

El Corrector, corrector ortogràfic i gramatical de català

Descripció general de l'arquitectura

Versió 1.5

GLiCom, Grup de Lingüística Computacional
Universitat Pompeu Fabra

Grup de Veu i Llenguatge
Barcelona Media Centre d'Innovació



22 Barcelona
Media

El Corrector, descripció general de l'arquitectura

Autor	Data i comentaris
FB	06/03/2006: conclusions reunió sobre arquitectura
	08/03/2006: esborrany sobre els objectes (1.1)
	17/03/2006: simplificació i reestructuració (1.2)
MQ	22/03/2006: revisió
MQ	26/05/2006: actualització de noms de mòduls, afegir taules d'estat de desenvolupament, etc. (1.3)
MQ	15/06/2006: actualització de l'estat de desenvolupament per a cada mòdul.
MQ	10/01/2007: actualització de la introducció a l'estat final de l'arquitectura i mòduls, proposta d'esquema per a la redacció de la documentació final de cada mòdul.
FB	10/01/2007: documentació de CotigBreaker
MQ	16/01/2007: documentació per a CotigDict, CotigLabeler, CotigGram i CotigSpell.
MQ	19/01/2007: documentació de CotigShared i CotigMulti.
MQ	26/01/2007: actualitzacions d'apartats relatius a CotigMulti i CotigChooser. Revisió general. (1.5)

1 Taula de continguts

1 Taula de continguts.....	3
2 Presentació general.....	5
2.1 Funcionalitats d'El Corrector.....	5
2.1.1 Definició lingüística d'El Corrector.....	5
2.1.2 Tipologia d'errors tractats.....	6
2.2 Arquitectura.....	6
3 Descripció de les llibreries DLL.....	9
3.1 CotigMain: motor de correcció.....	9
3.1.1 Tasca.....	9
3.1.2 Estratègia d'implementació.....	9
3.1.3 Interfície de mòdul: ICotigMain.....	9
3.2 CotigBreaker: segmentació del text.....	11
3.2.1 Tasca.....	11
3.2.2 Estratègia d'implementació.....	11
3.2.3 Interfície de mòdul: ICotigBreaker.....	12
3.3 CotigTypo: correcció tipogràfica.....	13
3.3.1 Tasca.....	13
3.3.2 Estratègia d'implementació.....	13
3.3.3 Interfície de mòdul: ICotigTypo.....	14
3.4 CotigLabeler: assignació d'etiquetes.....	14
3.4.1 Tasca.....	14
3.4.2 Estratègia d'implementació.....	14
3.4.3 Interfície de mòdul: ICotigLabeler.....	14
3.5 CotigSpell: correcció ortogràfica no contextual.....	15
3.5.1 Tasca.....	15
3.5.2 Estratègia d'implementació.....	15
3.5.3 Interfície de mòdul: ICotigSpell.....	16
3.6 CotigChooser: desambiguació morfosintàctica.....	16
3.6.1 Estratègia d'implementació.....	16
3.6.2 Interfície de mòdul: ICotigChooser.....	17
3.7 CotigMulti: tractament d'expressions multimot.....	17
3.7.1 Tasca.....	17
3.7.2 Estratègia d'implementació.....	17

El Corrector, descripció general de l'arquitectura

<u>3.7.3 Interfície de mòdul: ICotigMulti.....</u>	<u>18</u>
<u>3.8 CotigGram: correcció ortogràfica i gramatical contextual.....</u>	<u>18</u>
<u>3.8.1 Tasca.....</u>	<u>18</u>
<u>3.8.2 Estratègia d'implementació.....</u>	<u>18</u>
<u>3.8.3 Interfície de mòdul: ICotigGram.....</u>	<u>19</u>
<u>3.9 CotigDict: gestió de la càrrega i accés als diccionaris.....</u>	<u>20</u>
<u>3.9.1 Tasca.....</u>	<u>20</u>
<u>3.9.2 Estratègia d'implementació.....</u>	<u>20</u>
<u>3.9.3 Interfície de mòdul: ICotigDict.....</u>	<u>20</u>
<u>3.10 CotigShared: altres objectes lingüístics compartits.....</u>	<u>21</u>
<u>3.10.1 Tasca.....</u>	<u>21</u>
<u>3.10.2 Implementació i objectes lingüístics inclosos.....</u>	<u>22</u>

2 Presentació general

L'arquitectura interna del motor de correcció d'El Corrector és completament modular. Tot i això, exteriorment s'hi accedeix com si es tractés d'un únic motor, amb una sola API –interfície pública de funcionament–.

El motor de correcció està format per una sèrie de mòduls connectats seqüencialment de manera que la sortida d'un és l'entrada del següent, tot i que la connexió no és directa ja que està controlada pel mòdul corrector principal (CotigMain). Cada un d'aquests mòduls té una funcionalitat diferent dirigida a resoldre una tasca més o menys concreta, i tots ells en conjunt realitzen el procés de detecció i correcció d'errors tipogràfics, ortogràfics i gramaticals.

Els diferents mòduls s'agrupen per proximitat i funcionalitat en biblioteques DLL específiques, cosa que permet definir interfícies públiques individuals per a cada mòdul i també avaluar-los per separat. Això fa que cada un d'aquests mòduls, alhora, sigui fàcilment modificable i suprimible, sense requerir un esforç massa gran.

A més aquesta arquitectura permet que el responsable de cada mòdul tingui total llibertat sobre la implementació interna, l'únic requeriment és la implementació de la interfície corresponent, i la validació corresponent amb els tests funcionals d'integració.

Important: si esteu interessats en la modificació o ampliació d'algun dels mòduls o recursos (diccionaris o gramàtiques) d'El Corrector, heu de consultar també el document *Guia per al desenvolupament d'El Corrector*.

2.1 Funcionalitats d'El Corrector

El Corrector és un corrector tipogràfic, ortogràfic i gramatical de caire general, és a dir, està pensat per ser emprat en la redacció de documents escrits en català sense que hagin de pertànyer a una temàtica específica. Com tots els correctors, no està pensat per corregir qualsevol mena de text i com més s'acosti el text a un ús figuratiu de la llengua, més probabilitats tindrà l'usuari/usuària de no aconseguir-ne el resultat desitjat.

2.1.1 Definició lingüística d'El Corrector

El Corrector es defineix com a normatiu i diatòpic. És normatiu perquè segueix les normes lèxiques i gramaticals marcades per l'Institut d'Estudis Catalans i diatòpic perquè està preparat per corregir textos en les quatre grans variants dialectals del català. Aquestes variants són (per ordre alfabètic): balear, central, nord-occidental i valencià.

