Eines *E-Learning* UOC:
Detecció de plagi en documents *OpenOffice.org*

Alumne: Josep Cuartero Olivera
Enginyeria en Informàtica

Consultor: Robert Clarisó Viladrosa

Data Lliurament: 09-01-2006

ÍNDEX

AGRAÏMENTS 5

RESUM 7

0. INTRODUCCIÓ AL PFC 10
- **0.1 OBJECTIUS** 12
- **0.2 ESTRUCTURA** 12
- **0.3 TASQUES** 14
- **0.4 PLA DE TREBALL** 15
- **0.5 DIAGRAMA DE *GANTT*** 20
- **0.6 INCIDÈNCIES I RISCOS** 22
- **0.7 MATERIAL** 22

1. EL PLAGI AL LLENGUATGE NATURAL 24
- **1.1 INTRODUCCIÓ AL PLAGI** 25
  - 1.1.1 Què és el plagi 25
  - 1.1.2 El perquè del plagi 26
  - 1.1.3 Protecció contra el plagi 27
- **1.2 EL PLAGI AL LLENGUATGE NATURAL** 28
  - 1.2.1 El llenguatge natural i la informàtica 28
  - 1.2.2 Tipus de plagi en llenguatge natural 29
  - 1.2.3 Com evitar el plagi 29
- **1.3 DETECCIÓ DE PLAGI AL LLENGUATGE NATURAL** 31
  - 1.3.1 El plagi al món virtual de la UOC 31
  - 1.3.2 Detecció de plagi a simple vista 31
  - 1.3.3 Tècniques de detecció de plagi 31

2. PROGRAMARI PER DETECCIÓ DE PLAGI 34
- **2.1 *JPLAG*** 35
  - 2.1.1 Algoritme emprat 36
  - 2.1.2 Modificacions detectables 37
  - 2.1.3 Punts febles 37
- **2.2 *MOSS*** 38
  - 2.2.1 Algoritme emprat 39
  - 2.2.2 Modificacions detectables 39
  - 2.2.3 Punts febles 40
- **2.3 *YAP*** 40
- **2.4 ALTRES APLICACIONS** 41

3. ALGORITMES USUALS 43
- **3.1 ALGORITME *GREEDY STRING-TILING*** 44
- **3.2 ALGORITME *WINNOWING*** 45
- **3.3 DISTÀNCIA DE *LEVENSHTEIN*** 47

4. *OPENOFFICE.ORG XML* 49
- **4.1 *OPENOFFICE.ORG*** 50
  - 4.1.1 Utilització d'*OpenOffice.org* a la UOC 51
  - 4.1.2 Els arxius *OpenOffice.org* 51
- **4.2 EL FORMAT D'ARXIU *OPENOFFICE.ORG XML*** 51
  - 4.2.1 *Metadades*: El subdocument *meta.xml* 52
  - 4.2.2 Contingut: El subdocument *content.xml* 52

5. CONSTRUCCIÓ DEL PROGRAMARI 54
5.1 ANÀLISI DE REQUERIMENTS 55
5.1.1 Fiabilitat 55
5.1.2 Facilitat d'ús 55
5.1.3 Rapidesa 56
5.1.4 Claredat 56
5.2 PREPROCESSAMENT DEL CONTINGUT TEXTUAL 57
5.2.1 Eliminació de l'enunciat 57
5.3 MODEL DE DADES 58
5.3.1 Recuperació d'informació 58
5.3.2 Els *Handler XML* 58
5.3.3 Emmagatzemat de la informació 59
5.3.4 *Matches* entre documents: La classe *OpenOfficeMatch* 60
5.3.5 La classe *OpenOfficeProcessor* 60
5.3.6 Optimització del *Vector allMatches* 62
5.3.7 Control d'errors 64
5.4 DESENVOLUPAMENT 64
5.4.1 L'*interface* de l'aplicació: La classe *scrMain* 65
5.4.2 Descompressió dels documents 67
5.4.3 Lectura del contingut *XML* 67
5.4.4 L'*Entity Resolver*: La classe *OpenOfficeDTD* 67
5.4.5 Preprocés dels documents 68
5.4.6 Comparació dels documents 68
5.4.7 Informe de resultats 69
5.5 JOC DE PROVES 71

6. LÍNIES FUTURES DE TREBALL 72

CONCLUSIONS 74

GLOSSARI 76

BIBLIOGRAFIA 80

AGRAÏMENTS

El camí fins arribar al lliurament d'aquest PFC ha sigut força difícil. Crec que mai oblidaré les nits sense dormir, l'angoixa de les dates límit i, de vegades, el sentiment de que el temps se m'escapava de les mans a mesura que passaven els semestres.

Durant aquest temps, molta gent s'ha apropat al meu camí. Uns han marxat, d'altres s'han quedat. I alguns fins i tot m'han acompanyat per fer-ho una mica més fàcil. Vull dedicar aquest PFC a tota aquesta gent que d'una manera o una altra ha "sofert" amb mi el trajecte.

També el dedico, com sempre, a tot el personal i col·laboradors de la UOC per fer una mica més fàcil la dura tasca d'estudiar a distància. Sense el vostre suport moltes vegades un voldria simplement llençar la tovallola.

I molt especialment, vull dedicar el fruit de tot aquest temps a *F*. Prometo recordar sempre la manera en què em deies que un ha de fer el que ha de fer, no ha sigut gens fàcil per mi canviar temps d'estar amb tu per temps de treball en el projecte. Sembla que al final ho hem aconseguit i sempre agrairé que hagis aparegut a la meva vida.

RESUM

Està a l'ordre del dia la manera en què qualsevol àmbit té els seus punts negatius en quant a l'ús que se'n fa respecte determinades persones. En el comerç hi ha robatoris, en el sistema legal hi ha injustícies. Fins i tot avui dia en alguns mitjans de comunicació encara no hi ha una completa llibertat d'expressió.

Aquest fet, lògicament, afecta també al món de la tecnologia. La societat de la informació, precisament per ser tecnològicament avançada, ofereix potser més facilitats al frau. En el cas concret que ocupa aquest PFC no hi ha discussió possible: són tantes les fonts d'informació que es poden consultar que resulta gairebé impossible determinar si un text concret, presumptament original per part d'un autor, és en realitat tan original o es tracta d'una còpia d'altres fonts.

En realitat, el plagi es pot presentar de diverses maneres. Es pot tractar d'un simple duplicat d'un altre document idèntic o bé d'una "recopilació" de diferents fragments de text d'altres fonts. Els fragments utilitzats poden ser completament iguals o bé es pot intentar amagar el plagi, ja sigui canviant l'ordre de les paraules utilitzades, substituir algunes d'aquestes o tornant a redactar la mateixa idea amb altres paraules, cosa que no deixa de ser plagi ja que el contingut en realitat seguirà sent el mateix.

Pel que respecta als documents electrònics de text, un dels formats més usuals és el format *OpenOffice.org*, en part per tractar-se d'una aplicació d'ús gratuït i seguir la filosofia avui tan famosa del programari de codi obert. Aquest tipus de document guarda tant un text que es podria anomenar contingut en sí mateix, com una sèrie de *metadades* que també poden ser interessants per la determinació de plagi.

No hi ha dubte que el món acadèmic és un dels afectats pel plagi. Els alumnes fan les proves d'avaluació continuada a distància i fins i tot molt sovint aquestes són de caràcter individual. Així, no hi ha manera possible de controlar que aquests alumnes són realment els autors de les mateixes. En aquest PFC es planteja la possibilitat d'automatitzar el màxim possible la determinació de plagi entre una sèrie de documents en format *OpenOffice.org*.

La idea principal de fons del PFC no és trobar còpies exactes de documents, encara que també es fa, sinó estudiar quina és la "similitud" entre dos documents mitjançant les *metadades* dels mateixos i el propi contingut textual. Aquesta similitud es calcula en gran part mitjançant la cerca de fragments més o menys grans, idèntics entre tos dos documents. De fet, la paraula "idèntic" no es pot aplicar al 100% en aquest context, ja que abans d'iniciar la cerca d'aquests fragments coincidents es sotmeten els documents a un preprocessament en el qual s'eliminen totes aquelles parts que es considera que no són significatives per la cerca de coincidències, o bé que poden induir a errors en la detecció de plagi, com per exemple pot ser l'eliminació de comes, apòstrofs, accents, posar tot el text en minúscules, etc.. D'aquesta manera hom s'assegura que els documents que es comparen estan "canonitzats", és a dir, segueixen unes mateixes regles i d'aquesta manera abans d'iniciar la comparació ja són el més semblants possible.

Així, en aquest PFC primer s'estudia en què consisteix el plagi i de quines maneres es pot cometre i detectar. Posteriorment, s'aplica tot aquest coneixement a la implementació d'una utilitat que poden utilitzar els consultors de la UOC per detectar

plagi en els documents en format *OpenOffice.org* presentats pels alumnes. El resultat d'aquest PFC serà, per tant, aquesta memòria així com l'aplicació desenvolupada.

És obvi, però, que el temps utilitzat per desenvolupar aquest PFC no és suficient per implementar totes les utilitats que se li podrien donar a l'aplicació. Per aquest motiu, s'afegeix un capítol corresponent a línies futures de treball en el qual s'indica quines serien les optimitzacions o millores que es podrien considerar en un futur.

PARAULES CLAU

plagi, *paraphrasing*, còpia, comparació, *collusion*

0. INTRODUCCIÓ AL PFC

És un fet innegable que la societat de la informació ha canviat radicalment el concepte bàsic d'informació per convertir-lo en la més potent eina d'anàlisi a l'actualitat. La forma en què les empreses, institucions i fins i tot els consumidors fan servir la informació provoca que els mètodes de manipulació i anàlisi evolucionin a ritmes vertiginosos així com que apareguin noves tècniques per processar, analitzar i visualitzar la informació.

El plagi, malaltia de la societat de la informació i el coneixement, es presenta a pràcticament tots els vessants de la nostra vida quotidiana. Un d'aquests vessants és l'àmbit acadèmic, en què de vegades és difícil o fins i tot impossible comparar manualment el treball de milers d'alumnes en busca de còpies més o menys evidents.

Ja existeixen al mercat diferents eines com *JPLAG*, *MOSS* o *YAP* que fan servir diversos mètodes per a un resultat comú com és la detecció del plagi en documents en format electrònic. Aquest Projecte de Final de Carrera (PFC) es centra en l'estudi d'aquestes diferents maneres per detectar el plagi així com en el desenvolupament d'una nova eina que utilitzi algun d'aquests mètodes o una combinació de tots ells.

