-lo.

No s'han definit senyals per a aquesta vista.

6.4.3. *Word List Preferences*

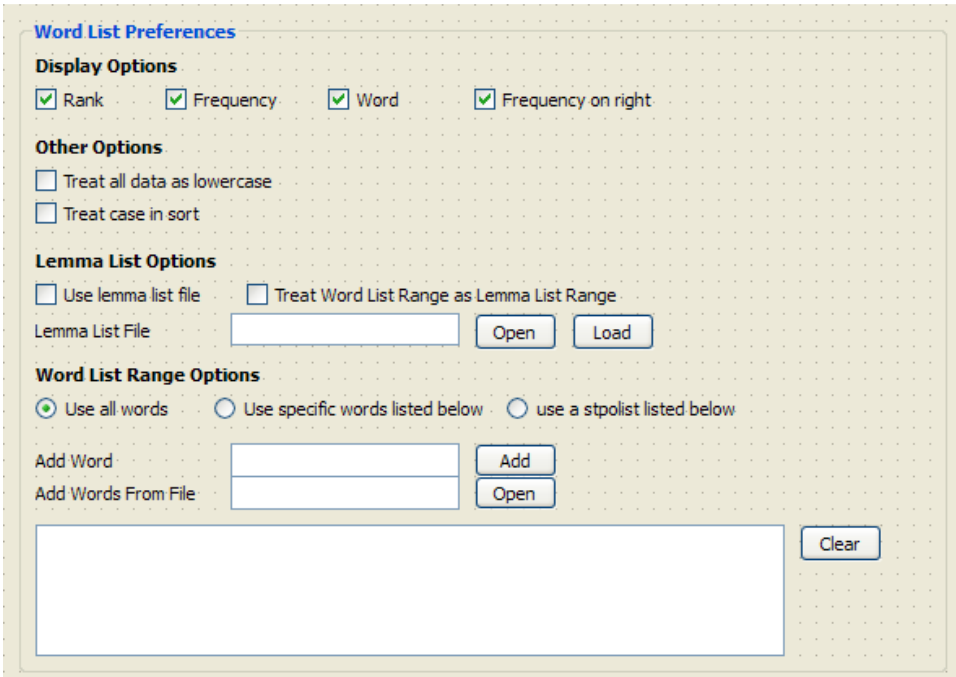

Fig. 6-K: Vista de Word List Preferences

Es crea a partir del template Widget.

Dins de la caixa amb el títol mostra diverses caselles, botons, quadres de text, etc. Insisteixo en que només està definida la vista, és a dir que els botons i la resta d'elements no tenen cap funcionalitat actualment.

No s'han definit senyals per a aquesta vista.

6.5. About

Fig. 6-L: Finestra About des del punt de vista del disseny

Es tracta d'una vista simple generada a partir del template Widget.

Conté un quadre de text on s'ha inclòs el text que es veu i s'ha canviat algunes propietats per tal que mostri aquest aspecte. Per una banda s'ha deshabilitat i tret el marc, i per altra s'ha activat la casella readOnly. Totes aquestes propietats es poden canviar mitjançant el Property Editor de Qt Designer.

No es defineixen senyals.

7. Estructures de dades i Models

Per tal de mantenir la informació més general amb la que ha de tractar l'aplicació s'ha definit un conjunt d'estructures de dates externes a les classes, a més s'ha definit un model de dades per a cada vista de llista (QListView).

7.1. Estructures de dades

Les estructures venen definides en el fitxer Dades.py. En aquest punt es recomana obrir el fitxer per veure el contingut.

Cada estructura ve donada per una classe de Python. Les estructures es detallen a continuació:

7.1.1. Class files

La classe files conté un llistat de cadenes, on cada cadena és el contingut d'un arxiu. Te diverses funcions per al manteniment de la llista. Permet afegir arxius, recuperar el contingut donant l'índex de l'arxiu, obtenir el nombre d'arxius o eliminar arxius donant l'índex.

7.1.2. Class searchWords

La classe searchWords guarda les darreres opcions de cerca. És necessària ja que si es fan canvis a la finestra Advanced Search i no es guarden, es fa clic a cancel·lar o es tanca la finestra sense acceptar-los aquests canvis no es guardaran. Per tant aquesta classe conté les opcions vàlides que es poden canviar mitjançant aquesta finestra.

A més el format dels valors o contingut dels elements de la finestra no sempre és el més apropiat per treballar amb les funcions. Per exemple la llista de paraules de Advanced Search conté un text en format string, mentre que a les funcions hem de passar una llista. El fet d'adaptar el contingut de la variable només quan aquesta canvia a un format més òptim, és a dir, en salvar els canvis a Advanced Search, evitarà haver d'estar fent canvis en cridar a cada funció.

La classe conté les següents variables:

- searchList: Llistat d'expressions a cercar.
- useSearchList: Booleà. Conté l'estat de la casella que indica si s'utilitza una llista de expressions.
- useContextWords: Booleà. Conté l'estat de la casella que indica si s'ha de comprovar el context.
- contextWords: Llistat de paraules a cercar en el context de l'expressió.

- contextFrom: Nombre de paraules a la dreta o esquerra de l'expressió on comença el context.
- contextTo: Nombre de paraules a la dreta o esquerra de l'expressió on acaba el context.

Aquesta classe no conté cap funció. Les variables son públiques i s'actualitzen i/o llegeixen directament.

7.1.3. Class kwikobj

Conté la informació de les llistes de la pestanya Concordance. Consta de dos variables, en ambdós casos llistes per emmagatzemar el contingut de les columnes. La columna Hit només mostra una llista incremental d'enters, pel que no és necessari tenir una variable que guardi aquesta informació. Les variables son les següents:

- kwikList: És una llista on cada element correspon a una línia de la columna KWIC
- fileList: Igual que l'anterior, però la correspondència és amb la columna File.

En aquest cas comptem amb les següents funcions:

- addItem: Permet afegir un nou element.
- clean: Neteja (buida) les llistes.
- count: Compta el nombre d'elements.