L'usuari/usuària d'El Corrector pot triar en quina variant dialectal vol escriure de manera que El Corrector tingui en compte quines formes farà servir. A la pràctica, aquesta adaptació a cada dialecte es reflecteix en el tractament de:

- formes verbals: per exemple, distinció entre *canto*, *cante* i *cant*
- aplicació d'algunes regles contextuais: per exemple, *per la tarda* és considerat un error en totes les variants tret de la valenciana

Altres variacions dialectals com poden ser les formes dels possessius (*meva/meua*, etc.), les dels articles (*el/es*, etc.), i les variacions lèxiques (*escombra/granera*, etc.) es donen per bones sigui quina sigui la variant triada per l'usuari/usuària.

2.1.2 Tipologia d'errors tractats

Els errors que tracta estaven definits inicialment en forma de llistat al *Plec de clàusules* que definia el concurs de licitació fet per la Generalitat de Catalunya. Aquest llistat s'ha traduït a unes especificacions concretes que detallen en quins contextos s'espera que El Corrector sigui capaç de detectar la presència d'un error, així com de marcar la paraula o paraules afectades i de fer-ne sempre que sigui possible una proposta de correcció adequada.

Els tipus de correcció que El Corrector implementa són:

- Correcció tipogràfica: detecció i proposta de correcció de l'ús incorrecte de determinats caràcters tipogràfics, com per exemple, l'ús del caràcter de l'accent tancat (˘) en lloc del de l'apòstrof ('), o l'ús del punt (.) en lloc del punt volat (·) en la *ela* geminada.
- Correcció ortogràfica no contextual: detecció i proposta de correcció per a tots aquells errors que resulten en “no paraules”, és a dir, en seqüències de lletres que no formen part del català normatiu. Aquests errors s'anomenen no contextuais perquè el sol fet de ser presents en un text, independentment de les paraules que tinguin a dreta o esquerra, els delata.

Per exemple, si escrivim “ccasa” hem escrit una paraula que no forma part del català normatiu, i ho podem afirmar sense haver de mirar quines paraules l'acompanyen. Així mateix passaria si escrivíssim “sapigut”, que tot i ser habitual en algunes variants del català parlat, no és acceptada com a forma normativa per l'Institut d'Estudis Catalans.

- Correcció ortogràfica o gramatical contextual: detecció i proposta de correcció per a tots aquells errors que resulten en seqüències de paraules incorrectes (tot i que les paraules per separat siguin paraules pròpies del català normatiu). Aquests errors s'anomenen contextuais perquè per determinar si la seqüència de paraules és correcta o no, cal fixar-se en les paraules que tenen a dreta o esquerra.

Per exemple, si escrivim “les raons econòmics de la decisió” només podem dir que “econòmics” és un error (o com a mínim que no encaixa en aquest context) si ens fixem que a la seva esquerra hi ha un article i un nom en femení plural.

Si voleu accedir a les especificacions detallades d'errors vegeu l'Annex A, document titulat *Especificacions d'errors per a El Corrector*. També el podeu trobar a l'adreça d'internet següent: <http://parles.upf.es/corrector/Downloads/AnnexA.pdf>.

2.2 Arquitectura

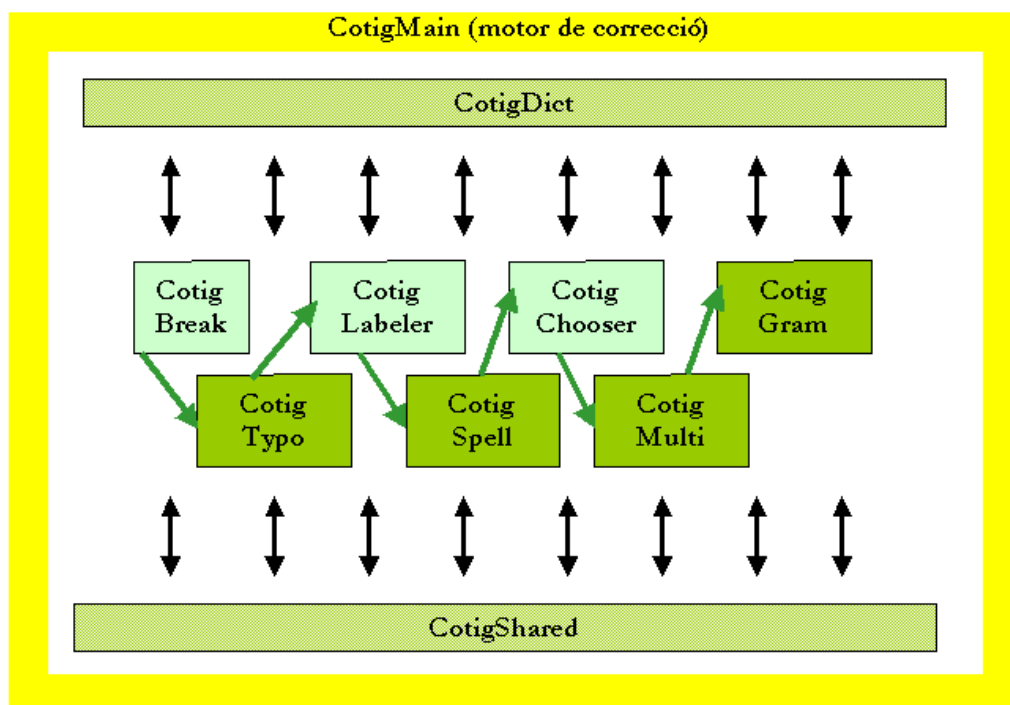
L'API del motor de correcció d'El Corrector està implementada en la biblioteca anomenada CotigMain. Qualsevol aplicació que vulgui comunicar-se amb el motor de correcció ho ha de fer a través d'aquesta biblioteca o a través d'alguna biblioteca que encapsuli aquesta biblioteca.

El CotigMain té la funció de gestionar el flux de processament de cada text. Determina com se segmenta el text en blocs de correcció (paràgrafs, frases, mots, etc.) i determina també quina mena de recursos es fan servir (diccionari, gramàtiques, etc.) en funció de les variants dialectals o els modes de correcció opcionals (tipogràfic i gramatical) triats per l'usuari/usuària.