L'exemple concret que justifica el desenvolupament d'aquest PFC és el món universitari virtual de la UOC. Els alumnes de la UOC generalment han de presentar el resultat de l'avaluació continuada o exàmens en format electrònic. Un dels formats admesos és el format *OpenOffice.org*, que basa l'estructura dels seus documents en el metallenguatge *XML*. Davant d'aquests fets, es presenta la possibilitat de fer servir la informàtica per automatitzar i executar molt més ràpidament la cerca de plagis en aquest àmbit.

Comentario: Els exàmens no es presenten en format electrònic.

A continuació s'especifica els objectius del PFC, l'estructura d'aquesta memòria i el Pla de Treball que es farà servir durant el semestre.

0.1 OBJECTIUS

Els objectius principals que es volen assolir amb la realització d'aquest PFC són:

- Obtenir un coneixement el més acurat possible de què és un plagi i en què consisteix
- Conèixer el funcionament de les eines informàtiques existents a l'actualitat que permeten la detecció del plagi
- Desenvolupar una aplicació que permeti detectar possibles plagis entre diferents documents en format *OpenOffice.org*

0.2 ESTRUCTURA

Aquesta memòria està estructurada tenint en compte l'evolució del mateix PFC, és a dir, seguint d'una manera natural el cicle d'aprenentatge necessari per desenvolupar-lo. A continuació es llisten els apartats de què consta i entre parèntesi s'especifica un número aproximat de pàgines que ocuparà cada apartat.

- **ÍNDEX (2)**

- **AGRAÏMENTS (1)**

- **CAPÍTOL 1: "EL PLAGI AL LLENGUATGE NATURAL" (9)**

  S'expliquen les bases que fonamenten aquest PFC. Què és el plagi, quins tipus de plagi hi ha i de quina manera es poden detectar o tractar d'ocultar en un document electrònic. Es divideix en els següents apartats:

  1.1 Introducció al plagi (4)
  1.2 Detecció de plagi al llenguatge natural (5)

- **CAPÍTOL 2: "PROGRAMARI" (10)**

  S'estudia el programari més usual existent al mercat per detectar plagi així com la problemàtica del plagi en documents en format electrònic, en aquest cas concret en format *OpenOffice.org*. Consta dels següents apartats:

  2.1 *JPLAG* (2)
  2.2 *MOSS* (2)
  2.3 *YAP* (2)

2.4 Altres eines (4)

- **CAPÍTOL 3: "*OPENOFFICE.ORG XML*" (7)**

  S'estudia i es recull l'estructura dels documents en format *OpenOffice.org*. Conté els apartats següents:

  3.1 Introducció a *OpenOffice.org* (3)
  3.2 Format d'arxiu *OpenOffice.org XML* (4)

- **CAPÍTOL 4: "CONSTRUCCIÓ DEL PROGRAMARI" (18)**

  S'especifica pas a pas la construcció del programari resultat d'aquest PFC. Es detallen els jocs de proves efectuats per comprovar el correcte funcionament del mateix. Es divideix en els següents apartats:

  4.1 Anàlisi de requeriments (3)
  4.2 Model de dades (5)
  4.3 Desenvolupament (8)
  4.4 Joc de proves (2)

- **CAPÍTOL 5: "CONCLUSIONS" (3)**

  Conclusions del PFC

- **CAPÍTOL 6: "LÍNIES FUTURES DE TREBALL" (3)**

  Es raona quines podrien ser les línies futures de treball per continuar aquest PFC

- **RESUM (2)**

- **GLOSSARI (3)**

- **BIBLIOGRAFIA (3)**

0.3 TASQUES

El PFC s'ha orientat cap al seu vessant més pràctic, com és el cas concret a què fa referència: detecció de plagi en documents *OpenOffice.org*. Així, a grans trets, les tasques que es portaran a terme en el transcurs del desenvolupament del PFC són:

- **TASCA 1: "Estudi d'abast"**. S'analitza el PFC a desenvolupar per poder determinar quin serà el seu abast. Un cop definits aquests objectius, es planificarà el més acuradament possible les activitats a desenvolupar durant el semestre

- **TASCA 2: "El plagi al llenguatge natural"**. Es documenta d'una manera introductòria el concepte de plagi. Es redactarà el capítol "El plagi al llenguatge natural", a on s'explicarà què és i en què consisteix un plagi des del punt de vista de diferents autors i després s'estudiarà de quines maneres es pot plagiar o tractar d'ocultar el plagi en documents electrònics *OpenOffice.org*

- **TASCA 3: "Estudi del programari existent"**. S'estudia una mostra del programari de detecció de plagi ja existent al mercat. Es redactarà el capítol "Programari per detecció de plagi", que recollirà un resum del funcionament intern d'aquestes eines

- **TASCA 4: "Estudi del format *OpenOffice.org XML*"**. S'investiga a fons el format dels documents *OpenOffice.org XML*, ja que serà un dels pilars sobre els que treballarà el programari que es desenvoluparà posteriorment. Aquesta informació es plasmarà al capítol "*OpenOffice.org XML*"

- **TASCA 5: "Construcció del programari"**. És el nucli del PFC. S'efectua una anàlisi de requeriments i un modelat de les dades necessàries per detectar possibles plagis entre diferents documents *OpenOffice.org*. Es desenvolupa l'aplicació que permetrà aquestes deteccions i s'efectua un joc de proves per comprovar el seu correcte funcionament

- **TASCA 6: "Integració, repàs i acabament PFC"**. S'efectuen en la mesura del que sigui possible algunes modificacions al programari per millorar la seva eficàcia i se'l sotmet a un nou i últim joc de proves. Es donen els últims retocs al PFC: capítols de síntesi, integració dels capítols que composen la memòria, índex, etc.

- **TASCA 7: "Confecció Presentació Virtual"**. Es construeix una presentació virtual mitjançant el programari *PowerPoint* que permetrà una bona aproximació del contingut d'aquesta memòria a qualsevol persona, tingui o no coneixements previs sobre el tema tractat

- **TASCA 8: "Debat Virtual"**. Es celebrarà un debat virtual entre l'alumne i el tribunal d'avaluació sobre la feina desenvolupada i presentada com a PFC

0.4 PLA DE TREBALL

A continuació es mostra un quadre on s'especifiquen totes les tasques i fites en què es divideix aquest PFC. El quadre està estructurat en les següents columnes:

- **Activitat:** Número d'activitat
- **Fita:** Número de fita
- **Setmana:** Data d'inici i de fi de la setmana en curs
- **Data Inici:** Per les activitats, correspon a la data d'inici de les mateixes
- **Data Fi:** Per les activitats, correspon a la data de fi de les mateixes
- **Tasca:** Resum de la tasca a efectuar. L'inici de les tasques s'identifica pel fons de color taronja, cas en què no hi ha data de fi sinó només la data en què s'inicia. Els lliuraments i esdeveniments importants s'identifiquen pel fons de color groc, i la data també és la de l'esdeveniment

S'ha de tenir en compte que el temps diari mitjà de dedicació al PFC serà d'unes 3 hores aproximadament.

ACTIVITAT	FITA	SETMANA	DATA INICI	DATA FI	TASC
		1 (26/9 - 2/10)	***26-sep***		***Inici Tasca 1: E***
1			26-sep	28-sep	1.1 Definir els objectius del projecte i l'abast
2			29-sep	02-oct	1.2 Definir les diferents tasques. Realitzar la planificació
	1		***02-oct***		***Lliurament planific***
	2	2 (3/10 - 9/10)	***03-oct***		***Lliurament PAC 1***
			03-oct		***Inici Tasca 2: El plagi a***
3			03-oct	04-oct	2.1 Recopilació informació sobre plagis
4			05-oct	05-oct	2.2 Redacció capítol "El plagi al llenguatge natural", apar
5			06-oct	07-oct	2.3 Redacció capítol "El plagi al llenguatge natural", apar
			08-oct		***Inici Tasca 3: Estudi de***
6			08-oct	09-oct	3.1 *JPLAG*
7		3 (10/10 - 16/10)	10-oct	11-oct	3.2 Redacció capítol "Programari per detecció de plagi",
8			12-oct	13-oct	3.3 *MOSS*
9			14-oct	15-oct	3.4 Redacció capítol "Programari per detecció de plagi",
10			16-oct	---	3.5 *YAP*
10		4 (17/10 - 23/10)	---	17-oct	
11			18-oct	19-oct	3.6 Redacció capítol "Programari per detecció de plagi",
12			20-oct	20-oct	3.7 Altres eines
13			21-oct	21-oct	3.8 Redacció capítol "Programari per detecció de plagi",
	3		***21-oct***		***Lliurament provisional PAC 2 (Capíto "Programari per de***
			22-oct		***Inici Tasca 4: Estudi del for***
14			22-oct	---	4.1 Recopilació informació *OpenOffice.org XML*
14		5 (24/10 - 30/10)	---	29-oct	

15			30-oct	---	4.2 Redacció capítol "*OpenOffice.org XML*"
15		**6 (31/10 - 6/11)**	---	31-oct	
			01-nov		***Inici Tasca 5: Construc***
16			01-nov	02-nov	5.1 Anàlisi de requeriments
	4		***03-nov***		***Lliurament PAC 2 (Capítols "El plagi a per detecció***
17			03-nov	05-nov	5.2 Redacció capítol "Construcció del programari", aparta
18			06-nov	---	5.3 Creació Model de Dades
18		**7 (7/11 - 13/11)**	---	07-nov	
19			08-nov	10-nov	5.4 Redacció capítol "Construcció del programari", aparta
20			11-nov	---	5.5 Desenvolupament del programari
20		**8 (14/11 - 20/11)9 (21/11 - 27/11)**	---	27-nov	
21		**10 (28/11 - 4/12)**	28-nov	30-nov	5.6 Proves del programari
22			01-dic	03-dic	5.7 Redacció capítol "Construcció del programari", aparta
	5		***03-dic***		***Lliurament provisional PAC 3 (Ca "Construcció del programari"***
23			04-dic	---	5.8 Repàs capítol "Construcció del programari"
23		**11 (5/12 - 11/12)**	---	06-dic	
24			07-dic	---	Repàs general
24		**12 (12/12 - 18/12)**	---	12-dic	
	6		***12-dic***		***Lliurament PAC 3 (Capítol "Construc funciona***
			13-dic		***Inici Tasca 6: Integració, r***
25			13-dic	---	6.1 Millores en el programari
			---	18-dic	
26		**13 (19/12 - 25/12)**	19-dic	19-dic	6.2 Proves del programari

27			20-dic	21-dic	6.3 Redacció capítol "Línies futures de treball"
28			22-dic	23-dic	6.4 Redacció capítol "Conclusions"
29			24-dic	25-dic	6.5 Redacció capítols "Índex", "Agraïments", "Introducció
30		**14 (26/12 - 1/1)**	26-dic	27-dic	6.6 Repàs i integració memòria
	7		***27-dic***		***Lliurament provisional***
			28-dic		***Inici Tasca 7: Confecci***
31			28-dic	---	7.1 Confecció Presentació Virtual
			---	30-dic	
	8		***30-dic***		***Lliurament provisional***
32			31-dic	---	
32		**15 (2/1 - 8/1)**	---	---	Repàs general
32		**16 (9/1 - 15/1)**	---	09-ene	
	9		***09-ene***		***Lliurament Final PFC (Memòria, P***
			10-ene		***Inici Tasca 8: D***
33			10-ene	---	8.1 Debat Virtual
33		**17 (16/1 - 22/1)**	---	---	
33		**18 (23/1 - 29/1)**	---	29-ene	

0.5 DIAGRAMA DE *GANTT*

Seguidament es presenta el diagrama de *Gantt* que compren, en el mateix ordre, les tasques definides a l'apartat 0.4. La llegenda del diagrama és la que es mostra a continuació:

Lògicament, com que aquesta planificació s'ha fet al començament del projecte, encara no hi ha cap barra de progrés al diagrama. El diagrama es mostra a la pàgina següent:

Comentario: No s'han afegit les barres de progrés per controlar la durada real de les tasques.