7.1.4. Class wlobj

Molt semblant a l'anterior i amb les mateixes funcions, tot i que ara la informació que s'emmagatzema és la de les columnes de la pestanya Word List. En aquest cas tenim la columna Rank la qual es comporta com Hit de la pestanya Concordance, pel que no fa falta tenir una variable. La resta de columnes, per les quals tenim una variable de tipus llista a la classe, son les següents:

- freqList: Llista on cada element correspon a una línia de la columna Freq.
- wordList: Llista on cada element correspon a una línia de la columna Word.
- lemmaList: Llista on cada element correspon a una línia de la columna Lemma Word Form(s). Aquesta part depèn de la càrrega d'un fitxer en la secció Word List de Tool Preferences, que com no ha estat implementada encara no funciona. Tot i així s'ha definit la llista dins la classe per facilitar la continuació del desenvolupament de l'aplicació. Actualment la variable lemma s'omple amb cadenes buides fins tenir la funcionalitat per carregar els arxius i trobar lemma.

Les funcions les obviem ja que son iguals al cas anterior.

7.2. Models

Cada vista en Qt te assignat un model, i cada columna dissenyada és una vista d'una llista (QListView) a les pestanyes Concordance i FileView en tenim varies. A més tenim la llista d'arxius que també requerirà el seu propi model.

A més s'ha inclòs en aquest apartat una altra classe que no correspon a un model, si no que serveix per aplicar un estil determinat a les vistes de la pestanya Concordance. D'aquesta manera podem veure el text com HTML enlloc de text pla, el que permetrà aplicar color a les paraules de cerca i ordenació.

Tots aquests elements venen definits a l'arxiu Models.py que es recomana obrir per veure com es defineix i de quines parts consta un model. Si es pretén continuar amb el desenvolupament de l'aplicació és molt important parar cura en aquest apartat, ja que s'haurà de repetir el que s'ha fet aquí per qualsevol nova vista a la que vulguem donar funcionalitat.

Cada model pot tenir o se li pot assignar una o varies variables o una classe de la que mostrarà informació. S'ha de definir com mostrar la informació i es pot afegir comportaments addicionals com mostrar informació en passar el ratolí per sobre de l'element que es mostra –En podem veure un exemple en el model de la columna File a la pestanya Concordance–.

Quan s'actualitza l'element al que apunta el model la vista no s'actualitza sola. És necessari reiniciar el model mitjançant una crida a la funció reset() com veurem més endavant en els capítols dedicats als controladors.

7.2.1. Models de la pestanya Concordance

- kwikKModel: És el model de la columna KWIC. Mostra la informació de la variable kwikList de la classe kwikobj.
- kwikFModel: És el model de la columna File. Mostra informació de la variable fileList de la classe kwikobj. A més mostra la mateixa informació si passem el ratolí per sobre d'algun element d'aquesta columna.
- hitModel: És el model de la columna Hit. Enlloc de mostrar contingut d'una llista mostra el número de fila. De totes maneres està lligat a la variable fileList de la classe kwikobj per tal de controlar el nombre de files a mostrar.

S'inclou aquí la classe que proporciona comportament especial als elements d'aquesta columna.

- HTMLDelegate: Recalco que no és un model. Només proporciona cert aspecte a les vistes que se li assignin – en el nostre cas les de la pestanya Concordance –. Te un comportament similar per a totes les vistes, permetent que aquestes mostrin contingut HTML. S'ha aplicat a totes ja que aplicar-ho només a la columna KWIC provocava que els ítems d'aquesta fossin més alts fent que les tres llistes es desquadressin. A més fa que la columna File mostri només el nom de l'arxiu i no

tota la ruta tal com apareix en el llistat de fitxers. En cas d'eliminar la utilització d'aquesta classe en un futur, s'hauria de configurar el model kwikFModel per que ho faci directament.

7.2.2. Models de la pestanya Word List

- WLfreqModel: És el model de la columna Freq. Mostra la informació de la variable freqList de la classe wlobj.
- WLwordModel: És el model de la columna Word. Mostra informació de la variable wordList de la classe wlobj.
- WLlemmaModel: És el model de la columna Lemma. Mostra informació de la variable lemmaList de la classe wlobj. Tot i no estar implementada la funcionalitat està implementat el model per facilitar el desenvolupament.
- WLrankModel: Enlloc de mostrar contingut d'una llista mostra el número de fila. Està lligat wordList de la classe wlobj per controlar el nombre de files a mostrar.

7.2.3. Model de la llista de fitxers

- FileListModel: Aquesta classe conté el model que s'encarregarà de controlar la llista de fitxers. A diferència dels altres, aquest conté les variables on guardarem la llista, deixant com a part externa la classe files que com ja hem dit conté el contingut dels fitxers i no altra informació d'aquests.

  La informació que guardarem en aquesta classe és el nom dels fitxers –òbviament en una llista – i un enter amb el nombre d'arxius.

  Tot i que s'emmagatzema el nom de fitxer sencer amb la ruta, el model mostrarà només el nom d'aquest, però en passar el ratolí per sobre indicarà el nom complert incloent la ruta.

  Aquesta classe inclou, a més, diverses funcions per tal de afegir o treure fitxers de la llista. Permet les següents operacions:

  - Afegir un arxiu: Utilitza la funció addFile.
  - Afegir tots els arxius d'un directori i els directoris fills: Utilitza la funció setDirPath.
  - Eliminar un arxiu: Utilitza la funció removeFile a la que se li passa la posició de l'arxiu que volem eliminar.
  - Eliminar tots els arxius: Reinicia les variables eliminant la informació que te sobre els arxius.
  - Buscar quina posició ocupa un arxiu en la llista donant el nom: Utilitza la funció findByName.

8. Controladors

Seguint amb el propòsit didàctic d'aquesta memòria s'ha canviat l'ordre de les finestres per tal de començar amb el controlador més senzill i anar incrementant la dificultat.

S'aprofita el primer apartat per explicar el funcionament del controlador a un nivell bastant simple.

8.1. Finestra About

Si obrim el fitxer Python generat a partir del fitxer que crea Qt Designer podem observar que tota la vista està inclosa en una classe anomenada Ui_About. De fet la classe tindrà com a nom "Ui_" seguit del nom que se li dona a l'objecte.

El controlador de la finestra About, així com la resta exceptuant la principal estan definits en el fitxer OtherWindowFuncs.py

La part del fitxer que ens interessa és prou curta i senzilla per incloure-la aquí i fer un estudi:

```
class About(QtGui.QWidget):
    def __init__(self):
        QtGui.QWidget.__init__(self)

        self.ventana = Ui_About()
        self.ventana.setupUi(self)

self.ventana.textEdit.setFocus()
```

La primera línia defineix la nova classe que creem, la qual com s'ha comentat en la vista hereta de la classe QWidget

Després defineix la funció a la que es crida en crear una instancia de la classe. Es crea amb una línia de l'estil:

```
x = About()
```

x conté ara una instància de la classe About.