El Corrector, descripció general de l'arquitectura

Com dèiem inicialment, es tracta d'una arquitectura completament modular, de manera que qualsevol mòdul pot ésser fàcilment suprimit, o substituït per un altre mòdul. Evidentment, aquests canvis implicaran un canvi en les prestacions d'El Corrector i han de ser realitzades per persones amb coneixements de programació i a partir de la documentació tècnica associada.

La figura que apareix en la pàgina següent reflecteix el flux de processament del text. V a seguida d'unes taules que expliquen breument les funcionalitats de cada una de les llibreries.



Biblioteca	Funcionalitat
CotigBreak.dll	Segmentació de blocs, oracions i mots
CotigTypo.dll	Correcció tipogràfica (de caràcters)
CotigLabeler.dll	Etiquetatge de mots i elements textuais especials
CotigSpell.dll	Correcció ortogràfica
CotigChooser.dll	Desambiguació de lectura morfosintàctica
CotigMulti.dll	Tractament i correcció d'entitats multimot
CotigGram.dll	Correcció ortogràfica i gramatical contextuals

Pel que fa a les biblioteques comunes a tots els mòduls (CotigDict.dll i CotigShared.dll):

El Corrector, descripció general de l'arquitectura

Biblioteca	Funcionalitat
CotigDict.dll	Gestiona la càrrega de diccionaris
CotigShared.dll	Defineix funcions comunes a totes les biblioteques del motor de correcció

3 Descripció de les llibreries DLL

3.1 CotigMain: motor de correcció

3.1.1 Tasca

El CotigMain és el motor principal d'El Corrector, que encapsula tots els mòduls independents que el formen. D'aquesta manera se'n simplifica la utilització i incorporació en d'altres aplicacions. La seva funció principal és la correcció pròpiament dita. A més, inclou un parell de funcions auxiliars per facilitar la segmentació del text.

Per utilitzar la funció de correcció cal passar-li el paràgraf que es desitja corregir, aquest bloc de text es pot marcar com a cadena de text o com a llista de tokens. La primera opció és la que poden fer servir cada un dels mòduls externs que es vulguin comunicar amb El Corrector per segmentar un text.

La segona opció té l'avantatge de poder localitzar amb més precisió la posició dels errors, ja que estan indexats per tokens, i és útil per al desenvolupament. En aquest cas les funcions auxiliars de segmentació i tokenització són útils per tenir la certesa que els criteris utilitzats seran totalment coincidents, ja que l'aplicació que crida el motor de correcció pot demandar aquestes tasques al motor igualment.

3.1.2 Estratègia d'implementació

El CotigMain s'encarrega de coordinar el flux de dades entre els diferents mòduls i de fer d'intermediari quan una aplicació externa inicialitza determinats paràmetres de funcionament que han de transmetre's als diferents mòduls (essencialment la variant dialectal i el mode de correcció).

També ofereix altres mètodes públics orientats a la fase de desenvolupament, que giren al voltant de l'accés als fitxers de bitàcola (logs) i als temps de processament de cada una de les tasques o mòduls.

3.1.3 Interfície de mòdul: ICotigMain

A l'hora de corregir un text mitjançant el motor principal, la utilització més directa és passar-li un bloc o paràgraf, la unitat mínima de correcció, i rebre una llista (array) d'objectes Error amb els problemes detectats:

```
Error[] DetectErrors(string paragraph);
```

De vegades no és senzill aïllar els paràgrafs o els títols, i per tant es pot utilitzar el motor per tal que a partir d'una cadena de text, amb tot el document, el segmenti per obtenir els blocs individuals (el Block és la unitat mínima de correcció). Això es fa amb el mètode següent:

```
BlockList SegmentBlocks(string lines);
```

Cada objecte Block conté, entre d'altres propietats, un string corresponent al segment del document associat. El següent pas és tokenitzar aquest bloc per obtenir la seqüència de tokens (TokenList) que processaran i analitzaran els diferents mòduls:

```
TokenList Tokenize(string block);
```

A partir d'aquesta llista de tokens podem utilitzar qualsevol dels mètodes següents:

```
Error[] DetectErrors(TokenList block);
```

El Corrector, descripció general de l'arquitectura

```
Error[] DetectTypographicErrors(TokenList block);  
Error[] DetectSpellingErrors(TokenList block);  
Error[] DetectGrammarErrors(TokenList block);
```

Habitualment s'utilitza el primer, ja que el motor inclou altres mètodes per activar i desactivar els tipus d'error que es volen detectar, però en determinades aplicacions pot ser interessant utilitzar les funcions més específiques.

Finalment ofereix dos mètodes molt concrets que serveixen per gestionar el diccionari d'usuari/usuària, de manera que es pugui personalitzar a l'ús de cada u la llista de mots que no s'ha de considerar error. El primer mètode afegeix una paraula al diccionari d'usuari/usuària del mòdul de correcció ortogràfica no contextual (CotigSpell), el segon mètode desa completament aquest diccionari.

```
int AddToUserDictionary(string word);  
bool SaveUserDictionary();
```

3.1.3.1 Mètodes comuns a tots els correctors

Els següents mètodes, a més d'oferir-los el CotigMain, són comuns a totes les interfícies dels mòduls correctors, concretament estan implementats per CotigTypo, CotigSpell, CotigMulti i CotigGram.

El Corrector permet activar i desactivar els diferents modes de correcció, per això els mòduls han de ser notificats mitjançant el següent mètode que els permet saber quins tipus d'error han de mirar de detectar:

```
bool SetCurrentCheckings( bool chkTypo, bool chkSpell,  
                          bool chkGram);
```

Una altre funcionalitat comú als mòduls correctors és la possibilitat d'indicar-los que ignorin determinats errors a partir de cert moment. La mecànica és proporcionar-li un objecte Error detectat prèviament i utilitzar-lo per generalitzar futurs errors que el mòdul ha de considerar similars, a efectes pràctics “similar” significa que hagin estat detectats per una mateixa regla.

Els dos mètodes següents permeten indicar al corrector quins errors s'han d'ignorar fins que se li indiqui el contrari. El primer mètode permet mostrar al corrector un error prèviament detectat perquè l'incorpori a una llista d'excepcions, el segon mètode permet buidar aquesta llista d'excepcions.

```
int AddToIgnoreErrors(Error error);  
bool ClearIgnoreErrors();
```

3.1.3.2 Mètodes comuns a tots els mòduls

Els següents mètodes, oferts pel CotigMain, són comuns a totes les interfícies dels mòduls, tant correctors com processadors, concretament estan implementats per a CotigTypo, CotigSpell, CotigMulti, CotigGram, CotigBreaker, CotigLabeler i CotigChooser.