FITA	TASQUA	INICI	FI
	1. ESTUDI D'ABAST	**26/09/05**	**02/10/05**
	1.1 Definir els objectius del projecte i l'abast	26/09/05	26/09/05
	1.2 Definir les diferents tasques. Realitzar la planificació	29/09/05	02/10/05
1	***LLIURAMENT PLANIFICACIÓ PROVISIONAL***	**02/10/05**	**02/10/05**
2	***LLIURAMENT PAC 1 (Planificació)***	**03/10/05**	**03/10/05**
	2. EL PLAG EN EL LLENGUATGE NATURAL	**03/10/05**	**07/10/05**
	2.1 Recopilació informació sobre plags	03/10/05	04/10/05
	2.2 Redacció apartat "Introducció al plag"	05/10/05	05/10/05
	2.3 Redacció apartat "Detecció de plagi al llenguatge natural"	06/10/05	07/10/05
	3. ESTUDI DEL PROGRAMARI EXISTENT	**06/10/05**	**21/10/05**
	3.1 JPLAG	06/10/05	09/10/05
	3.2 Redacció capítol "Programari", apartat "JPLAG"	10/10/05	11/10/05
	3.3 MOSS	12/10/05	13/10/05
	3.4 Redacció capítol "Programari", apartat "MOSS"	14/10/05	15/10/05
	3.5 YAP	16/10/05	17/10/05
	3.6 Redacció capítol "Programari", apartat "YAP"	18/10/05	19/10/05
	3.7 Altres eines	20/10/05	20/10/05
	3.8 Redacció capítol "Programari", apartat "Altres eines"	21/10/05	21/10/05
3	***LLIURAMENT PROVISIONAL PAC 2***	**21/10/05**	**21/10/05**
	4. ESTUDI DEL FORMAT OPENOFFICE-XML	**22/10/05**	**31/10/05**
	4.1 Recopilació informació OpenOffice-XML	22/10/05	25/10/05
	4.2 Redacció capítol "OpenOffice / XML"	30/10/05	31/10/05
	5. CONSTRUCCIÓ DEL PROGRAMARI	**01/11/05**	**06/12/05**
	5.1 Anàlisi de requerriments	01/11/05	02/11/05
4	***LLIURAMENT PAC 2***	**03/11/05**	**03/11/05**
	5.2 Redacció capítol "Const. del programari", apartat "Anàlisi de requerim."	03/11/05	05/11/05
	5.3 Creació Model de Dades	06/11/05	07/11/05
	5.4 Redacció capítol "Construcció del programari", apartat "Model de Dades"	08/11/05	10/11/05
	5.5 Desenvolupament del programari	11/11/05	27/11/05
	5.6 Proves del programari	28/11/05	30/11/05
	5.7 Redacció capítol "Const. del programari", apartat "Desenvolupament"	01/12/05	03/12/05
5	***LLIURAMENT PROVISIONAL PAC 3***	**03/12/05**	**03/12/05**
	5.8 Repàs capítol "Construcció del programari"	04/12/05	06/12/05
	REPÀS GENERAL	**07/12/05**	**12/12/05**
	Repàs general	07/12/05	12/12/05
6	***LLIURAMENT PAC 3***	**12/12/05**	**12/12/05**
	6. INTEGRACIÓ, REPÀS I ACABAMENT PFC	**13/12/05**	**27/12/05**
	6.1 Millores en el programari	13/12/05	18/12/05
	6.2 Proves del programari	19/12/05	19/12/05
	6.3 Redacció capítol "Línies futures de treball"	20/12/05	21/12/05
	6.4 Redacció capítol "Conclusions"	22/12/05	23/12/05
	6.5 Redacció "Índex", "Agrain.", "Introd.", "Resum", "Glossari" i "Bibliog."	24/12/05	25/12/05
	6.6 Repàs i integració memòria	26/12/05	27/12/05
7	***LLIURAMENT PROVISIONAL MEMÒRIA COMPLETA***	**27/12/05**	**27/12/05**
	7. CONFECCIÓ PRESENTACIÓ VIRTUAL	**28/12/05**	**30/12/05**
	7.1 Confecció Presentació virtual	28/12/05	30/12/05
8	***LLIURAMENT PROVISIONAL PRESENTACIÓ VIRTUAL***	**30/12/05**	**30/12/05**
	REPÀS GENERAL	**31/12/05**	**05/01/06**
	Repàs general	31/12/05	05/01/06
9	***LLIURAMENT FINAL PFC (Memòria, Presentació Virtual i Programari)***	**09/01/06**	**09/01/06**
	8. DEBAT VIRTUAL	**10/01/06**	**25/01/06**
	8.1 Debat virtual	10/01/06	25/01/06

0.6 INCIDÈNCIES I RISCOS

Les possibles incidències i riscos amb què hom es pot trobar en el transcurs del desenvolupament d'aquest PFC es recullen a la següent taula, incloent un possible pla de contingència:

INCIDÈNCIA	PLA DE CONTINGÈNCIA
Dificultat per trobar material tècnic sobre algun dels temes a tractar	Posar-se en contacte amb el consultor per demanar-li on poder trobar més informació
Imprevistos laborals (viatge, hores extra, etc.)	S'intentaria arribar a un acord amb l'empresa per poder recuperar el temps perdut de feina del PFC. En cas negatiu, es dedicarien més hores de les previstes inicialment al PFC quan finalitzessin els imprevistos
Arriba la data de lliurament d'alguna PAC o d'acabament del PFC i encara no està enllestit	Es demanaria vacances els dies necessaris a la feina per poder acabar els lliuraments a temps

0.7 MATERIAL

El material que es farà servir per a un correcte desenvolupament del PFC és el següent:

- ***PC Pentium IV 2.0GHz amb 640MB de RAM i connexió a Internet***, on s'instal·larà tot el programari necessari i es farà la cerca d'informació per elaborar el PFC
- ***Sistema Operatiu Windows XP Professional***
- ***Microsoft Project 98***, amb el qual es farà el diagrama de *Gantt* corresponent a la planificació del PFC
- ***Microsoft Word XP***, que es farà servir per redactar la memòria
- ***Microsoft PowerPoint XP***, que es farà servir per confeccionar la Presentació Virtual
- ***OpenOffice.org 2.0***, última versió estable d'aquest programari en el moment de l'elaboració del PFC i que es farà servir per l'estudi del seu format d'arxiu
- ***Java 2 Platform Standard Edition (J2SE)***, que serà l'entorn base de desenvolupament que s'utilitzarà per programar l'aplicació resultat d'aquest PFC

- ***Borland JBuilder 9 Personal Edition***, entorn *IDE* utilitzat per sobre de *J2SE* per facilitar el desenvolupament

1. EL PLAGI AL LLENGUATGE NATURAL

Com ja es va introduir al capítol 0, es pot considerar el plagi com una malaltia del coneixement. Tenint en compte l'esforç de la humanitat per evolucionar, el plagi no aporta res més que una pèrdua de temps per la mateixa ja que no introdueix nous pensaments ni coneixement, sinó que es limita a "disfressar" els prèviament existents.

La comunicació és el mitjà amb què les persones es transmeten sentiments, idees i coneixement. Això es pot fer de moltes i molt diverses maneres: parlant, amb senyals de fum, amb signes o fins i tot amb mirades i gestos. De vegades aquesta comunicació, però, esdevé ambigua en el sentit que el receptor pot entendre quelcom diferent d'allò que l'emissor vol expressar. Això es dóna, per exemple, en el llenguatge natural, la manera més usual de comunicació entre les persones; en aquest tipus de llenguatge una mateixa cosa es pot expressar de moltes maneres diferents i, de manera similar, una expressió es pot tenir diverses interpretacions diferents.

Dintre del marc que envolta aquest PFC, en aquest capítol s'introdueix el concepte de plagi aplicat al llenguatge natural. Es veurà de quines maneres es pot plagiar en aquesta forma de comunicació i s'aplicarà aquest estudi al món acadèmic virtual de la UOC.

1.1 INTRODUCCIÓ AL PLAGI

La societat actual està regida per la informació. La informació ens ajuda a obtenir coneixement que, al seu torn, ens facilita la presa de decisions així com l'anàlisi de la causa de certs fets que han esdevingut. Mitjançant la informació del passat, per exemple, una empresa pot extreure el coneixement de perquè hi ha hagut una davallada en els beneficis i, de la mateixa manera, amb la informació del present pot crear el coneixement necessari per prendre una decisió i actuar en conseqüència perquè aquesta situació no es torni a donar.

La duplicitat és un dels problemes més usuals amb què s'ha de lluitar per obtenir una informació de qualitat. Aquesta situació encara empitjora quan aquesta duplicitat s'oculta als ulls del lector com succeeix amb el plagi, tema principal d'aquest PFC.

En aquest apartat s'explica què és el plagi i quines són les causes que afavoreixen la seva aparició.

1.1.1 Què és el plagi

El plagi, bàsicament, consisteix a atribuir-se com a pròpies certes idees o coneixement prèviament creats.

Segons la *RAE* [1] (*Real Academia Española*), plagiar és "copiar substancialment obres alienes, donant-les com a pròpies".

---

[1] http://www.rae.es/

El fet d'utilitzar una sèrie de paraules o frases d'una altra font constitueix plagi. Però no només això és plagi; fins i tot un autor es pot auto-plagiar a sí mateix, ja que és freqüent que un dels seus treballs anteriors a què està fent referència en un altre més actual amb afany de lucre tingui una sèrie de drets adquirits per l'editor que el va publicar originalment. De totes maneres, encara que es tracti d'un treball propi i sense drets adquirits per part d'un tercer, es fa palès el desprestigi que proporciona el fer servir repetidament les mateixes paraules i idees.