Si volguéssim crear la instància passant paràmetres aquests es poden definir en la segona línia. Serà suficient amb incloure el nom del paràmetre darrere de self. La línia i l'assignació quedarien d'aquesta manera:

```
def __init__(self, param):
```

```
x = About(1)
```

Python permet la sobrecàrrega de funcions –com s'haurà pogut detectar en el capítol corresponent a la definició de les funcions bàsiques–, així doncs podríem tenir múltiples definicions d'__init__.

Tornant al nostre codi, el que fa en crear la instància és, en primer lloc crear al mètode __init__ de la classe pare QWIdget. Després crea una variable de la classe anomenat ventana de tipus Ui_About i crida a la funció setupUi, que si mirem l'arxiu About.Py és l'encarregat de crear tots els ítems de la finestra.

En darrer lloc posem el focus al quadre de text que conté About. Aquesta línia és un exemple de codi que s'ha afegit per tal de aplicar una funcionalitat.

Aquesta primera classe és senzilla, però com s'ha vist o veurem més endavant una classe pot heretar d'una altra, podem tenir més d'un __init__ depenent del nombre de paràmetres que es passin, podem afegir el codi per tenir funcionalitat extra a la inicialització de la instància, podem definir noves funcions o elements de la classe que es poden cridar més endavant, etc.

Aquest capítol ha servit com a introducció i és una excepció en quant a la exhaustivitat de les explicacions. Els següents es centraran més en les funcionalitats concretes de cada controlador sense copiar codi, pel que es recomana per tal de seguir les explicacions que s'obri el fitxer OtherWindowsFunc.py.

8.2. Finestra Global Settings

La classe GlobalSettings conté el controlador d'aquesta finestra, però tal i com s'ha comprovat en la vista, la vista constava de una classe extra per cada opció del menú. Per tal de controlar les classes corresponents a les vistes filles, necessitarem per tant tenir una classe en el controlador per a cada una.

Per tant, dins de la classe GlobalSettings podem observar com s'inclouen totes les altres classes, on en els les crides a__init__ podem observar com permet passar el paràmetre parent.

En inicialitzar la classe es crea la finestra Global Settings i acte seguit es crea una per cadascuna de les altres vistes, i en el moment de la creació es passa com a paràmetre el quadre que hi ha a la part dreta de la finestra Global Settings per tal de que les altres vistes es carreguin en aquest espai i no com finestres independents.

Per tant la instància de la classe GlobalSettings conté instàncies a les classes de les vistes filles, a les quals podrem accedir fàcilment.

Després de crear controladors de totes les vistes, amaga aquelles que no son necessàries, és a dir totes excepte la de File Settings que és la opció que apareix remarcada per defecte en el llistat, i per tant la vista que ha d'aparèixer al quadre.

Finalment crea una connexió que farà que la senyal currentRowChanged(int) de la llista cridi a la funció changed de la classe.

Així doncs només ens queda definir aquesta funció que mostrarà una de les vistes depenent del valor que li passi la senyal.

La funció es pot veure definida a continuació i no te major complexitat per entretenir-nos a comentar-la.

En conclusió, podem veure que la única funcionalitat implementada és el canvi de vista depenent de la selecció que es faci en la llista.

8.3. Finestra Tool Preferences

El controlador de la finestra Tool Preferences s'ha resolt completament igual que el de la finestra Global Settings canviant les vistes que es carreguen.

Tampoc hi ha cap funcionalitat implementada a part de la navegació per les diferents vistes utilitzant la llista que serveix com menú en el costat esquerre, pel que no s'allargarà més aquest capítol.

8.4. Finestra Advanced Search

La finestra Advanced Search comença a ser complicada, conté varies senyals, així com diverses funcions i variables. A més, redefineix el comportament d'algun dels seus components.

En primer lloc es crea la classe passant com a paràmetre la classe que la crida. La classe AdvancedSearch inclou variables pròpies utilitzades per al control dels components que es vol que actuïn de manera especial. S'utilitzarà una altra variable que contindrà el document que tindrem en el quadre de text per al llistat d'expressions a cercar.

Després de crear la interfície assignem com a model del desplegable Search Term el mateix que utilitza el desplegable a la pantalla principal, ja que ambdós desplegables han de mostrar el mateix contingut.

A més definim que aquest desplegable afegeixi les noves insercions al inici de la llista enlloc de afegir-les al final.

Definim que el llistat d'expressions no talli les línies mitjançant la funció setLineWrapMode d'aquest component.

Tot seguit es defineixen les connexions entre senyals i funcions, les quals venen detallades en el llistat i no detallaré. Per tant passo directament a explicar les funcions:

- **clearSearchList**: Neteja el quadre

- **apply**: Funció cridada en clicar el botó apply. Passa el valors en els components de la finestra a les variables de la classe searchWords – ja explicada en l'apartat d'estructures –.
  Per a la variable searchList converteix el text de la finestra en una llista partint-lo en trobar un salt de línia.
  Practica una conversió de contextFrom i contextTo amb el format ja explicat a un enter, essent aquest negatiu si la lletra que segueix el nombre és L.
  Un cop assignats tots els valors tanca la finestra.

- **loadAdvancedFile**: Carrega un arxiu fent servir una funció de la classe QFileDialog per a aquest fi. Passa el text d'aquest document al document creat per al quadre de text de les expressions a cercar.

- **addContextWord**: Llegeix la línia on s'ha escrit una nova paraula i si aquesta és una paraula l'afegeix a la llista. Si no ho és mostra un missatge.

- **manageCFrom** i **manageCTo**: Funcions per a modificar el comportament dels spinbox de Context Horizon. Aquests mostren sempre valors positius seguits d'una L o una R depenent si la paraula està a la dreta o esquerra de l'expressió cercada o res si és la mateixa expressió. Per tal d'aconseguir imitar aquest comportament s'utilitza una variable a la classe que contindrà el valor real assignant valors negatius quan estiguem en el costat esquerre i tinguem, per tant, el sufix L.
  En canviar el valor sumarà o restarà 1 a aquesta variable i farà la transformació passant com a valor del component el valor absolut de la variable i afegint el sufix corresponent depenent si el valor és zero, positiu o negatiu.