El Corrector pot utilitzar-se per textos amb diferents variants dialectals del català, que es poden activar o desactivar de forma independent. Per indicar al mòdul quines són les variants acceptades per una correcció determinada s'utilitza el mètode següent:

El Corrector, descripció general de l'arquitectura

```
bool SetCurrentDialect(PoS Dialect dialect);
```

Els diferents mòduls utilitzen diferents fitxers de dades on s'emmagatzemen tant els diccionaris com gramàtiques, regles o d'altres paràmetres de configuració. Tots aquests fitxers s'han de trobar a la carpeta “./Data”, relativa a la carpeta de treball. Per definir-li dinàmicament la carpeta de treball s'utilitza el mètode següent:

```
bool SetCurrentFolder(string path);
```

Amb la idea que en cas que sigui necessari El Corrector pugui ampliar-se per incorporar lèxic i regles específics de determinats àmbits (mèdic, jurídic, etc.) s'ha declarat aquest mètode que indica als mòduls on han de buscar les dades:

```
bool SetCurrentPackages(string[] packages);
```

Una vegada s'ha modificat dinàmicament la carpeta de treball o els paquets que es volen utilitzar, cal indicar-li que torni a carregar tots els fitxers externs segons les noves indicacions:

```
bool LoadExternalFiles();
```

I en el cas que es vulgui substituir dinàmicament el diccionari principal per un objecte diccionari prèviament carregat a memòria es pot fer amb el següent mètode:

```
bool SetCoreDictionary(CotigDict coreDictionary);
```

Finalment, s'implementen una sèrie de mètodes generals per obtenir metainformació dels mòduls: conèixer el resultat de la darrera operació, obtenir el fitxer de bitàcola (log) de les darreres operacions realitzades, i obtenir la versió i l'autoria del mòdul:

```
bool IsAllOk();  
string GetDebugInfo();  
string GetCurrentVersion();  
string GetCredits();
```

3.2 CotigBreaker: segmentació del text

3.2.1 Tasca

El CotigBreaker és bàsicament un segmentador de text de diferents nivells: segmenta un document en blocs, detecta possibles límits d'oració i tokenitza les unitats textuais (lèxiques o no).

3.2.2 Estratègia d'implementació

En primer lloc analitza un document de text i segmenta els diferents blocs que el formen (recordeu que és la unitat mínima de correcció). A més intenta classificar els tipus de bloc per diferenciar els que són textuais dels que no.

Un tipus important de bloc és el corresponent a etiquetes [tags] de control, al processar textos amb anotacions XML/XHTML les etiquetes externes es consideren blocs independents als quals no cal aplicar el procés de correcció.

També analitza un bloc de text i el segmenta en elements textuais no ambigus, que serviran com a peces bàsiques per formar els *tokens*. Per arribar a la formació dels tokens fa primer una divisió en àtoms. Un àtom és una seqüència consecutiva de caràcters del mateix tipus (majúscules, minúscules, numèrics, espais o puntuació).

El Corrector, descripció general de l'arquitectura

En el cas de textos amb marques XML/HTML els *tags* i els seus paràmetres formen àtoms per si mateixos que hauran de ser ignorats per la resta de mòduls.

Exemples:

- |El| |mètode| |API|Setup|(|date|)| |retorna| |14|. |95|\$|. |
- |La| |lluna| |il|. |lumina| |les| |cases|. |
- |Envia|-|m|' |ho| |a| |francesc|@|cogitoergosum|. |com|. |

CotigBreaker utilitza diferents heurístiques per determinar els límits d'oració dins d'un bloc, principalment a partir de regles contextuais de seqüències de *tokens*. Insereix un *token* especial en els límits d'oració.

Exemples:

- ||L'exemple té dues oracions. ||Aquesta és la segona. ||
- ||El Sr. Garcia viu a l'Avda. Diagonal.||
- ||El Joan viu a l'Avda. ||Demà vindré.||

Analitza la seqüència d'àtoms per detectar seqüències que formin *tokens* específics, a més d'identificar-los els classifica segons la seva naturalesa.

Exemples:

- |El| |mètode| |API|Setup|(|date|)| |retorna| |14.95|\$|. |
- |La| |lluna| |il·lumina| |les| |cases|. |
- |Envia|-|m|' |ho| |a| |francesc@cogitoergosum.com|. |

3.2.3 Interfície de mòdul: ICotigBreaker

```
BlockList SegmentBlocksFromFile(string fileName);
BlockList SegmentBlocksFromFile(string fileName,
                                FileFormat format);
BlockList SegmentBlocksFromFile(string fileName,
                                FileFormat format,
                                bool emptyText);

BlockList SegmentBlocksFromDocument(string lines);
BlockList SegmentBlocksFromDocument(string lines,
                                    FileFormat format);
BlockList SegmentBlocksFromDocument(string lines,
                                    FileFormat format,
                                    bool emptyStrings);

TokenList SegmentTokens(Block block);
TokenList SegmentTokens(string paragraph);

TokenList SegmentAtomicTokens(string paragraph);
```

El Corrector, descripció general de l'arquitectura

```
TokenList MergeAtomicTokens (TokenList block);  
TokenList SplitAtomicTokens (TokenList block);
```

També implementa els mètodes comuns a tots els mòduls (vegeu secció 3.1.3.2), que es poden consultar en les seccions corresponents.

3.3 CotigTypo: correcció tipogràfica

3.3.1 Tasca

El CotigTypo és un mòdul corrector que té com a tasca principal detectar la utilització incorrecta de símbols i caràcters tipogràfics, així com fer suggeriments de possibles correccions.

La funció de correcció rep com a paràmetre d'entrada una seqüència de tokens, TokenList, corresponent a un bloc de text. El que retorna aquesta funció és una llista d'errors tipogràfics presents al paràgraf, indicant el seu codi, la posició exacta de la seqüència incorrecta, una llista amb possibles correccions, i informació sobre el mòdul i regla que ha detectat l'error.

És important destacar que com a mòdul corrector no ha de corregir ni afegir cap mena d'informació a la llista de tokens que està processant, simplement la llegeix per trobar possibles errors ortotipogràfics. Després, quan en troba, genera un objecte ErrorList amb la informació relativa a cada error trobat (lloc on comença, lloc on acaba, codi d'error, proposta de correcció, etc.).