No cal fer menció al dilema ètic que es presenta quan es comet plagi, molt més encara en el món acadèmic. El fet que un autor plagiï un altre no només li fa perdre credibilitat, sinó que també pot ocasionar una pèrdua de prestigi de la institució a la qual pertany, ja sigui una universitat, una editorial o un centre d'investigació. Aquests, per tant, es veuen obligats a prendre mesures per combatre i prevenir l'aparició del plagi.

El que queda clar, però, és la dificultat que hi ha en definir el límit que separa el que és plagi del que no ho és. Això es veurà més endavant, a l'apartat 1.2.

1.1.2 El perquè del plagi

Crear, fins i tot la simple tasca d'escriure, és bo per l'intel·lecte, la facilitat de comprensió i l'aprenentatge. Per demostrar-ho només s'ha de veure, per exemple, la facilitat amb què s'assimilen coneixements només pel fet de prendre notes sobre algun tema. Bo i així la creació de coneixement ja sigui parlat o escrit, com a qualsevol altre art, no és una tasca fàcil i sovint això afavoreix l'aparició de plagi. No és d'estranyar, doncs, iniciar una cerca d'informació a Internet i trobar desenes de vegades els mateixos continguts amb diferents paraules i, alguns cops, fins i tot amb les mateixes.

Un dels àmbits més freqüents en què es dóna el plagi és l'àmbit acadèmic. Com s'explica a [1], l'estudiant necessita estímuls del seu entorn per poder desenvolupar un treball original i d'aquesta manera evitar el plagi. De vegades, fins i tot, pot arribar a ser més estimulant per un alumne intentar ocultar el plagi comès que el fet de desenvolupar un nou treball amb el coneixement adquirit, convertint-se en una recopilació d'informació sense cap aportació nova.

A més, les noves tecnologies estan afavorint i facilitant molt la millora en la presentació dels documents. Si es té en compte, però, la gran inversió de temps que representa confeccionar una bona presentació d'un document, no és difícil imaginar que aquests alumnes no estaran disposats o bé no es podran permetre invertir el mateix o fins i tot més temps a l'àrdua tasca que és "omplir" el document, més encara sabent la facilitat amb què es pot trobar informació i recursos sobre pràcticament qualsevol tema, principalment a Internet.

Un altre punt que afavoreix el plagi a l'àmbit acadèmic és la proliferació d'espais *web* de recursos *on-line* gratuïts per a estudiants, com "*El Rincón del Vago"* [2]. En aquests espais es poden trobar treballs d'altres estudiants que de fet, molt sovint, ja estan plagiant altres documents i que fins i tot de vegades es tornen a presentar de manera idèntica com a nous, sense haver-ne fet cap canvi.

---

[2] http://www.rincondelvago.com

Tots aquests fets estan minant l'esperit acadèmic d'investigació. Cada cop és més difícil ser original i creatiu i els alumnes es recolzen en els recursos ja existents, els quals s'haurien de fer servir únicament com a eina de suport.

El plagi, però, no té perquè ser sempre voluntari. Moltes vegades, sobretot amb els recursos existents a Internet, hom pensa que aquests són de domini públic i que no tenen propietari. Així, és molt usual trobar treballs d'estudiants en què el document es basa en una recopilació de "retalls" d'altres documents trobats a Internet, amb la conseqüent pèrdua de consistència i homogeneïtat. Altres vegades és la manca d'informació sobre aquesta pràctica la que provoca la seva aparició.

1.1.3 Protecció contra el plagi

Una de les eines que tenen els autors per defensar-se contra el plagi són els anomenats drets d'autor, que proporcionen un arma vers l'ús fraudulent de les seves obres. Tal i com s'explica a [4], el dret d'autor és "un terme jurídic que descriu els drets atorgats als creadors per les seves obres literàries i artístiques. [...] El tipus d'obres que abasta el dret d'autor comprèn: obres literàries com novel·les, poemes, obres de teatre, documents de referència, diaris i programes informàtics; bases de dades; pel·lícules, composicions musicals i coreografies; obres artístiques com pintures, dibuixos, fotografies i escultura; obres arquitectòniques; publicitat, mapes i dibuixos tècnics".

A tot el món hi ha organismes com la *OMPI* [3] (*Organización Mundial de la Propiedad Intelectual*) o la *SGAE* [4] (*Sociedad General de Autores y Editores*), que es dediquen a protegir aquests drets. El cas concret de la *OMPI* és potser el més rellevant, ja que hi estan adscrits la major part dels països del món per col·laborar conjuntament a favor dels drets de la propietat intel·lectual.

[3] http://www.ompi.int/portal/index.html.es
[4] http://www.sgae.es

1.2 EL PLAGI AL LLENGUATGE NATURAL

A l'apartat anterior es veia com el plagi no només afecta el món de les lletres, sinó el món de l'art i el coneixement en general. Aquest PFC, però, es centre en l'estudi del plagi al llenguatge natural.

Comentario: centra

El llenguatge natural és el que fan servir les persones per comunicar-se. Aquest tipus de llenguatge està ple de matisos i ambigüitats, ja que el nombre de maneres en què es pot dir una mateixa cosa és molt ampli, només limitat per la capacitat de vocabulari i d'expressió de l'interlocutor.

Aquestes característiques fan que el plagi al llenguatge natural sigui molt més difícil de detectar que en els llenguatges formals, com per exemple un llenguatge de programació, en què el significat d'una frase ve donat només per la disposició de les seves paraules.

En aquest apartat s'introdueix la dificultat en processar el llenguatge natural, de quines maneres s'hi pot cometre plagi i com es pot evitar.

1.2.1 El llenguatge natural i la informàtica

El llenguatge natural no és un tipus de llenguatge fàcil d'interpretar. La gramàtica, que és la manera en què s'utilitzen les paraules per formar expressions, permet expressar una mateixa cosa amb diferents disposicions de paraules. Això, òbviament, dificulta la tasca de processar el llenguatge en aplicacions informàtiques.

La manera informàtica de processar qualsevol tipus de llenguatge és mitjançant el *parsing*. Aquest és el mètode que es fa servir, per exemple, en interpretar un llenguatge de programació per un compilador. El que es fa amb el *parsing* és aplicar les regles gramaticals del llenguatge, sovint construint una estructura en forma d'arbre, del qual les fulles es coneixen amb el nom de *token*. Amb aquests *tokens* es pot arribar a interpretar el significat de les paraules que s'han processat i permet comparar documents molt més fàcilment.

Això, com s'ha comentat anteriorment, no és gens fàcil quan es tracta de processar el llenguatge natural. Les màquines han demostrat ser molt superiors a l'ésser humà en quant a rapidesa i precisió en càlculs matemàtics, però no disposen de moltes altres característiques imprescindibles per a processar el llenguatge natural, com per exemple el sentit comú. El llenguatge natural és un llenguatge ambigu però les persones, mitjançant la seva intel·ligència i el seu coneixement previ, poden arribar a interpretar inequívocament quelcom que és molt difícil d'interpretar per un ordinador, almenys a l'actualitat.

Les tècniques *NLP* (*Natural Language Processing*) són una branca de la intel·ligència artificial que estudien la interpretació informàtica del llenguatge natural. Fins el moment s'ha treballat molt en el camp del procés i reconeixement d'aquest llenguatge i les tècniques desenvolupades ja permeten el seu procés amb més o menys èxit i potser proporcionaran les màquines, en un futur, l'habilitat de comprendre el significat de les

Comentario: estudia

paraules de la mateixa manera que ho fa una persona. D'aquesta manera s'obriria un nou món de possibilitats en l'àmbit informàtic.

1.2.2 Tipus de plagi en llenguatge natural

Hi ha diferents maneres d'utilitzar recursos ja existents que es consideren com a plagi, uns més evidents que altres. A [2] s'expliquen aquestes maneres amb alguns exemples clarificadors.

A continuació s'esmenten diverses formes usuals de cometre plagi:

- Copiar directament de la font original
- Canviar les paraules emprades, però mantenir el significat de la frase i usualment la mateixa estructura. Això es coneix amb el nom de *paraphrasing* i es pot fer de diverses maneres com canviant l'ordre de les paraules emprades o emprant sinònims per substituir-ne algunes a la frase original
- Atribuir-se com a propi un treball d'un tercer sense ni tan sols fer-hi cap canvi
- No citar les fonts d'on s'ha extret la informació original
- Retallar i enganxar de diferents documents origen per crear un "nou" treball
- Usualment en l'àmbit acadèmic també es dóna el que s'anomena col·lisió, *collusion* en anglès. En aquest cas, dos o més alumnes treballen junts per desenvolupar una tasca que se'ls ha assignat individualment. Així, el treball dels alumnes seria idèntic o bé molt similar

Aquest PFC es centra en la detecció d'aquest últim cas, la col·lisió, en el treball dels alumnes de la Universitat Oberta de Catalunya.

1.2.3 Com evitar el plagi

Com es va veure a l'apartat 1.1.2, moltes vegades el plagi no és voluntari. Així, es fa necessari aprendre de quina manera es poden emprar les diferents fonts utilitzades sense caure en aquest parany.

Primer de tot s'ha de tenir en compte que tota expressió, teoria, idea o dada que apareix en una obra s'assumeix com a pròpia i original sempre que no es citi la referència d'on s'ha extret; en cas contrari es corre el perill d'estar cometent plagi.

Tal i com es pot veure a [3], existeixen certes formes d'evitar el plagi encara que es faci servir material d'un tercer. Això s'aconsegueix citant les fonts emprades en els següents casos:

- Sempre que es faci servir idees, opinions i/o teories d'una altra persona s'ha de citar la font

- Si a més es fan servir les paraules textuals originals, aquestes han d'aparèixer entre cometes

- Encara que no s'emprin exactament les mateixes paraules, si el contingut i el significat és el mateix també s'ha de citar la font original

- També s'ha de citar la font en cas que es faci servir estadístiques, dades o gràfics que no siguin de coneixement general

- La majoria d'institucions acadèmiques importants vetllen per la desaparició del plagi informant convenientment d'aquesta metodologia als seus alumnes i castigant amb severitat el no compliment de la mateixa, sigui o no intencionadament.