  Després de tot el que he treballat amb Python i PyQt me n'adono de que la redefinició de components tal i com la he fet no és la manera més òptima. De fet s'hauria de crear una classe que heretés del component que es vol canviar – QSpinBox en el meu cas– amb les funcionalitats requerides i una variable pròpia per controlar el valor. D'aquesta manera no seria necessari tenir una funció i una variable per cada component que tingui aquest comportament.

8.5. Finestra Principal

La finestra principal de l'aplicació és la que te més dificultat ja que inclou totes les funcionalitats de l'aplicació, per aquesta raó s'ha dedicat un arxiu sencer separant-la de les altres finestres. El codi en aquest arxiu està comentat molt més exhaustivament que en els altres per tal de facilitar-ne la comprensió del que fa cada part del codi.

L'arxiu que s'utilitza és MainWindowFuncs.py. Es recomana que s'obri per a fer el seguiment a mida que avancen les explicacions.

8.5.1. Creació de la finestra

La primera diferència que podem apreciar amb respecte a les altres finestres consisteix en que la classe on implementem el controlador hereta de QMainWindow i no de QWidget com ha passat fins ara. En aquest cas la finestra es passa com a paràmetre en la definició de la classe i no s'assigna a una variable com ha passat en els casos anteriors.

Per tant per cridar a la finestra no és necessari accedir a una variable com en el cas dels altres controladors, si no que la mateixa classe –i per tant utilitzarem self per accedir-hi– conté la finestra.

Les primeres línies són similars a les que hem vist fins ara. Fan una crida a la inicialització de la classe pare i a la funció setupUi de la classe Ui_MainWindow on hi ha el contingut de la finestra.

A continuació creem instàncies de les diferents finestres tal i com s'ha mostrat a la creació del controlador de la finestra About. Advanced Search es crea passant com a paràmetre la finestra principal, mentre que totes les altres no passen paràmetres.

A continuació creem variables per a controlar les dades dels spinbox que han de tenir un comportament no habitual tal i com hem fet en la finestra Advanced Search. De nou insisteixo en que la manera més optima de realitzar això hauria estat crear una classe que heretés de QSpinBox que contingués la variable i definís el comportament.

Es generen diverses variables de les quals s'explica la funció al codi. Tot i així m'agradaria explicar aquí el funcionament de la variable results, la qual és un llistat de resultats en fer una cerca i cada resultat conté la següent informació:

- Enter amb la posició de la concordança a l'arxiu.
- Expressió trobada: Això és útil en cas de que la cerca sigui sobre un llistat d'expressions.
- Nom de l'arxiu: Necessari en cas de treballar amb més d'un arxiu.
- Línia original complerta: Necessària ja que en mostrar-la afegim codi HTML per a la coloració i per tant perdem la informació sobre la línia real, els índex ja no son els mateixos, etc. Aquesta línia te la grandària que defineix el valor a Search Windows Size.
- Enter amb la posició de la concordança a la línia

- Número de concordança de entre les trobades a l'arxiu. Necessari per a marcar la concordança correcta en File View quan cliquem una línia en la pestanya Concordance
- Tupla amb la posició inicial i final de la paraula a la posició que marca el nivell 1 de l'ordenació.
- Tupla amb la posició inicial i final de la paraula a la posició que marca el nivell 2 de l'ordenació.
- Tupla amb la posició inicial i final de la paraula a la posició que marca el nivell 3 de l'ordenació.

A continuació es generen els models i s'assignen als elements. Després apliquem l'estil definit a HTMLDelegate a les llistes de la pestanya Concordance.

Seguidament definim l'ample inicial de les diferents columnes o llistes mitjançant els splitters que contenen aquestes. Un cop oberta l'aplicació podem canviar aquest ample amb el ratolí arrossegant l'espai entre dues columnes.

Tal i com hem fet a la finestra Advanced Search canviem el comportament del desplegable searchTerm per tal que incorpori els nous ítems al final de la llista i no al final com és el comportament normal d'aquesta classe.

En el capítol dedicat a les vistes explicàvem el funcionament d'algunes pestanyes que contenien conjunts de llistes degut a que la utilització d'una taula no permetia el comportament desitjat, així doncs és el moment de definir aquest comportament sobre el desplaçament vertical de les llistes.

Com es pot observar al codi el procediment consisteix en definir com a barra de desplaçament d'una llista la d'una altra. En aquest punt és molt important posar atenció a l'ordre que es segueix per a realitzar aquesta assignació. Hem d'assignar a la segona llista començant per l'esquerra la barra de la primera, a la tercera la de la segona, i així successivament fins a arribar a la darrera llista del grup.

S'ha realitzat això per les pestanyes actives, és a dir Concordance i Word List, pel que si s'activa Collocates serà necessari fer això mateix amb les llistes de la pestanya.

A la vista de la finestra principal només s'ha definit el menú File en la barra de menús, ja que Qt Designer només permetia afegir menús a la barra i accions i submenús als menús, mentre que nosaltres volem tenir accions directament sobre la barra, així que es defineixen a continuació.

Els botons que s'afegiran com accions son Global Settings, Tool Preferences i About.

Acte seguit es genera la llista de connectors que es pot consultar al codi, pel que no entraré en detalls. Alguns d'aquests no actuen directament sobre senyals d'elements de la finestra principal, si no que responen a senyals d'altres finestres. Tenir-los aquí permet aprofitar funcions.

A continuació connecta la senyal numberPopulated del model FileListModel amb la funció setFileNum per tal d'actualitzar aquest quadre quan canviï el nombre d'arxius carregats i per tant aquesta senyal sigui cridada.

Després amaguem algunes parts de la zona del formulari que no es mostraran, tot i que es podria haver fet més fàcilment cridant a la funció que s'utilitza en canviar la pestanya i passant com a paràmetre el valor 0, ja que estem a la primera pestanya.

Definim una variable on emmagatzemar el cursor que s'utilitzarà per a marcar paraules al quadre de text de la pestanya File View.

Per últim es desactiven les pestanyes no implementades.

8.5.2. Control de pestanyes

Com s'ha explicat en la vista, el formulari te diversos elements, alguns dels quals hem agrupat en quadres per tal de facilitar mostrar-los o amagar-los. Això s'ha de fer quan canviem de pestanya, i s'encarrega la funció **manageKwicSort** que s'ha connectat en la inicialització de la classe amb la senyal que envia el widget que conté les pestanyes en canviar d'una a l'altra.

El que fa aquesta funció és simple: Comprova quina és la pestanya activa i mostra o amaga els diferents elements.