3.3.2 Estratègia d'implementació

L'estratègia d'implementació està basada en la detecció de seqüències concretes de tokens corresponents a utilitzacions incorrectes dels elements ortotipogràfics. En certa manera cada una d'aquestes seqüències prototípiques es correspon a una regla. Com que el número de regles és de poc més d'una vintena s'ha evitat afegir-hi complexitat innecessària.

La implementació actual està basada en un anàlisi seqüencial de la llista de tokens, on per cada posició es comprova l'aplicació de les diferents regles. Les regles estan definides com a funcions privades que a partir del TokenList i de l'índex actual verifiquen les condicions d'activació. Aquestes condicions fan referència a les propietats del token indexat i dels tokens adjacents.

Aquestes funcions privades retornen NULL en cas que no s'hagin acomplert les condicions, i retornen un objecte Error en cas de detectar una seqüència incorrecta. I són les mateixes funcions, en el cos de la condició, les que omplen el contingut de l'objecte Error, indicant el codi i les correccions suggerides.

És important remarcar que les condicions de les regles s'han definit amb relativa sistematicitat, de manera que en un futur no fos massa complex extreure les regles a un fitxer extern de text on estiguessin definides a partir d'un formalisme *ad hoc*.

En el cas d'haver d'afegir alguna altra regla és recomanable partir d'alguna de les ja existents i modificar-la segons les necessitats, però recordant que per optimitzar la velocitat de la comprovació cal començar per les condicions que a) amb més freqüència donin com a resultat "false" i b) siguin menys costoses de computar.

Finalment, cal afegir que per facilitar la creació de l'estructura de suggeriments, una llista (array) de TokenList, el mòdul inclou un mètode auxiliar privat que el construeix a partir d'una cadena de text a l'interior de les regles.

3.3.3 Interfície de mòdul: ICotigTypo

El mòdul ofereix dos constructors, un sense paràmetres i un altre que rep un objecte diccionari. Normalment el corrector ortotipogràfic rebrà el diccionari principal, gestionat pel CotigMain i compartit entre tots els mòduls, però conté certes heurístiques que el permetrien funcionar sense diccionari, encara que amb menor precisió:

```
public CotigTypo()  
public CotigTypo(CotigDict sharedCoreDictionary)
```

La funció principal d'El Corrector és la detecció d'errors tipogràfics. Aquesta funció s'obté mitjançant el mètode public següent (que a partir de la seqüència de tokens retorna una llista d'errors detectats):

```
ErrorList DetectTypographicErrors(TokenList block);
```

També implementa els mètodes comuns a tots els mòduls correctors (vg. 3.1.3.1) i els comuns a tots els mòduls (vg. 3.1.3.2).

3.4 CotigLabeler: assignació d'etiquetes

3.4.1 Tasca

El mòdul CotigLabeler s'encarrega d'etiquetar tots els elements del text basant-se en l'ús i consulta del diccionari principal i unes heurístiques en cas que l'element no estigui al diccionari.

El mètode principal rep un segment de text, en una TokenList, i retorna aquesta mateixa TokenList actualitzada amb les lectures que pot tenir cada Token de la llista. No és tasca d'aquest mòdul decidir quina és la millor lectura d'entre totes les possibles, sinó donar a cada token totes les possibles.

3.4.2 Estratègia d'implementació

Per poder etiquetar el text, el mòdul es basa en un diccionari que conté totes les formes amb les seves possibles lectures. Quan es rep la TokenList, aquesta és recorreguda seqüencialment. Per a cada token, si existeix al diccionari, se li assignen les lectures que té al diccionari. Si el token no fos al diccionari, s'intenta donar una lectura basant-se en unes heurístiques.

Tokens que, per exemple, no apareixen als diccionaris, són els símbols de puntuació, les xifres, adreces de correu electrònic, números de telèfon o DNI, tokens no visibles que serveixen per a marcar final d'oració, etc.

3.4.3 Interfície de mòdul: ICotigLabeler

El mòdul CotigLabeler té un constructor, que rep com a paràmetre un objecte CotigDict, que és el diccionari principal, on es faran totes les consultes per poder etiquetar el text.

```
CotigLabeler(CotigDict coreDictionary)
```

El mètode principal per a etiquetar un text és LabelTokens, i rep com a paràmetre una TokenList, que correspon al segment de text que es vol etiquetar. Com a sortida, es

El Corrector, descripció general de l'arquitectura

retorna la `TokenList` d'entrada, on, per a cada `Token` d'aquesta llista, s'ha emplenat el camp `PossibleLabels` amb tota la llista de possibles lectures.

```
TokenList LabelTokens(Cotig.Shared.TokenList tokenList)
```

El següent mètode etiqueta únicament un objecte `Token` que rep com a paràmetre. Com a valor de retorn, torna el mateix objecte `Token` amb totes les seves possibles lectures.

```
Token LabelToken(Cotig.Shared.Token token)
```

També implementa els mètodes comuns a tots els mòduls (vg. 3.1.3.2).

3.5 CotigSpell: correcció ortogràfica no contextual

3.5.1 Tasca

El `CotigSpell` és un mòdul corrector, que té la tasca de detectar les paraules desconegudes i proposar-hi suggeriments de correcció.

La funció de correcció rep una seqüència de tokens, `TokenList`, que representa la frase que es vol analitzar. Un cop processada aquesta frase, s'hauran detectat els errors existents i s'hi proposaran suggeriments de correcció. Aquesta funció retorna una llista d'errors ortogràfics, `ErrorList`, on per a cada error trobat a la frase, s'indica el codi d'error, la posició exacta, la llista amb els suggeriments de correcció i altres dades addicionals.

La correcció ortogràfica es basa en generar variants d'un mot incorrecte, comprovar que són correctes i donar-los un pes

Com la resta de mòduls correctors, aquest mòdul no modifica la informació que rep d'entrada.

3.5.2 Estratègia d'implementació

Per detectar els mots incorrectes d'una oració, es comprova un a un, que tots aquells tokens que han estat etiquetats com a paraula siguin en un dels diccionaris carregats. Aquests diccionaris estan carregats a memòria i l'accés sol ser bastant ràpid, a canvi d'un temps de càrrega una mica major.

Quan un mot no es troba al diccionari, s'inicia una sèrie de processos per trobar propostes de correcció. Les propostes poden ser *ad hoc* o poden ser generades per un conjunt d'heurístiques.

Per a les propostes *ad hoc*, que és el primer pas, es comprova si el mot incorrecte és al diccionari de correccions *ad hoc*. En cas afirmatiu, les propostes seran aquelles que hi ha en aquest diccionari. En cas que no hi sigui, el següent pas serà aplicar una sèrie de processos que generaran propostes de correcció.