- Hi ha tutorials específics, com *Glatt Plagiarism Teaching Program* [5], que ensenyen els usuaris a distingir quan es produeix plagi. Aquest, a més, proporciona una eina per fer un *test* d'un document de manera automàtica per informar d'un possible problema de plagi

[5] http://www.plagiarism.com/teaching

1.3 DETECCIÓ DE PLAGI AL LLENGUATGE NATURAL

El plagi, com és obvi, és impossible d'eradicar. Resulta del tot inviable contrastar un determinat treball amb totes les fonts que l'autor podria haver consultat, ja siguin treballs d'altres alumnes, recursos a Internet o recursos bibliogràfics. Així, l'únic "remei" que s'hi pot aplicar és la seva detecció amb el posterior càstig a l'autor.

En aquest apartat s'estudia quins són els fets a tenir en compte en la detecció de plagi al llenguatge natural i quins són els mètodes més usuals que s'empren per efectuar aquesta detecció.

1.3.1 El plagi al món virtual de la UOC

La naturalesa virtual de l'entorn acadèmic de la UOC pot afavorir també l'aparició de plagi, ja que les proves d'avaluació continuada no són presencials i això en certa manera dóna certa llibertat per consultar tot el material que l'estudiant vol. Bo i així, un possible plagi a l'avaluació continuada es pot detectar en realitzar la prova de validació, en què es fan certes preguntes a l'alumne per comprovar que realment les proves d'avaluació continuada presentades les ha redactat ell mateix. Per un alumne que no hagués estat l'autor d'aquestes, o bé aquestes són resultat de plagi, seria difícil explicar amb claredat quin és el contingut o com ha arribat a certes conclusions.

És més, encara que l'alumne hagués plagiat altres fonts, si és capaç de superar una prova de validació implicaria que ha assimilat els conceptes principals continguts a les proves, havent aconseguit –potser d'una manera menys lícita- els objectius de l'assignatura en qüestió.

1.3.2 Detecció de plagi a simple vista

Quan s'està estudiant l'obra d'un determinat autor, la detecció de plagi és més fàcil per una persona familiaritzada amb la seva obra ja que coneix bé les paraules més emprades per l'autor, el tipus de veu que fa servir normalment, el seu vocabulari, l'estil de redacció, etc. En l'àmbit acadèmic, per exemple, un professor que porti un cert número de cursos corregint el material d'un alumne determinat, podria veure immediatament quelcom estrany en el treball que corregeix en cas que s'hagués produït plagi.

Una cosa semblant succeeix quan la persona que examina el treball coneix molt bé el tema a què fa referència, així com les fonts de referència més usuals. En aquest cas també sabrà d'immediat si l'autor està plagiant alguna d'aquestes fonts.

1.3.3 Tècniques de detecció de plagi

De vegades detectar el plagi és una tasca difícil. Alguns cops, fins i tot existint un plagi evident, aquest passa inadvertit per la gran quantitat de documents que s'han de processar, motiu pel qual existeixen determinades tècniques que faciliten aquest procés.

Abans de processar els documents a la cerca de plagi s'ha de fer un "preprocessament" per deixar-los tots en un format que es podria anomenar estàndard. Aquest format, segons [12], hauria de complir les següents normes:

- Insensibilitat al format del text. És a dir, el procés de comparació no hauria de tenir en compte caràcters de format com espais en blanc, salts de línia, signes de puntuació, etc.

- Supressió de soroll. Les coincidències trobades haurien de ser suficientment significatives com per ser considerades. Així, les coincidències molt curtes, com per exemple una preposició, també s'haurien de discriminar

- Independència de posició. El canvi en l'ordre d'aparició del text al document no hauria d'afectar la fiabilitat de la detecció. Així, afegir text al document original no hauria d'afectar la resta. De la mateixa manera, esborrar text del document original no ha de suposar cap dificultat per la detecció de la resta del document

D'aquesta manera es pot reduir un possible marge d'error en identificar el plagi. També s'haurien d'eliminar les paraules específiques del tema tractat al document, ja que encara que puguin ser idèntiques entre diversos documents no demostren plagi.

La forma de plagi més difícil de detectar sovint és el *paraphrasing*, explicat anteriorment a l'apartat 1.2, degut a que la majoria de vegades requereix d'una anàlisi semàntica completa del fragment presumptament plagiat per comprovar si efectivament s'està produint plagi o no. En aquests casos, és millor efectuar el preprocessament abans esmentat i aplicar altres tècniques més adients.

Tal i com es pot veure a [2], algunes de les tècniques més usuals per detecció de plagi són:

- Estudiar variacions en el tipus de vocabulari emprat, l'aparició de paraules poc usuals o incongruències/inconsistències al llarg del treball. Això sovint és senyal de què s'ha utilitzat el material de diferents fonts sense haver-ne estudiat el seu contingut per sintetitzar-lo en un únic document

- L'aparició de referències al text que no queden reflectides a la bibliografia indica que s'ha copiat el text d'una altra font, sovint efectuant-hi *paraphrasing* per amagar el plagi, i no s'ha copiat la bibliografia corresponent

- Quan es fa una comparació entre dos o més documents es poden fer servir determinades mètriques. Aquestes mètriques representen dades característiques de cada document, com poden ser la longitud mitjana de les frases, el número d'errors ortogràfics, la mateixa estructura sintàctica del text o fins i tot cadenes de text idèntiques. Resulta més fàcil comparar els documents per aquestes mètriques que fer-ho amb el text complet. Així, si es troba una coincidència de mètriques prou significativa, es pot realitzar una anàlisi més exhaustiva entre els documents coincidents. El problema d'aquesta tècnica és

que només són útils quan s'apliquen a documents sencers i no només a fragments dels mateixos. Algunes d'aquestes mètriques són les conegudes com *Gunning-Fog* [6], *Flesch Reading Ease* [7] o *SMOG* [8]

- També existeixen algunes eines on-line de detecció de plagi, com per exemple *plagiarism.org* [9]. Aquestes eines recullen tots els documents que s'han de processar i es comparen amb d'altres que ja s'havien lliurat prèviament, amb recursos d'universitats i amb recursos a Internet. Després de processar els documents, es tornen a l'avaluador juntament amb un informe de les parts en què hi ha una certa sospita de cometre plagi. Aquestes eines es veuran amb més deteniment al següent capítol

A l'hora de decidir si una certa similitud entre dos documents és plagi o no, un dels principals problemes és on situar-hi el límit. Per exemple, és clar que si un document coincideix per mètriques amb un altre en un 100% significa que s'ha produït plagi. De la mateixa manera, si les coincidències són d'entre un 0 i un 5% es pot afirmar que no s'ha produït. Però què passa si només coincideixen en un 50%? Això per si sol no demostra plagi, fent-se necessari en aquest cas la intervenció humana per prendre una decisió. Cal dir, però, que ni tan sols la intervenció humana pot garantir la detecció del plagi.

[6] http://en.wikipedia.org/wiki/Gunning-Fog_Index
[7] http://en.wikipedia.org/wiki/Flesch-Kincaid_Reading_Ease
[8] http://en.wikipedia.org/wiki/SMOG_Index
[9] http://www.plagiarism.org

2. PROGRAMARI PER DETECCIÓ DE PLAGI

Com es va veure al capítol anterior, les tècniques per a processar el llenguatge natural encara no permeten a les màquines comprendre i interpretar aquest llenguatge d'una manera eficient i sense equívocs.

Bo i així, ja existeixen al mercat algunes eines que fan servir diverses tècniques per a detectar el plagi en el llenguatge natural, encara que la major part serveixen només per llenguatges de programació.

En aquest apartat es farà un breu repàs a algunes d'aquestes aplicacions més importants. No s'aprofundirà, però, en el funcionament en detall dels mateixos ja que no és l'objectiu d'aquest PFC. D'altra banda sí que s'explicarà més acuradament en un darrer apartat els algoritmes que utilitzen per la possibilitat d'implementar-los al nou programa que es desenvoluparà.

2.1 *JPLAG*

JPlag és un programari per detecció de plagi als llenguatges de programació *Java*, *C*, *C++* i *Scheme*, encara que també es pot fer servir amb llenguatge natural. Va ser creat per Guido Malphol a la universitat alemana de *Karlsruhe* [10], començant com un treball de recerca que posteriorment s'ha continuat desenvolupant al departament d'informàtica d'aquesta universitat.

El programa es pot executar des de *web*, versió disponible a la pàgina oficial de *JPlag* [11]. Per poder fer servir el programari, però, és necessari demanar als autors per correu electrònic [12] un número d'usuari.

A l'apartat 1.3.3 es va explicar de quina manera es pot detectar plagi mitjançant una sèrie de mètriques o mesures. *JPlag* utilitza un sistema semblant, comparant els documents objecte d'estudi i determinant un índex de similitud entre ells a partir de les igualtats que troba als texts.

Un cop s'han processat els documents es presenta un informe amb el resultat de les comparacions. En un principi es poden descartar aquells que tenen una similitud amb els altres igual o inferior al 5%. Si la similitud està per sobre d'un percentatge determinat, els documents es presenten en pantalla dos a dos, tal i com es veu a la Figura 2.1.1 extreta de [8]. D'aquesta manera el corrector podrà avaluar d'una manera molt més fàcil fins a quin punt s'ha produït plagi.

Com es va comentar a l'apartat 1.2.1, el *parsing* és la forma més usual en informàtica per processar un document en qualsevol llenguatge, ja sigui natural o de programació, convertint les sentències d'un document en una sèrie de *tokens* que venen donats per les regles lèxiques del llenguatge en qüestió. Aquest és el primer pas que efectua *JPlag* sobre tots els documents a comparar.

---

[10] http://www.uni-karlsruhe.de/
[11] http://www.jplag.de/
[12] A l'adreça jplag@ira.uka.de

Figura 2.1.1 Documents avaluats dos a dos per *JPlag*

És necessari que els *tokens* generats en efectuar el *parsing* siguin prou representatius com perquè no estiguin afectats pels canvis que es puguin efectuar per intentar ocultar el plagi. Així, com s'explicava a l'apartat 1.3.3, el preprocés ha d'incloure l'eliminació d'espais en blanc, comentaris, etc..

Un cop preprocessats tots els documents, és a dir, convertits en conjunts de *tokens*, es comparen els *tokens* de cada parell de documents. El procés de comparació intenta substituir els *tokens* d'un dels documents amb un o més *tokens* de l'altre, construint d'aquesta manera una llista de percentatges de similitud entre cada parell de documents. A la figura 2.1.1 es podia veure de quina manera es visualitzen els fragments coincidents.