8.5.3. Control (customització) dels spinbox

Com passava en la finestra Advanced Search, la finestra principal té també alguns elements que hem de personalitzar. De nou insisteixo que la implementació d'una classe hauria estat millor solució, però degut al desconeixement inicial de les eines es va fer d'aquesta manera. La posterior falta de temps ha provocat que no es canviés més endavant, ja que hi ha hagut altres prioritats.

En aquest cas tenim dos tipus diferents de comportament. El primer és igual al de Advanced Search, i per tant no entrarem. El segon cas correspon a la pestanya Concordance Plot. L'element zoom d'aquesta vista hauria de tenir un increment exponencial, i només conté valors que siguin potència de 2. Qt Designer permet definir el pas d'un spinbox, però en aquest cas el pas ha d'anar canviant depenent del valor que tinguem, i per tant aquest comportament ha de ser definit aquí.

Com en els altres casos s'ha utilitzat una variable per tal de portar un control del valor real.

S'ha hagut d'utilitzar un total de quatre funcions i variables, una per cada spinbox a modificar, mentre que amb la utilització de classes hauria estat suficient amb un parell. Una per definir cada comportament.

8.5.4. Control de fitxers

Aquest apartat conté totes les funcions que s'utilitzen per a controlar els fitxers.

- **openFile**: Permet obrir un o més arxius. Crida a una funció de QFileDialog que s'encarrega d'obrir el quadre de diàleg estàndard de Windows i retorna un llistat amb la selecció. Per cada element de la llista cridem a la funció addFile del model.

- **openDir**: Té una funcionalitat similar al cas anterior, tot i que ara la funció de QFileDialog permet seleccionar un directori enlloc de fitxers. Si el que retorna la funció no és una cadena buida, és a dir que hi ha un directori seleccionat i s'ha clicat acceptar, es crida a la funció setDirPath del model que ja s'encarrega de la recursivitat si hi ha directoris fills.

- **closeFile**: Elimina els fitxers seleccionats. Crea una llista amb els fitxers seleccionats i per cada element crida a la funció removeFile del model. Aquest s'encarrega d'eliminar els continguts de l'estructura files.

- **closeAllFiles**: Elimina tots els arxius de la llista cridant a la funció del model removeAllFiles.

- **setFileNum**: Posa el valor que rep al quadre Total No. Aquesta funció està connectada amb la senyal que llença el model en afegir o eliminar elements.

Les següents funcions no pertanyen directament al grup de control dels fitxers, però estan relacionades ja que funcionen en seleccionar-ne un i la visualització d'aquest en la pestanya File View.

- **fileClicked**: Carrega el contingut d'un arxiu en el quadre de text de la pestanya File View quan un arxiu és clicat. Marca en blau les concordances i selecciona la primera.

- **fileSetCursor**: Situa el cursor en la n-èssima concordança, on n és el paràmetre que ha de rebre la funció.

8.5.5. Control de finestres

Aquest apartat detalla les funcions que obren finestres i contenen diverses funcionalitats relacionades amb la inicialització.

- **about**: Mostra la finestra About.

- **globalSettings**: Mostra la finestra Global Settings, abans, però, mostra un missatge avisant de que no funciona.

- **toolPreferences**: Obre la finestra Tool Preferences amb el mateix missatge que en el cas anterior.

- **advancedSearch**: Obre la finestra Advanced, però abans carrega els valors que ha de contenir i que es guarden a la estructura searchWords. Així en cas de que la finestra s'hagi tancat anteriorment sense guardar els canvis, aquests canvis son eliminats i carreguem els darrers canvis vàlids en tornar-la a obrir.

8.5.6. Accions en clicar llistats

Alguns llistats tenen un comportament especial quan una línia és clicada. Aquests necessiten tenir implementades unes funcionalitats específiques per tal de poder realitzar el conjunt d'accions que donen lloc a aquest comportament.

- **concKwikListclicked**: En clicar una línia de la columna KWIC a la pestanya Concordance l'aplicació ens porta a la pestanya File View, on mostra el fitxer i ressalta la concordança.
  Aquesta funció obté la primera coincidència en el model que controla els fitxers d'un amb el nom del fitxer on s'ha trobat la concordança clicada.
  Acte seguit carrega aquest fitxer en el quadre de la pestanya FileView, obre aquesta pestanya, i ressalta la expressió de la concordança.

- **WLwordListclicked**: En clicar a una paraula a la columna Word de la pestanya File View, es realitza una cerca de la paraula a la pestanya Concordance.
  La funcionalitat d'aquesta funció és fàcil. Primer afegeix la paraula a la llista i la selecciona. Després desactiva la casella "Use search term(s) from list below" de la finestra Advanced Search, així força a utilitzar només la paraula seleccionada.
  Obre la pestanya Concordance, i realitza la cerca.

8.5.7. Accions dels botons

Son les funcions assignades als botons Start, Stop i Sort, tot i que Stop està deshabilitat.

- **start**: S'utilitza la mateixa funció per a totes les pestanyes, però realitza algunes accions depenent de les condicions. Separo el contingut per l'activitat genèrica i per cada pestanya:
    - Activitats genèriques: Sempre es realitzen. Posa la barra de progrés a zero, afegeix la nova cerca al desplegable i la selecciona i prepara les llistes de paraules per a la cerca i de context per a poder-les utilitzar amb les funcions.
    - File View: La condició per que s'executi aquesta part és que hi hagi un sol arxiu seleccionat, i és independent de la pestanya oberta. Carrega l'arxiu seleccionat ja que en fer la càrrega es realitzarà la nova cerca.
    - Concordance: Aquesta part s'executa només si la pestanya Concordance està activa. Comença buidant l'estructura on apunten els models de les columnes, així com la variable results.

      Per cada arxiu incrementa parcialment la barra de progrés, va cridant a la funció bàsica searchTerms i guarda els resultats a la variable results local a la funció. Tot i compartir nom amb la variable de la classe, son diferents.

      A continuació, per cada resultat obté la línia original, una altra pintant-la, afegeix a les variables a les quals apunten els models de les llistes la informació a mostrar i concatena a cada resultat el nom de l'arxiu, la línia original i un enter que va incrementant per definir el número de concordança dins del fitxer.

      A més actualitza el quadre Concordance Hits amb el nombre de concordances trobades per tal de que es vagi incrementant per cada arxiu i no solament ho indiqui al final.