Aquests processos utilitzen unes heurístiques molt senzilles que s'apliquen sobre les lletres del mot, i que intercanvien les lletres concurrents, n'afegeixen o n'eliminen. Cada una d'aquestes variacions rep una puntuació que indica com de semblant és al mot incorrecte, i es comprova que la variació sigui al diccionari. En cas afirmatiu s'emmagatzema en una llista de propostes de correcció. Finalment aquesta llista de propostes s'ordena per la similitud amb el mot incorrecte, que serà la que es presentarà a l'usuari/usuària.

3.5.3 Interfície de mòdul: ICotigSpell

Aquest mòdul té un únic constructor, que rep com a paràmetre un objecte CotigDict, que conté el diccionari principal. Aquest diccionari ha d'haver estat carregat prèviament.

```
CotigSpell(CotigDict coreDictionary)
```

El mètode principal del mòdul es diu DetectSpellingErrors i rep com a paràmetre una TokenList que correspon a la frase que es vol analitzar i detectar els errors ortogràfics que té. El mètode retorna una llista d'errors en un objecte ErrorList que conté tots els errors ortogràfics detectats.

```
ErrorList DetectSpellingErrors(TokenList tokenList)
```

Si el que es vol és obtenir tan sols una llista de propostes de correcció donada una paraula incorrecta es pot utilitzar el mètode GetSpellingAlternatives, que rep com a paràmetre un Token que conté la paraula incorrecta com a forma, i rebrem com a valor de retorn un objecte TokenList amb la llista de propostes de correcció. Cada element d'aquesta llista, que és un Token, es correspon a un suggeriment.

```
TokenList GetSpellingAlternatives(Token token)
```

CotigSpell també implementa els mètodes comuns a tots els mòduls correctors (vg. 3.1.3.1) i els comuns a tots els mòduls (vg. 3.1.3.2).

3.6 CotigChooser: desambiguació morfosintàctica

El CotigChooser és un mòdul de desambiguació morfosintàctica. Rep un TokenList en el qual cada objecte Token conté una llista de PossibleLabels (que conté les lectures possibles de cada mot). El que retorna és la mateixa TokenList havent afegit al camp BestLabel el que segons els models lingüístics és la millor lectura donat el context.

3.6.1 Estratègia d'implementació

El CotigChooser implementa un desambiguador estadístic basat en models ocults de Markov. La selecció de l'etiqueta més probable per a cada Token es realitza considerant la TokenList com un model ocult de Markov on per establir la probabilitat d'aparició de cada element de la cadena només es consideren els elements d'un context proper. En la implementació d'El Corrector en particular, la finestra màxima considerada han estat trigramas per establir la probabilitat de les etiquetes.

L'etiqueta escollida per cada Token ha estat doncs aquella que per la cadena formada per cada TokenList maximitzava el producte de probabilitats de cada parell Token-BestLabel donat per:

$$P(\text{Label}_i | \text{Token}_i) = P(\text{Label}_i | \text{Label}_{i-2} \text{Label}_{i-1}) * P(\text{Token}_i | \text{Label}_i)$$

on

$$P(\text{Label}_i | \text{Label}_{i-2} \text{Label}_{i-1}) = \frac{(\text{Freq}(\text{Label}_{i-2} \text{Label}_{i-1} \text{Label}_i))}{(\text{Freq}(\text{Label}_{i-2} \text{Label}_{i-1}))}$$

i

$$P(\text{Token}_i | \text{Label}_i) = \frac{(\text{Freq}(\text{Token}_i \text{Label}_i))}{(\text{Freq}(\text{Label}_i))}$$

Per tal d'obtenir aquestes freqüències es requereix la construcció prèvia a partir d'un corpus etiquetat d'un model d'n-grames i d'un model lèxic que continguin les freqüències d'aparició en el corpus.

Per tal d'escollir la combinació de Labels per a cada Token que maximitza la probabilitat de la cadena formada per la TokenList s'implementa l'algorisme de Viterbi

Per tractar els casos de paraules no contingudes en el model o paraules desconegudes així com d'aquells n-grames que no hi apareguin, s'han aplicat tècniques d'*smoothing* de la probabilitat, en particular un *add-one smoothing* per a la probabilitat lèxica i un *backoff* per a la probabilitat dels trigrammes.

3.6.2 Interfície de mòdul: ICotigChooser

```
TokenList ChooseBestLabel(TokenList words);  
TokenList ChooseBestLabel(TokenList words, bool isFullSentence);
```

També implementa els mètodes comuns a tots els mòduls (vg. 3.1.3.2), que es poden consultar en les seccions corresponents.

3.7 CotigMulti: tractament d'expressions multimot

3.7.1 Tasca

El CotigMulti és un mòdul corrector que té la tasca de detectar errors en entitats multi mot, és a dir, que siguin compostes per dos o més mots. El mòdul ha de proposar suggeriments de correcció per a aquestes entitats en cas que trobi que no estan correctament escrites.

La funció de correcció rep una seqüència de tokens, TokenList, que representa la frase que es vol analitzar. Un cop processada aquesta frase, s'hauran detectat els possibles multimots i els errors corresponents, i s'hi proposaran suggeriments de correcció.

La funció retorna una llista d'errors, ErrorList, on, per a cada error multimot trobat a la frase, s'indica el codi d'error, la posició exacta, la llista amb els suggeriments de correcció i altres dades addicionals.

Com la resta de mòduls correctors, aquest mòdul no modifica la informació que rep d'entrada.

3.7.2 Estratègia d'implementació

Per detectar les entitats multi mot d'una oració, s'utilitzen dos diccionaris de multimots, independents dels diccionaris de mots simples. El primer diccionari multimot (Core) conté multimots com ara topònims, noms propis i abreviatures. El segon diccionari (*ad hoc*) conté una llista de barbarismes i les seves formes correctes associades. Aquests diccionaris estan carregats a memòria i l'accés sol ser bastant ràpid, a canvi d'un temps de càrrega una mica major.

Internament el mòdul realitza tres processos per determinar candidats:

- tractament de majúscules en entitats multimot
- llista d'errors comuns ad hoc (barbarismes, *cd-rom* per *lector de cd*, ...)
- heurístiques (inserció d'espai, esborrat d'espai extra dins el multi mot)

L'ordre dels processos determina l'ordre final de les propostes, si canvien la caixa i el multimot és al diccionari multimot, és la primera proposta. Si la trobem com a error comú és la següent proposta. Finalment, afegim les propostes d'heurístiques.