2.1.1 Algoritme emprat

L'algoritme emprat per la comparació és l'anomenat *Greedy String Tiling*, com presenta Michael Wise a [7] i s'explica amb més profunditat al capítol 3. El procés de comparació entre dos *tokens A* i *B* és el següent :

- Qualsevol *token* d'*A* només pot correspondre amb exactament un *token* de *B*. Això implica que és impossible que parts del document origen tinguin una correspondència completa al document destí si aquestes han estat duplicades

- La reordenació dels *tokens* en una cadena no afectarà la detecció, és a dir, aquest tipus de plagi serà efectivament detectat

- Com és obvi, en la comparació tindran molta més importància les cadenes llargues coincidents que les curtes, ja que proporcionen un percentatge de fiabilitat més alt

Com que els documents s'han de comparar dos a dos, el temps d'execució augmentarà de forma quadràtica en incrementar el nombre de treballs a avaluar. En quant a la longitud dels treballs, aquesta fa incrementar el temps d'execució pràcticament de forma lineal.

Comentario: El temps d'execució en cas pitjor és cúbic.

2.1.2 Modificacions detectables

Hi ha certes modificacions que es poden fer al document original però que no passarien inadvertides a *JPlag*. A la taula 2.1.1 es presenten algunes d'aquestes possibilitats i la manera en què *JPlag* les detecta.

2.1.3 Punts febles

De la mateixa manera, hi ha altres modificacions que sí que poden passar desapercebudes en major o menor grau per *JPlag*. Es mostren a la taula 2.1.2 algunes de les modificacions amb més èxit juntament amb la raó de què puguin no ser detectades.

Modificació	Detecció
Format del document: afegir o esborrar línies en blanc, afegir espais, canviar la indentació, etc.	*JPlag* esborra tots aquests caràcters en preprocessar els documents, de manera que no suposen cap canvi
Inserció, esborrat o modificació de comentaris	*JPlag* també esborra tots els comentaris en preprocessar, no es tenen en compte
Modificació de noms de variables, mètodes o classes	*JPlag* tampoc té en compte aquests noms
Modificació de valors constants	*JPlag* tampoc té en compte els valors constants

Taula 2.1.1 Modificacions al document original detectables per *JPlag*

Modificació	Problema
Combinació de subexpressions en una sola o viceversa	*JPlag* genera *tokens* per aquestes subexpressions i per tant seran diferents per tots dos documents
Redundància: afegir mètodes o variables no usats	S'afegeixen *tokens* entre mig d'una seqüència d'ells que ocasiona la no coincidència
Reestructuració del codi en diferents classes o canvi de classes per altres equivalents	Això ocasiona molts canvis al codi indetectables fins i tot per una persona

Taula 2.1.2 Modificacions amb possible èxit vers *JPlag*

2.2 *MOSS*

MOSS [10] és un altre programa de detecció de plagi en llenguatges de programació. Es fa servir usualment pel llenguatge *C*, encara que funciona també amb altres llenguatges com *C++*, *Java*, *Pascal*, *Ada* i altres [13]. Va ser desenvolupat el 1994 per *Alex Aiken* a la universitat *UC Berkeley*. Les sigles de *MOSS* venen donades pel seu nom en anglès, *Measure Of Software Similarity*, que ja indica que es basa també, com *JPlag*, en diverses mètriques de similitud entre documents.

Per fer servir aquest programari també és necessari demanar un compte d'usuari [14]. El servidor proporciona un *script* en llenguatge *Perl* executable en principi a sistemes *Unix* que disposin de *Perl*, *UUEncode*, *mail* i algun programa de compressió de fitxer, ja sigui *zip* o *tar*. També existeix la possibilitat d'executar *MOSS* en entorns *Windows* amb una versió de l'*script* per aquest Sistema Operatiu.

Aquest programari també té disponibles dues versions. La primera funciona enviant per correu electrònic a un servidor els documents que s'han de processar i esperant la resposta amb els resultats. La segona funciona de manera similar però amb una connexió directa al servidor en lloc de mitjançant correu electrònic.

En executar el programa amb els paràmetres adequats, s'envien els documents a comparar al servidor de l'aplicació *MOSS*, ubicat a Berkeley. El servidor retorna llavors un *link* a una pàgina *web* que proporciona els resultats de la comparació i que estarà disponible durant els 14 dies següents. El temps de resposta, òbviament, canviarà en funció de la quantitat i longitud dels documents enviats, però podria ser fins i tot el mateix dia de l'enviament per un conjunt de programes estàndard d'una avaluació continuada. A la figura 2.2.1, extreta de [11], es mostra un exemple de la pàgina web amb els resultats.

---

[13] La llista completa de llenguatges que suporta és *C*, *C++*, *Java*, *C#*, *Python*, *Visual Basic*, *Javascript*, *FORTRAN*, *ML*, *Haskell*, *Lisp*, *Scheme*, *Pascal*, *Modula2*, *Ada*, *Perl*, *TCL*, *Matlab*, *VHDL*, *Verilog*, *Spice*, ensamblador *MIPS assembly*, ensamblador *a8086*, *HCL2*

[14] A l'adreça moss-request@cs.berkeley.edu

Moss Results

Sun Mar 14 15:24:02 PST 1999

Options -l c -m 10

[Text Report | How to Read the Results | Tips | FAQ | Contact Moss | Submission Scripts | Credits]

File 1	File 2	Tokens Matched	Lines Matched
mike_wolf.c (79%)	mike_fox.c (80%)	463	139
bill_smyth.c (86%)	bill_smith.c (88%)	456	133
jane_white.c (59%)	jane_blanco.c (68%)	354	111
john_doe.c (100%)	john_deer.c (100%)	220	49

Any errors encountered during this query are listed below.

Figura 2.2.1 Resultats de la comparació amb *MOSS*

Aquesta figura mostra els parells de documents sospitosos de plagi, proporcionant el número de *tokens* i línies coincidents entre els documents, així com el percentatge de solapament entre ells. Es podria considerar que un solapament del 50% o més seria un indicador de plagi, encara que com és natural –a no ser que es tracti d'un percentatge del 90% o més- ha de ser un avaluador humà qui finalment dictamini si certament es tracta de plagi o no.

Com es veu a la Figura 2.2.2, es pot mostrar el detall de les similituds entre dos documents de la mateixa manera que es podia fer amb *JPlag* a l'apartat 2.1, fent *click* a sobre d'ells. A més, aquesta informació es proporciona amb codis de color, que fan més fàcil encara la seva revisió.

2.2.1 Algoritme emprat

És conegut que *MOSS* efectua optimitzacions sobre alguns dels algoritmes més usuals. L'autor, però, no vol explicitar de quins consta o quines són aquestes optimitzacions. Aquesta informació es pot preservar perfectament ja que, com s'ha explicat anteriorment, el procés de comparació dels documents s'efectua sempre a la part servidora de l'aplicació. Per la informació presentada a la pàgina principal del producte es dedueix, però, que l'algoritme principal emprat en la detecció podria ser el *WINNOWING*, estudiat més endavant a l'apartat 3.2.

2.2.2 Modificacions detectables

Tal i com es va explicar a l'apartat 2.1 (*JPlag*), *MOSS* és també capaç de detectar certs canvis no trivials als documents. Accions com canviar l'ordre d'aparició de les funcions, el text dels comentaris o el nom de totes les variables no suposarien cap diferència en la detecció.

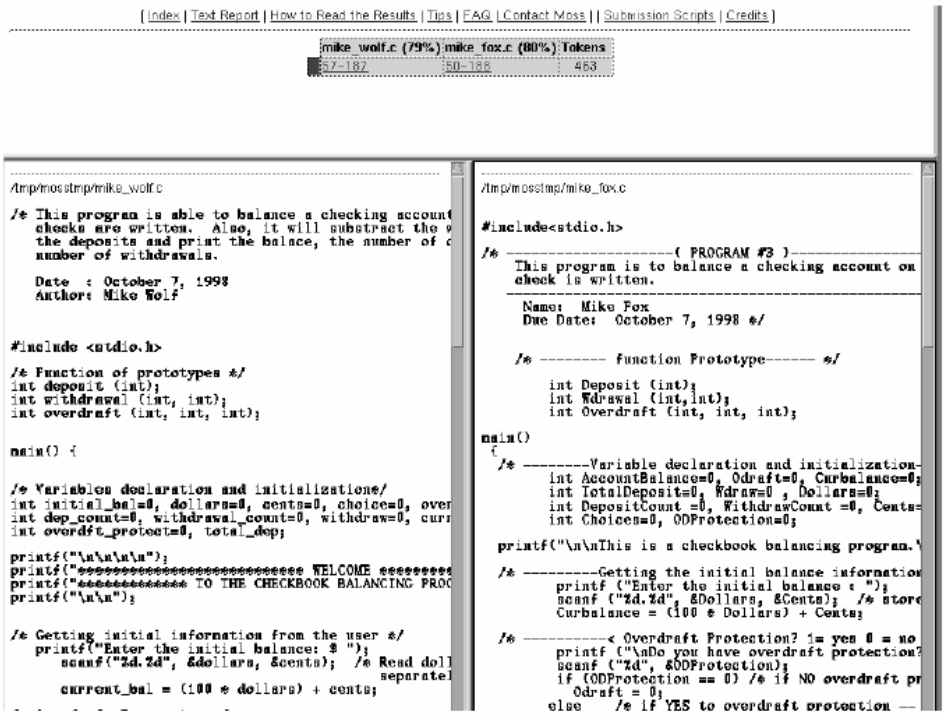


Figura 2.2.2 Detall de similituds amb *MOSS*

2.2.3 Punts febles

Com es veurà a l'apartat 3, *k* és un paràmetre d'alguns algoritmes de detecció de plagi que indica la longitud mínima en *tokens* que han de tenir les coincidències entre documents perquè siguin considerades. El punt més feble de *MOSS* és el fet que un canvi en totes les cadenes del document de menys de *k tokens*, en conseqüència, farà del tot impossible la detecció.

2.3 *YAP*

YAP és una altra aplicació de detecció de plagi en codi de programari que també es pot aplicar a altres tipus de documents en llenguatge anglès. El seu nom ve de les sigles de "*Yet Another Plague*", que traduït significaria una cosa com "una altra plaga". Aquest nom deriva del fet que *YAP* és una evolució d'un altre programa per detecció de plagi més antic, anomenat simplement "*Plague*" [15]. El nom *Plague* ve de *PLAGiarism in University Environments* i es va crear precisament per detecció de plagi entre documents d'alumnes i entorns universitaris.

[15] http://www.csse.monash.edu.au/projects/plague/

El programari *YAP* ha estat evolucionant fins trobar-se ara a la seva tercera versió, anomenada simplement *YAP3*. En aquesta darrera versió es fa servir l'algoritme *Running Karp-Rabin Greedy String-Tiling* (*RKR-GST*), que tal i com s'explica a l'apartat 3.1 és una variació de l'algoritme bàsic *Greedy String-Tiling*.