      En acabar el bucle copia el llistat results local a la variable results de la classe. Reinicia els models afectats, posa la barra de progrés a 100, i actualitza el text de Concordance Hits amb el valor final.
    - Word List: Només executa aquesta part si la pestanya Word List està activa. Comença com l'anterior buidant l'estructura on apunten els models de les columnes.

      Després obté els valors de que haurà de passar a la funció wordList, els quals tenim en dos caselles i un desplegable i que defineixen l'ordenació de les paraules, així com la diferenciació o no entre majúscules i minúscules.

      Crida a la funció a la qual passa els paràmetres que s'han preparat prèviament. Si aquesta retorna resultats omple la classe a la qual apunten els models de les columnes d'aquesta pestanya.

      Omple els espais per text de la zona superior de la pestanya. Com que la cerca de coincidències no està implementada en aquesta pestanya els Hits sempre seran zero.

      Acte seguit reinicia els models afectats i posa la barra de progrés a 100.

- **sort**: És la funció encarregada de la ordenació a la pestanya Concordance.
  Obté la llista de resultats i els valors dels tres camps sota Kwic Sort que contenen les posicions dels tres nivells per a l'ordenació.
  Utilitzant la funció sortWList amb quatre paràmetres obté un llistat d'enters que detalla el nou ordre que s'haurà d'aplicar, el llistat es guarda a la variable order. Aquesta variable serveix com a índex, i per tant podem accedir a la línia n dels resultats ordenats accedint a la línia que està en la posició donada per l'enter order[n].
  A continuació, es copien els valors de la classe a la que apunten els models afectats, així com la variable result de la classe, a variables locals amb el mateix nom. Buidem les variables i reiniciem els models. Això permetrà reinserir tota la informació de manera ordenada.
  Anem accedint als elements de forma ordenada utilitzant com a index order[n].
  Per cada element a results obtenim les posicions inicial i final de cadascuna de les paraules que han afectat a la ordenació, i les afegim a result per tal de poder-les marcar en color al mostrar la línia.
  Pintem la línia i l'afegim a l'objecte kwikobj, al qual apunta el model de la columna Kwic. Afegim també el nom del fitxer i finalment afegim la línia a la variable resultat de la classe.
  Finalment reiniciem els models de la pestanya Concordance per tal de que mostrin la informació actualitzada.

- **stop**: És la funció que hauria de parar una cerca o una ordenació.
  Com el procés que segueix l'aplicació és lineal no es pot polsar fins que ha acabat la cerca u ordenació, pel que és un botó que no te utilitat en la nostra aplicació.
  Per tal de que aquest pugui ser utilitzar amb la finalitat que suposadament té, hauríem d'incorporar l'ús de threadings.
  De totes maneres el temps de resposta en aquesta aplicació és millor que el que proporciona AntConc, però això no justifica no implementar el botó.

8.5.8. Control de Advanced Search

La finestra Advanced Search te certs components relacionats amb altres a la finestra principal. Les accions que es fan sobre uns s'han de reflectir en els altres, això s'implementa en el grup de funcions que es descriuen a continuació. Amb aquestes funcions, a més es permet utilitzar la finestra principal tenint Advanced Search oberta, ja que les accions en els elements comuns es repliquen. AntConc no permet aquest comportament, que deixa la finestra principal deshabilitada mentre està oberta Advanced Search.

- **searchTermSelect**: En seleccionar una expressió en el desplegable Search Term en una de les dues finestres, aquest s'ha de seleccionar també en l'altra. Senzillament el que es fa és seleccionar en ambdues l'element en la posició que es rep com a paràmetre.
- **wordCheck**, **caseCheck** i **regexCheck**: El mateix passa amb les caselles. En seleccionar-ne una en una finestra s'ha de seleccionar la mateixa en l'altra, i el mateix al desseleccionar-la. Tenim una funció per cada casella, que, com en el cas anterior activaran o desactivaran les dues segons el paràmetre rebut. Les funcions són pràcticament idèntiques entre si, amb la diferència de que en activar la casella Regex es deshabiliten les altres.

8.5.9. Control d'esdeveniments

Aquesta secció defineix les funcions associades a esdeveniments de la finestra. Actualment només tenim una funció, **closeEvent**, a la qual crida la finestra en tancar-se. D'aquesta manera podem mostrar un missatge per a preguntar a l'usuari si vol tancar, i en cas de que accepti es tanquen les finestres obertes.

9. Main

Aquest capítol pretén explicar el funcionament de l'arxiu pfc.pyw i mostrar la funció main que fa una crida a la finestra principal i com aquesta crida tots els altres arxius.

Tal i com podem veure en l'arxiu, la funció és simple. Es crea l'aplicació i una instància de MainWindow, i aquesta es mostra directament. No es passa cap paràmetre ni hi ha cap complicació.

Així doncs, ja hem acabat d'explicar Main, el que ara ens quedaria és establir un diagrama de qui crida cada element:

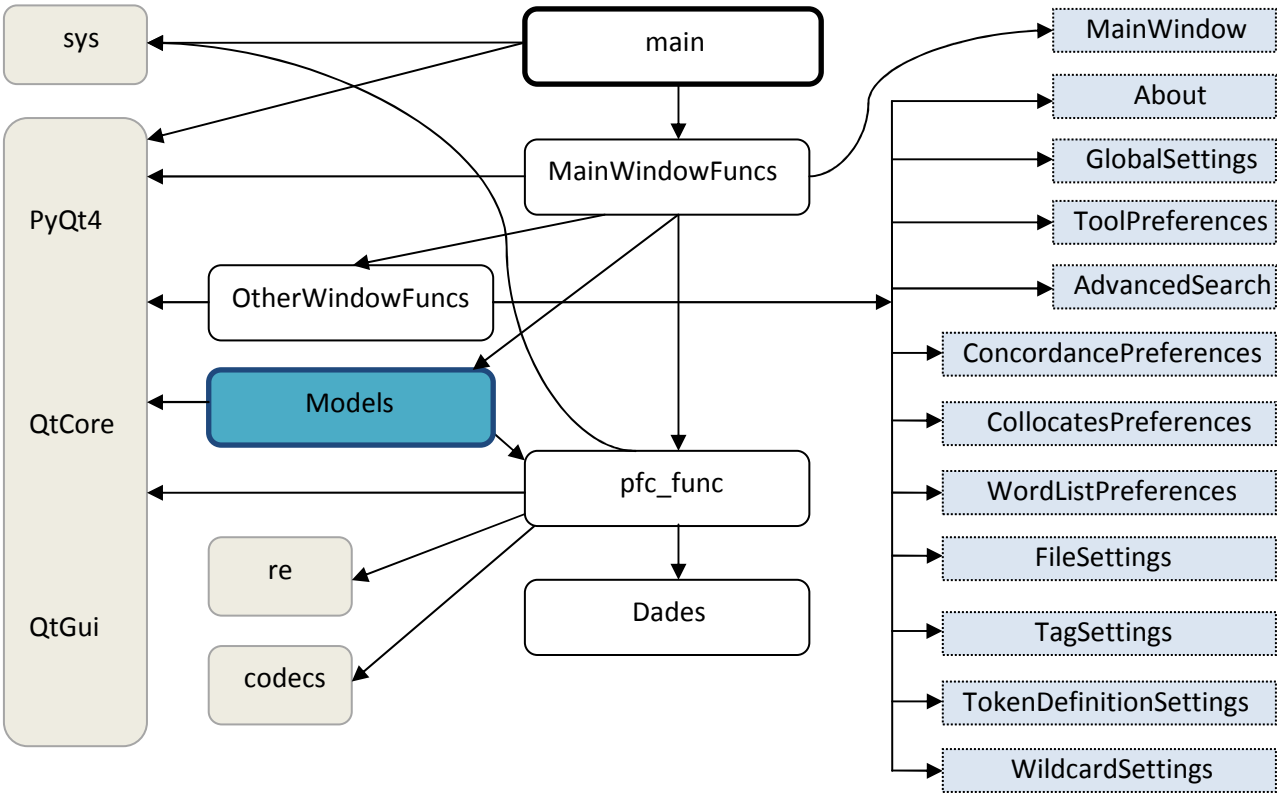

Fig. 9-A: Diagrama de components

10.Com continuar

La intenció d'aquest capítol és facilitar a altres desenvolupadors la continuació de l'aplicació i l'ampliació d'aquesta. Per tal de poder seguir, el primer pas seria tenir localitzat en quin lloc s'han deshabilitat certes opcions per tal de poder-les tornar a activar. No és necessari activar-les fins que es necessitin, però és important saber com fer-ho. Després es proporcionarà un llistat de les funcionalitats que queden per definir i s'aportarà idees.

En aquest punt ja es suposa que les eines per a poder desenvolupar ja estan instal·lades en l'equip, si no, hi ha informació al respecte al tema d'Introducció a les Eines.

Seria important per al desenvolupament canviar l'extensió de l'arxiu pfc.pyw a py. La diferència és que un .pyw no requereix d'una finestra de DOS per executar-se, mentre que un arxiu .py sí. D'aquesta manera obtindrem els missatges d'error així com qualsevol altra cosa que vulguem imprimir amb print, el que ajudarà al desenvolupament i depuració d'errors.

10.1. Habilitar els elements desactivats

Aquest apartat s'estructura seguint les diferents capes de l'aplicació, i per tant els primers són aquells elements que es poden reactivar a Qt Designer. Per tal d'accedir a aquests primers elements obrim MainWindow.ui.

Accions del menú File: Despleguem el menú File de la finestra principal de UOCConc en Qt Designer i allà podem comprovar com la major part de elements estan deshabilitats. S'ha d'anar seleccionant cada element i en Property Editor marcar la casella enabled.

Botó Stop: També està desactivat en la vista. Es reactiva seleccionant-lo i marcant la mateixa casella que en el cas anterior.

La resta de elements a reactivar es troben a la capa controlador, concretament en la classe MainWindow, així que obrim el fitxer MainWindowFuncs.py en un editor.

Pestanyes deshabilitades (Concordance Plot i Collocates): S'han deshabilitat en la inicialització de la classe MainWindow, si anem al final de la definició de __init__ les últimes línies s'encarreguen de la desactivació d'aquestes dues pestanyes, pel que serà suficient amb eliminar-les o comentar-les.

Missatges en obrir Global Settings i Tool Preferències: Els missatges que veiem en obrir aquestes finestres estan implementats en les funcions que s'encarreguen d'obrir-les. Estan en la zona de Control de Finestres de la classe MainWindow. S'haurà d'eliminar o comentar la línia que obre la finestra.

10.2. Funcionalitats per definir

Aquest apartat no cerca només fer un llistat de coses a fer per continuar amb el desenvolupament si no que s'intenta donar unes pautes per a la implementació de les funcionalitats necessàries.

En primer lloc és important saber el punt en que està el projecte i la dificultat tenint el compte el que ja s'ha fet que pot tenir cada una de les parts.

10.2.1. Finalització de la pestanya Word List

La darrera part implementada va estar la pestanya Word List, i és per això que no està acabada. Queda que les paraules incloses en la cerca quedin marcades en blau, així com la navegació amb Hit Location i la funcionalitat de Search Only.

La manera més senzilla (i menys correcta) d'implementar la part de pintar les paraules que concordin amb la cerca consisteix a utilitzar HTML com he fet en la pestanya Concordance. El problema a que porta això i del que em vaig adonar més tard és que no podem treballar amb codi HTML quan AntConc si que ho permet.

Una altra manera, més complicada però molt millor, és implementar l'assignació de color tal i com s'ha fet en la pestanya File View. Aquí s'utilitzen múltiples cursors per marcar parts del text i aplicar un format especial a aquesta selecció. Això requereix bastant estudi de les classes que ofereix PyQt, però a la llarga val la pena.

El fet de comprovar si una paraula es concordant amb l'expressió de cerca no és difícil, el que permetrà generar una llista de tuples on cada coincidència es pot guardar com l'índex que te aquesta paraula a la llista, la posició inicial en la paraula on comença la coincidència i la posició on acaba. Amb això ha de ser fàcil implementar la navegació.

El botó Search Only no ha de tenir major complexitat un cop implementada tota la resta.

10.2.2. Modificació en la pestanya Concordance

Si en l'apartat anterior s'ha optat per implementar l'assignació de color tal com ho fa File View, proposo implementar el canvi aquí eliminant HTMLDelegate. Amb aquest canvi l'aplicació ja hauria de poder tractar arxius HTML i XML.

10.2.3. Pestanya Collocates

La pestanya Collocates és semblant a Word List, pel que es pot prendre aquesta com a model per a implementar les funcionalitats.

S'ha d'implementar també el model i l'estructura a la qual apuntarà i preparar el comportament de les llistes.