Amb aquestes estratègies, es poden detectar errors multi mots com els exemples que siguin:

- “cd-rom” per “lector de cd”,
- “Antoni M. clARet” per “Antoni M. Claret”
- “joana - francesca de Chantal” per “Joana-Francesca de Chantal”
- “aeroport del prat” per “Aeroport del Prat”

3.7.3 Interfície de mòdul: ICotigMulti

Aquest mòdul té un únic constructor que rep com a paràmetre un objecte CotigDict, que conté el diccionari principal. Aquest diccionari ha d'haver estat carregat prèviament. Els diccionaris multi mots són carregat internament pel propi mòdul.

```
CotigMulti (CotigDict coreDictionary)
```

El mètode principal del mòdul es diu DetectMultiErrors i rep com a paràmetre una TokenList que correspon a la frase que es vol analitzar i sobre la qual es volen detectar els errors ortogràfics. El mètode retorna una llista d'errors en un objecte ErrorList que conté tots els errors multimot detectats, i la la posició exacta

```
ErrorList DetectMultiErrors(TokenList tokenList)
```

També implementa els mètodes comuns a tots els mòduls correctors (vg. 3.1.3.1) i els comuns a tots els mòduls (vg. 3.1.3.2).

3.8 CotigGram: correcció ortogràfica i gramatical contextual

3.8.1 Tasca

La biblioteca CotigGram.dll s'encarrega de la detecció d'errors gramaticals i de la possible proposta de suggeriments de correcció. Aquest mòdul llegeix, analitza i compila una gramàtica de detecció d'errors. Aquesta gramàtica ha estat desenvolupada per lingüistes en un formalisme dissenyat específicament per a la tasca de correcció.

El formalisme es compon d'una sèrie de regles que s'apliquen en contextos locals. Les regles poden aplicar-se amb èxit o no. En cas que una regla s'apliqui (o s'activi), llavors aquesta du la informació necessària per fer una llista de propostes de correcció.

Cada regla **R** es compon d'una capçalera i d'una llista de restriccions lingüístiques que determinen l'activació d'**R**. La capçalera inclou un identificador de regla, un codi d'error, dos nombres naturals que indiquen l'abast de l'error dintre del context detectat, una llista de (màxim 3) variants dialectals en què s'ha d'aplicar la regla, i finalment, una llista d'accions de correcció que inclouen inserció de mots (també espais), i generació de formes a partir del lema i la categoria morfològica.

3.8.2 Estratègia d'implementació

La biblioteca CotigGram.dll conté quatre classes: CotigGram que implementa la interfície ICotigGram, Grammar (la classe principal), Parser i Rule. Un objecte de classe Grammar s'encarrega de la detecció d'errors gramaticals i del possible suggeriment de

propostes de correcció. Per dur a terme aquesta tasca, la classe CotigGram declara un objecte Grammar que ha pogut accedir a una gramàtica **G** (desenvolupada per l'equip de lingüistes) via un objecte de tipus Parser que ha estat l'encarregat de llegir i compilar **G**.

3.8.3 Interfície de mòdul: ICotigGram

És el mètode més important. Mitjançant un objecte de tipus Grammar, aplica la gramàtica i recull la llista (possiblement buida) d'objectes de tipus Error en què apareixen els errors detectats (en quina posició del document) i les possibles propostes de correcció.

```
virtual ErrorList^ DetectGrammarErrors(TokenList^ block);
```

Proporciona la versió actual del mòdul CotigGram.

```
virtual String ^GetCurrentVersion();
```

Retorna un string amb el nom dels autors del mòdul.

```
virtual String ^GetCredits();
```

Aquest mètode retorna un booleà que indica si la tasca de càrrega de la gramàtica s'ha produït sense incidents.

```
virtual String ^GetDebugInfo();
```

```
virtual bool IsAllOk();
```

Indica a CotigGram (i a l'objecte Grammar) quina varietat dialectal està activada.

```
virtual bool SetCurrentDialect(PoS Dialect dialect);
```

Aquest mètode indica a CotigGram que ha de carregar la gramàtica (via un objecte de tipus Parser).

```
virtual bool SetCurrentFolder(String ^path);
```

```
virtual bool SetCurrentPackages(array<String^> ^packages);
```

```
virtual bool LoadExternalFiles();
```

Altres mètodes que implementa són els següents:

```
virtual Boolean ClearRulesDictionary();
```

```
virtual Boolean ConsultSharedDictionary  
(Boolean consultSharedDictionary);
```

```
virtual Boolean LoadRulesDictionary(String^ fileName);
```

```
virtual bool SetCoreDictionary (Cotig::Dict::CotigDict ^);
```

```
virtual bool SetCurrentCheckings (bool chkTypo, bool  
chkSpell,
```

```
bool chkGram);
```

```
virtual int AddToIgnoreErrors(Error ^);
```

```
virtual bool ClearIgnoreErrors();
```

També implementa els mètodes comuns a tots els mòduls correctors (vg. 3.1.3.1) i els comuns a tots els mòduls (vg. 3.1.3.2).

3.9 CotigDict: gestió de la càrrega i accés als diccionaris

3.9.1 Tasca

El mòdul CotigDict s'encarrega de gestionar els diccionaris utilitzats per El Corrector. Permet carregar arxius de diccionari que es troben a disc i posar-los en memòria amb una estructura optimitzada per obtenir un accés ràpid i amb tota la informació que requereixen els mòduls que els utilitzaran.

Aquest no és un mòdul corrector, sinó que encapsula un conjunt de dades i mètodes per accedir i modificar aquestes.

3.9.2 Estratègia d'implementació

El mòdul es basa en un format d'arxiu de diccionari molt versàtil i flexible que es va dissenyar específicament per a les necessitats d'El Corrector. Els arxius de diccionari poden estar desats en format text, llegible i que pot ser modificat fàcilment i també en un format binari, on les dades estan més comprimides però no són modificables. Aquest últim format és útil quan es vol estalviar espai en disc i també per tal que la càrrega a memòria sigui més ràpida.

Un cop el diccionari està en memòria, es desa en una estructura de llista associativa (hash), que proporciona un accés molt ràpid i de temps constant independentment de la mida del diccionari. La primera tasca, que és carregar l'arxiu de diccionari a memòria, intenta carregar l'arxiu en format binari, ja que és més ràpid. En cas que aquest no existeixi, intenta utilitzar el diccionari desat en format text.