A [13] es pot veure el procés de detecció. El primer pas que efectua, com és habitual, és un preprocessament que treu comentaris, constants de cadena, converteix tot el text a minúscules i associant el màxim possible diferents expressions amb una comuna que les representi a totes, reordenant les funcions en el mateix ordre que es criden i eliminant qualsevol *token* que no pertany al domini del llenguatge processat. Després, aplica l'algoritme *RKR-GST* per trobar la cadena de *tokens* coincidents més llarga. Un cop trobada, marca els *tokens* com utilitzats i busca una altra cadena coincident menor.

Com s'ha comentat, la diferència entre aquesta i les altres aplicacions esmentades fins ara és que *YAP* concentra més esforços en la detecció de plagi en llenguatge natural. Com s'explicava a l'apartat 1.2, és molt difícil interpretar artificialment el llenguatge natural amb la tecnologia actual, això sense comptar l'ampli vocabulari i riquesa gramatical que proporciona aquest tipus de llenguatge. Així, el primer que fa *YAP* és processar el màxim possible de documents en el llenguatge de destí per d'aquesta manera determinar els *tokens* que s'empraran al següent pas. Posteriorment s'eliminen les 150 paraules més comunes del llenguatge, paraules molt curtes (d'una o dues lletres) i noms propis.

Un cop fet això, es fa servir una adaptació de l'aplicació reconeixedora de llenguatge *PC-Kimmo* juntament amb el conjunt lèxic i de regles gramaticals *Englex10*, ambdós disponibles a la web del *Consortium for Lexical Research* (*CLR*) [16]. Amb aquest pas es fa un pseudo-reconeixement del llenguatge natural que permet també la detecció d'un possible plagi.

2.4 ALTRES APLICACIONS

Les eines per detecció de plagi no són un tema nou. Ja s'ha estudiat i desenvolupat molt en aquest camp, des de la comanda bàsica *diff* [17] dels sistemes operatius *UNIX*, que reporta els canvis necessaris per convertir un fitxer *A* en un altre fitxer *B*, fins a altres més actuals i potents, com *YAP3* vista a l'apartat anterior. Algunes d'aquestes eines són gratuïtes i d'altres de pagament. Aquestes aplicacions, com s'ha vist, poden concentrar els seus esforços en la detecció a codi font o al llenguatge natural, encara que també n'hi ha algunes que poden funcionar amb un èxit relatiu amb tots dos tipus de llenguatge. En alguns casos només treballen amb els documents que se li proporcionen i en altres les mateixes aplicacions busquen més recursos a servidors o a l'Internet.

A continuació es mostra, a la Taula 2.4.1, algunes de les altres aplicacions existents per detecció de plagi a l'actualitat, així com les seves fonts d'informació:

---

[16] http://clr.nmsu.edu/
[17] Veure http://unixhelp.ed.ac.uk/CGI/man-cgi?diff

APLICACIÓ	RECURS
CopyCatch Gold	http://www.cebe.heacademy.ac.uk/learning/Plagiarism/index.php
Sherlock	http://www.cs.usyd.edu.au/~scilect/sherlock/
EVE2	http://www.canexus.com/eve/index.shtml/
MyDropBox	http://www.mydropbox.com/
TurnItIn	http://www.turnitin.com/

Taula 2.4.1 Altres aplicacions per detecció de plagi

3. ALGORITMES USUALS

3.1 ALGORITME *GREEDY STRING-TILING*

Com es va explicar a l'apartat 2.1, l'algoritme *Greedy String-Tiling* (*GST*) busca seqüencialment cadenes de *tokens* coincidents entre els documents, començant per la cadena més llarga possible. Un cop trobada, es marquen els *tokens* inclosos en la coincidència perquè no es tinguin en compte a la següent iteració.

Aquest procés fa que les cadenes coincidents cada cop siguin més curtes ja que a cada iteració del procés hi ha menys *tokens* disponibles per fer la comparació, assegurant la finalització del bucle. A més, d'aquesta manera també s'assegura que, en cas que no hi hagi plagi entre dos documents, la cadena més llarga coincident serà de fet molt curta, podent descartar automàticament els documents implicats. És més, es defineix també el que s'anomena *Minimum Match Length* (*MML*) que, com el seu nom indica, és la longitud mínima de coincidència de cadenes entre dos documents per tal que siguin possibles candidats de plagi al resultat final. Així, totes les cadenes coincidents amb una longitud inferior al *MML* no es tindran en compte.

A la taula 3.1.1 es mostra el pseudo-codi corresponent a l'algoritme *GST*, extret de [8].

En quant al cost computacional de l'algoritme, òbviament és més costosa la cerca de coincidències entre cadenes de *tokens* que el marcatge posterior dels *tokens* utilitzats. L'algoritme *Karp-Rabin* [9], però, proporciona una optimització coneguda com *RKR-GST* (*Running Karp-Rabin Greedy String-Tiling*) que consta, segons s'explica a [8], dels següents passos:

- Primer es crea un resum o *hash* de cada subcadena amb longitud igual al *MML* d'ambdós documents *A* i *B*. Això implica afegir un cost addicional –encara que petit- al procés, que correspon a $O(|A| + |B|)$

- Aquest cost addicional, però, implica un gran benefici en el següent pas, més encara mentre més gran és l'*MML*, ja que la comparació s'efectua entre els *hash* de les subcadenes i no entre les subcadenes en sí. Només s'efectua una comparació exhaustiva entre els texts originals *token* a *token* si els *hash* de totes dues són idèntics, ja que això indica una possible coincidència entre subcadenes que comencin amb aquell *token*

- Per agilitzar la cerca de coincidències, es crea una taula de *hash* que es fa servir per la localització de subcadenes a *B* amb el mateix *hash* que s'està cercant

- Quan es troba una coincidència entre el *hash* d'un *token a* del document *A* i el *hash* d'un *token b* del document *B*, s'inicia la comparació *token* a *token* entre les cadenes originals però es fa en ordre invers a partir del *token a* + *maxmatch*. Això és així perquè tal i com indica l'algoritme interessa trobar una cadena coincident amb una longitud de com a mínim *maxmatch*, que és la longitud màxima trobada a la iteració anterior. D'aquesta manera també s'optimitza la cerca, ja que en cas que el *token a* + *maxmatch* no sigui idèntic a totes dues cadenes significa que la cadena coincident serà més curta i per tant s'interromp la cerca per passar a la següent iteració

```
0  Greedy-String-Tiling(String A, String B) {
1          tiles = {};
2          do {
3                  maxmatch = MinimumMatchLength;
4                  matches = {};
5                  Forall unmarked tokens A_a in A
6                          Forall unmarked tokens B_b in B
7                                  j = 0;
8                                  while (A_{a+j} = = B_{b+j} &&
9                                          unmarked(A_{a+j}) && unmarked(B_{b+j}))
10                                         j ++;
11                                 if (j = = maxmatch)
12                                         matches = matches ⊕ match(a, b, j);
13                                 else if (j > maxmatch) {
14                                         matches = {match(a, b, j)};
15                                         maxmatch = j;
16                                 }
17                         }
18                 }
19                 Forall match(a, b, maxmatch) ∈ matches {
20                         For j = 0 . . . (maxmatch – 1) {
21                                 mark(A_{a+j});
22                                 mark(B_{b+j});
23                         }
24                         tiles = tiles ∪ match(a, b, maxmatch);
25                 }
26         } while (maxmatch > MinimumMatchLength);
27         return tiles;
28 }
```

Taula 3.1.1 Pseudo-codi de l'algoritme *Greedy String Tiling*

Quan s'aplica l'algoritme *Karp-Rabin* per optimitzar l'algoritme *Greedy String-Tiling*, l'algoritme resultant es coneix com *Running-Karp-Rabin Greedy-String-Tiling* (*RKR-GST*), explicat amb més profunditat a [13].

3.2 ALGORITME *WINNOWING*

Aquest algoritme funciona, bàsicament, de la mateixa manera que el *Greedy String-Tiling* vist a l'apartat anterior, és a dir mitjançant la comparació de *hash*. La diferència és la manera de calcular els *hash* i l'àmbit dels mateixos.

En el cas de l'algoritme *WINNOWING*, explicat a [12], es genera el que s'anomena *fingerprints* del document, que no és més que una selecció d'alguns *hash* de diferents

fragments del document. El que es fa és, donats determinats paràmetres per part de l'usuari, construir fragments del document, generar un seguit de *hash* d'aquests fragments i d'entre ells triar segons unes regles els que configuraran el *fingerprint* del document sencer.

Els paràmetres inicials necessaris són *t* (*guarantee threshold*), que indica la longitud mínima de cadena plagiada que es garanteix que es trobarà (en cas que existeixi) i *k* (*noise threshold*), que indica la longitud mínima que han de tenir les coincidències perquè es tinguin en compte.

Amb els paràmetres inicials es calcula el paràmetre *w*, corresponent a la mida de *finestra* o *window*, i que és igual a $w = t - k + 1$. Aquestes *finestres* defineixen de quina manera s'agrupen els *hash*.

Bucle principal de *WINNOWING*, extret de [12]:

```
0  void winnow(int w /*window size*/) {
1          // circular buffer implementing window of size w
2          hash_t h[w];
3          for (int i=0; i<w; ++i) h[i] = INT_MAX;
4          int r = 0; // window right end
5          int min = 0; // index of minimum hash
6          // At the end of each iteration, min holds the
7          // position of the rightmost minimal hash in the
8          // current window. record(x) is called only the
9          // first time an instance of x is selected as the
10         // rightmost minimal hash of a window.
11         while (true) {
12                 r = (r + 1) % w; // shift the window by one
13                 h[r] = next_hash(); // and add one new hash
14                 if (min == r) {
15                         // The previous minimum is no longer in this
16                         // window. Scan h leftward starting from r
17                         // for the rightmost minimal hash. Note min
18                         // starts with the index of the rightmost
19                         // hash.
20                         for(int i=(r-1)%w; i!=r; i=(i-1+w)%w)
21                                 if (h[i] < h[min]) min = i;
22                         record(h[min], global_pos(min, r, w));
23                 } else {
24                         // Otherwise, the previous minimum is still in
25                         // this window. Compare against the new value
26                         // and update min if necessary.
27                         if (h[r] <= h[min]) { // (*)
```

```
                    min = r;
                    record(h[min], global_pos(min, r, w));
                }
            }
        }
}
```

Taula 3.2.1 Pseudo-codi de l'algoritme *WINNOWING*

3.3 DISTÀNCIA DE LEVENSHTEIN

També coneguda com *Distància d'Edició*, la *Distància de Levenshtein* [14] és u*na* altra manera diferent de buscar similituds entre dos documents. El seu nom es deu al científic rus *Vladimir Levenshtein*, que el va concebre l'any 1965. Aquesta distància, bàsicament, es refereix a la quantitat de canvis que s'ha d'efectuar a un document *A* per tal que sigui idèntic a un altre document *B*. Aquestes distàncies podrien ser, per exemple, els esborrats, insercions o substitucions necessaris al text original perquè passi a ser igual al text estudiat.