Per a la implementació de les funcions és important tenir molt clar que fan les funcions bàsiques i el que podem fer amb elles. Per exemple podem utilitzar wordsInRange (i de fet s'utilitza) per trobar la paraula en una posició donada amb respecte a un punt si contextFrom i contextTo es passen amb la mateixa informació.

10.2.4. Pestanya Concordance Plot

No puc donar consells sobre el contingut d'aquesta pestanya, però no sembla fàcil d'implementar.

Per una banda s'ha d'aconseguir fer el gràfic. Aquesta és la part fàcil. La part difícil ve quan hem d'aconseguir que clicant certes parts ens porti a la concordança en l'arxiu.

Si pot ajudar PyQt proporciona suport per a les senyals genèriques i podria retornar les coordenades on s'ha clicat.

Així que si no es coneix a priori d'alguna eina que pugui facilitar la feina aquest apartat requerirà bastant estudi.

10.2.5. Botó Stop (threading)

Per tal de poder utilitzar el botó Stop és necessari executar start i sort en un thread. Actualment la funcionalitat dels threads a Python és bastant pobra i no permet ni tan sols acabar amb un thread des del pare.

Per sort PyQt implementa la classe QThread que ens pot ser útil. Cercant a Internet es pot trobar bastant informació per ajudar amb aquest tema.

S'haurà de vigilar a no tenir altres threads funcionant quan llancem un o toquem alguna variable, ja que podem obtenir resultats inesperats.

10.2.6. Finestres de configuració

Amb finestres de configuració ens referim a Global Settings i Tool Preferences.

Per cadascuna de les finestres s'hauria de fer un anàlisi de quines configuracions utilitzar i quines no, ja que permetre que certs elements siguin configurables pot implicar tenir que redefinir funcions i classes. Per tant recomanaria anar afegint les opcions poc a poc i anar fent els canvis a mida que sigui necessari.

Per tal de mantenir les configuracions proposaria l'ús d'una o més classes en l'arxiu Dades.py, creant-la com a estructura.

10.2.7. Arreglar el codi

Tot i que no és una prioritat a llarg termini tenir el codi ben estructurat pot facilitar la feina, pel que proposo estructurar el que ja existeix, resultat de la codificació d'un inexpert en les eines.

El primer punt seria implementar classes per als spinbox amb comportament especial i cridar aquesta classe cada cop que es genera un component d'aquest tipus.

Per altra banda es pot netejar codi creant funcions que evitin les repeticions, o en el cas de la inicialització de MainWindow, aprofitar funcions ja existents com passa en seleccionar els elements del formulari que volem mostrar i amagar. Una crida a la funció que tracta el canvi de pestanya hauria estat suficient.

11.Relació d'arxius lliurats

Junt amb aquesta memòria es lliuren els següents arxius:

- pubach_files.zip: Conté els arxius del projecte.
- pubach_manuals.zip: Conté manuals útils per a entendre el funcionament de PyQt que serviran com a referència a qui continuï el projecte.

Els arxius del projecte continguts en el zip segueixen aquesta estructura:

- /PFC_pubach/
    - pfc.pyw
    - About.py
    - AdvancedSearch.py
    - CollocatesPreferences.py
    - ConcordancePreferences.py
    - Dades.py
    - FileSettings.py
    - GlobalSettings.py
    - MainWindow.py
    - MainWindowFuncs.py
    - Models.py
    - OtherWindowFuncs.py
    - pfc_func.py
    - TagSettings.py
    - TokenDefinitionSettings.py
    - ToolPreferences.py
    - WildcardSettings.py
    - WordListPreferences.py
    - UOC1.ico
    - /PFC_pubach/Designer/
        - About.ui
        - AdvancedSearch.ui
        - CollocatesPreferences.ui
        - ConcordancePreferences.ui
        - FileSettings.ui
        - GlobalSettings.ui
        - MainWindow.ui
        - TagSettings.ui
        - TokenDefinitionSettings.ui
        - ToolPreferences.ui
        - WildcardSettings.ui

        - WordListPreferences.ui
    - / PFC_pubach/test/
        - TEXT1.TXT
        - TEXT2.TXT
        - TEXT3.TXT

El fitxer pubach_manuals.zip conté els següent:

- PyQt4.pdf

Es tracta d'un tutorial senzill que serveix com a introducció pràctica a PyQt.

12.Conclusions

El projecte m'ha permès conèixer les funcionalitats bàsiques que ha de tenir un analitzador de concordances, tot i així, no ha permès endinsar-me gaire en el camp de la representació del coneixement i del raonament per a obtenir conclusions.

Tot i així ha servit per a descobrir noves eines per a la programació tant funcional com gràfica.

Per altra banda, el concepte de continuïtat de l'aplicació que te el projecte m'ha servit per estructurar el codi d'una manera comprensible i fàcilment modificable i ampliable per a altres desenvolupadors, pel que considero que les facilitats aportades per continuar el projecte compensen el fet de que no hagi estat possible incloure totes les funcionalitats que es pretenia en un primer moment.

Durant el procés del projecte he adquirit una gran comprensió de les necessitats que pugui tenir qui hagi de continuar el projecte, pel que el resultat final no és solament una base tecnològica sobre la qual es pot seguir treballant, si no que és, a més, una base didàctica per que qui continuï pugui estalviar el temps que a mi m'ha costat buscar informació i comprendre el funcionament, així com evitar errors que ja s'han comés.

13.Bibliografia

Jesse Padilla Agudelo
Interfaces Gráficas de Usuario en Python: Primeros pasos en PyQt4
Document trobat a Internet. S'adjunta a la memòria a l'arxiu PyQt4.pdf

Mark Summerfield – Prentice Hall (2007)
Rapid GUI Programming with Python and Qt. **The Definitive Guide to PyQt Programming.**

PyQt4 Reference Guide
http://www.riverbankcomputing.co.uk/static/Docs/PyQt4/pyqt4ref.html

PyQt4 Class Reference
http://www.riverbankcomputing.co.uk/static/Docs/PyQt4/html/classes.html

A més s'ha utilitzat el manual que s'instal·la amb Python que és molt extens i molt útil gràcies a la cerca. També s'han fet cerques concretes a Internet per a resoldre problemes.

Tot i que per començar amb Python vaig utilitzar manuals i tutorials en línia, no faig referència a cap ja que eren per la versió 2 i em van portar més maldecaps que ajuda. Tot i així van servir per fer-me una idea del llenguatge.