Inversament, es pot desar a disc un diccionari que estigui carregat a memòria, en qualsevol dels dos formats esmentats anteriorment.

Existeix un mètode per obtenir les dades de qualsevol mot del diccionari, tan sols cal indicar-li el mot que volem consultar i aquestes es retornen mitjançant un objecte LabelList que conté totes les lectures possibles del mot, amb un conjunt d'informacions útils per als mòduls correctors que les necessitin.

3.9.3 Interfície de mòdul: ICotigDict

El mòdul CotigDict té tres constructors. El primer inicialitza un diccionari, però no carrega cap arxiu de disc.

```
CotigDict()
```

El segon constructor, rep com a paràmetre el nom d'un arxiu de diccionari. Per defecte, aquest diccionari no s'indexarà per lema i per tant no es podrà utilitzar el mètode GetForms.

```
CotigDict(string dictionaryFilename)
```

El tercer constructor, rep el nom de l'arxiu de diccionari i un valor cert o fals per si volem indexar el diccionari per lema (útil per si hi ha algun mòdul que vol utilitzar el mètode GetForms).

```
CotigDict(string dictionaryFilename, bool indexByLemma)
```

La funció per a carregar un diccionari es diu LoadDictionary i rep com a únic paràmetre una cadena que és el nom d'arxiu del diccionari. Es pot especificar el nom de l'arxiu amb o sense extensió. Si s'especifica (.dat o .bin) es carregarà l'arxiu amb l'extensió

El Corrector, descripció general de l'arquitectura

corresponent, si en canvi, no s'especifica cap extensió, s'intentarà carregar primer l'arxiu amb extensió .bin i després el d'extensió .dat.

```
bool LoadDictionary(string fileName)
```

Es pot desar un diccionari a disc utilitzant el mètode SaveDictionary, indicant el nom de l'arxiu, sense extensió, de forma que es desarà l'arxiu en format text i binari.

```
bool SaveDictionary(string fileName)
```

Per recuperar totes les lectures que té una paraula del diccionari, cal cridar el mètode GetLabels passant com a únic paràmetre una cadena de text indicant la paraula. Aquest mètode retornarà una llista de Labels (LabelList) amb totes les lectures de la paraula. En cas que la paraula no es trobi al diccionari es retornarà el valor NULL.

```
LabelList GetLabels(string word)
```

També es pot recuperar la llista de paraules que contenen una lectura en concret, utilitzant el mètode GetForms, es passa un paràmetre que és un objecte Label. Aquest objecte correspon a la lectura que volem buscar, i s'utilitza com a màscara (filtre), és a dir, que es poden indicar o no certs camps de dades de l'objecte Label.

```
TokenList GetForms(Label label)
```

Es poden afegir dades al diccionari utilitzant el mètode AddToDictionary, que rep com a paràmetre un objecte Token que conté tota la informació necessària, com la forma de la paraula, la categoria, i les possibles lectures que té, entre d'altres. Aquest mètode retornarà un valor cert o fals depenent de si s'ha pogut afegir la informació al diccionari o no respectivament.

```
bool AddToDictionary(Token newToken)
```

Amb el mètode ClearDictionary s'elimina tot el contingut del diccionari que s'emmagatzema a memòria.

```
bool ClearDictionary()
```

A part d'aquests mètodes, el mòdul CotigDict també implementa els següents que han estat definits a la secció 3.1.3.2:

```
bool IsAllOk();  
string GetDebugInfo();  
string GetCurrentVersion();  
string GetCredits();
```

3.10 CotigShared: altres objectes lingüístics compartits

3.10.1 Tasca

CotigShared és una biblioteca que defineix les comunicacions entre mòduls i els objectes lingüístics de dades que intercanvien. És una biblioteca transversal utilitzada per tots els elements del motor corrector, ja que defineix les interfícies de cada mòdul i els objectes que reben i retornen.

Les interfícies defineixen els mètodes que han d'implementar els diferents mòduls i els seus paràmetres. La idea és que qualsevol mòdul pugui ser substituït per un altre sempre i quan aquest segon implementi la interfície corresponent.

3.10.2 Implementació i objectes lingüístics inclosos

Les interfícies que s'han definit són:

- processadors: ICotigBreaker, ICotigLabeler i ICotigChooser
- correctors: ICotigTypo, ICotigSpell, ICotigMulti i ICotigGram
- altres: ICotigDict i ICotigMain

També tenim una sèrie d'objectes:

Block i BlockList: l'objecte Block defineix la unitat mínima de correcció, bàsicament conté el text, informació sobre la posició absoluta en el document original i diferents atributs que permeten conèixer la seva naturalesa. L'objecte BlockList és una col·lecció de blocs que permet manipular el conjunt de blocs que constitueixen el document.

Token i TokenList: l'objecte Token defineix l'element lèxic base de tot el processament posterior, inclou la seva forma superficial i normalitzada, atributs sobre la seva classificació i una sèrie de mètodes que faciliten la seva regeneració per les tasques de "síntesis" de propostes. L'objecte TokenList és una col·lecció de tokens que permet manipular la llista de tokens que constitueix un paràgraf.

Label i LabelList: l'objecte Label defineix una lectura morfosintàctica d'un element lèxic. Com a atributs inclou lema, categoria gramatical, trets morfosintàctics, freqüència, variants dialectals a les quals pertany, etc. L'objecte LabelList és una col·lecció de Labels que representa el conjunt de lectures possibles d'un element ambigu.

Error i ErrorList: l'objecte Error defineix l'element bàsic que retornen els mòduls correctors i el motor principal. Defineix un error (tipogràfic, ortogràfic o gramatical) associat a una determinada seqüència de tokens, el seu codi i missatge associat, així com una llista de possibles correccions. L'objecte ErrorList és una col·lecció d'Errors que constitueix el conjunt d'errors trobats en un paràgraf determinat.

Finalment la biblioteca CotigShared també inclou la definició d'una sèrie d'enumeracions que defineixen els valors de diferents propietats dels objectes anteriors, i algunes funcions estàtiques per facilitar la manipulació i transformació de les dades comunes:

- enum associats al Block: BlockType
- enum associats al Token: TokenType i TokenShape
- enum associats al Label: PoSCategory, PoSFeatures, PoSSpecial i PoSDialect
- altres enums generals: Module, IOError i FileFormat

Per veure amb detall la definició de les interfícies, consulteu la secció d'interfícies del mòdul corresponent. Per veure amb detall els mètodes i propietats dels diferents objectes, consulteu l'annex corresponent a la documentació generada a partir del codi font.