La detecció de plagi en documents és només una de les aplicacions d'aquest algoritme, ja que també s'utilitza en altres camps com correcció ortogràfica o reconeixement de la parla.

Mentre més gran és la diferència entre dos documents, major serà el valor de la distància de *Levenshtein*. És a dir, si dos documents són completament idèntics, la distància de *Levenshtein* seria igual a 0. Sovint s'anomena *cost* a aquest valor resultant.

El principal problema d'aquest algoritme és la reordenació del contingut dels documents. Això, que en principi pot semblar una modificació molt trivial, implica un gran canvi en la distància d'edició d'un document vers un altre. Així, encara que el contingut de dos documents sigui pràcticament idèntic, una diferent disposició dels paràgrafs ocasionaria una distància d'edició molt gran i el plagi no seria detectat.

A continuació s'especifica l'algoritme pel càlcul de la distància, tal i com es presenta a [14]. S'ha de tenir en compte que *s* i *t* són els documents a comparar.

1	· n = longitud (s) · m = longitud (t) · si $n = 0$, retornar m i acabar · si $m = 0$, retornar n i acabar · construir una matriu amb 0..m files i 0..n columnes
2	· inicialitzar la primera fila a 0..n · inicialitzar la primera columna a 0..m

3	· examinar cada caràcter de *s* (*i* de 1 a *n*)
4	· examinar cada caràcter de *t* (*j* de 1 a *m*)
5	· si *s*[*i*] = *t*[*j*], el resultat és 0 · si *s*[*i*] <> *t*[*j*], el resultat és 1
6	· assignar a la cel·la *d*[*i*, *j*] de la matriu el mínim de: a) La cel·la immediatament superior + 1: *d* [*i* -1, *j*] + 1 b) La cel·la immediatament a l'esquerra + 1: *d* [*i*, *j* - 1] + 1 c) La cel·la diagonal superior esquerra + cost: *d* [*i* – 1, *j* – 1] + cost
7	· en acabar la iteració (3, 4, 5, 6), la distància es troba a la cel·la *d* [*n*, *m*]

Taula 3.3.1 Passos per calcular la distància de *Levenshtein*

4. *OPENOFFICE.ORG XML*

Com s'ha comentat a la introducció d'aquesta memòria, la part pràctica del PFC consisteix en la detecció de plagi entre diferents arxius en format *OpenOffice.org*.

OpenOffice.org és una eina ofimàtica gratuïta, multi-plataforma i de codi obert. Els arxius creats amb aquesta aplicació mantenen l'estàndard de representació de dades *XML*, que els fa portables a qualsevol plataforma.

En aquest apartat s'estudiarà el format d'arxiu *OpenOffice.org XML* i de quina manera es pot extreure informació interessant per la detecció de plagi.

4.1 *OPENOFFICE.ORG*

OpenOffice.org [18] és una eina que agrupa diverses utilitats ofimàtiques. Aquesta aplicació s'incorpora a dins del marc del projecte *open-source* [19], és a dir, és un programari en què hom pot accedir al seu codi font.

En un primer moment es va decidir anomenar aquest programari simplement *OpenOffice* però degut a problemes amb marques ja registrades es va haver de canviar per l'actual *OpenOffice.org*.

Les aplicacions de què consta la versió 2.0, última versió estable d'*OpenOffice.org* en el moment de l'elaboració d'aquest PFC, són:

- Processador de texts *Writer* [20]
- Dissenyador de presentacions *Impress* [21]
- Full de càlcul *Calc* [22]
- Base de dades *Base* [23]
- Creador de funcions matemàtiques *Math* [24]
- Eina de dibuix *Draw* [25]

---

[18] http://www.OpenOffice.org/
[19] http://www.opensource.org/docs/definition.php
[20] http://www.OpenOffice.org/product/writer.html
[21] http://www.OpenOffice.org/product/impress.html
[22] http://www.OpenOffice.org/product/calc.html
[23] http://www.OpenOffice.org/product/base.html
[24] http://www.OpenOffice.org/product/math.html
[25] http://www.OpenOffice.org/product/draw.html

4.1.1 Utilització d'*OpenOffice.org* a la *UOC*

Un dels avantatges que té aquest conjunt d'aplicacions és el fet de ser gratuït. Això fa que sigui un competidor directe del conjunt d'aplicacions ofimàtiques més estès al mercat, *Microsoft Office* [26], ja que a més les eines que ofereix *OpenOffice.org* són bàsicament les mateixes.

No és d'estranyar, doncs, que una gran quantitat d'alumnes de la *UOC* desenvolupi les proves d'avaluació continuada amb aquest programari, més quan és un dels programaris estàndard utilitzats per la *UOC*. Degut a això, es fa necessària la implementació d'una utilitat de control de plagi capaç de processar la informació continguda en aquest format d'arxiu.

4.1.2 Els arxius *OpenOffice.org*

Les aplicacions incloses a *OpenOffice.org* comparteixen una cosa en comú: l'estàndard de representació de dades *XML* [27]. D'aquesta manera, tant un arxiu de base de dades com un document de text representaran internament les dades, *metadades* i contingut en una estructura *XML* molt fàcil d'interpretar en qualsevol plataforma i mitjançant qualsevol llenguatge de programació. Això, juntament amb l'àmplia informació que es pot trobar a *Internet* sobre aquest programari, són avantatges importants a l'hora d'implementar una utilitat que processi aquestes dades, vers altres aplicacions propietàries com *Microsoft Office* de les que per exemple no és fàcil trobar informació tècnica.

En aquest PFC s'estudia només els documents *OpenOffice.org* de text. Al següent apartat s'explica el seu format d'arxiu *XML* i de quina manera s'hi pot treure profit en la detecció de plagi.

4.2 EL FORMAT D'ARXIU *OPENOFFICE.ORG XML*

Com s'ha vist a l'apartat anterior, els arxius *OpenOffice.org* s'emmagatzemen en format *XML*. Això comporta molts avantatges tant pel que respecta a portabilitat, ja que el *metallenguatge XML* és el format més usual d'intercanvi de dades entre diferents plataformes, com per la facilitat de procés que proporciona aquest *metallenguatge*.

Els documents *OpenOffice.org* tenen l'extensió *sxw*. Aquests documents, però, són en realitat un conjunt de subdocuments comprimits a dins de l'arxiu *sxw* mitjançant el conegut format *zip*. Cadascun d'aquests arxius conté, al seu torn, un tipus diferent d'informació.

Aprofundint en el tema central d'aquest PFC, hi ha dos subdocuments dels documents *OpenOffice.org* interessants per la detecció de plagi: les *metadades* i el contingut en sí mateix. A continuació s'explica cadascun d'aquests subdocuments.

---

[26] http://office.microsoft.com/en-us/default.aspx
[27] http://www.w3.org/XML/

4.2.1 *Metadades*: el subdocument *meta.xml*

Les *metadades* proporcionen informació general sobre el document. Aquestes dades poden ser estadístiques com el número de pàgines del document o descriptives, com per exemple el nom de la persona que el va imprimir per últim cop.

Les *metadades* d'un document *OpenOffice.org* estan incloses al subdocument *meta.xml*, contingut a l'arxiu principal *sxw*. Totes les *metadades* d'un document *OpenOffice.org* estan contingudes dins de l'element <*office:meta*>, que normalment es troba a l'inici del document. Algunes d'aquestes dades no són importants per la detecció de plagi i d'altres, en canvi, són molt significatives com per exemple el nom de la persona que va crear el document. Tot seguit es mostra, a la llista 4.2.1, les diferents *metadades* que s'empraran com a base per la detecció de plagi a l'aplicatiu a desenvolupar.

Metadada	**Significat**
Creator	Nom de la persona que va modificar per últim cop el document
Date	Data en què es va modificar el document per última vegada
editing-cycles	Número de vegades que s'ha modificat el document
editing-duration	Temps total que s'ha estat editant el document

Taula 4.2.1 *Metadades* que s'empraran per detectar plagi

Altres *metadades* també incloses en aquest document, encara que no es faran servir per a la detecció de plagi en l'aplicació desenvolupada, són per exemple una referència al document que es va fer servir com a *template* per crear aquest document, (element <*meta:template*>, un conjunt de paraules clau sobre el mateix (element <*meta:keywords*>) o l'aplicació que es va fer servir per generar el document (element <*meta:generator*>).

4.2.2 Contingut: el subdocument *content.xml*

S'entendrà per contingut el text, imatges o gràfics que veu l'usuari quan obre un document. Això, lògicament, és un element clau en la determinació de si es produeix o no plagi entre dos o més documents.

El contingut d'un document de text *OpenOffice.org*, generalment, es troba al subdocument *content.xml* que és un dels altres arxius de què consta l'arxiu principal comprimit *sxw*. Les imatges no s'emmagatzemen en aquest subdocument sinó que ho fan a la carpeta *Pictures*, també inclosa al mateix arxiu comprimit. La manera d'accedir a aquestes imatges és mitjançant referències, de manera similar a com es fa al llenguatge *HTML*. Altres continguts en format binari, com per exemple arxius de so, també s'emmagatzemen al fitxer comprimit com a fitxers separats a no ser que siguin vinculats, cas en què només s'emmagatzemen en format de referència externa.

Encara que els estils de text estan emmagatzemats al subdocument *styles.xml*, el subdocument *content.xml* també inclou una sèrie d'estils automàtics que s'apliquen al contingut, com per exemple les fonts emprades en la indentació de títols i paràgrafs.

Encara que, com s'ha vist, l'arxiu *content.xml* conté quelcom més que el simple text del document, en el cas que ocupa aquest PFC la comparació consistirà bàsicament en extreure les cadenes de text d'aquests documents i comparar-les dos a dos. Ara bé, com es veurà a l'apartat 5.2, un dels punts més importants en la comparació de cadenes de text és el preprocessament de les mateixes per tal d'eliminar el major número possible d'intents d'amagar el plagi.

5. CONSTRUCCIÓ DEL PROGRAMARI

Fins el moment s'ha estat explicant en què consisteix el plagi i quines són les maneres en què es pot cometre. També s'ha vist en què consisteix el programari *OpenOffice.org* i quina és la manera en què s'emmagatzemen i es poden recuperar les dades dels arxius en aquest format. En aquest capítol es treu profit de tota aquesta informació, analitzant de quina manera desenvolupar una aplicació que ajudi a detectar el plagi en documents en format *OpenOffice.org*.

Primer s'analitzaran els requeriments de l'aplicació i el preprocessament dels documents. Després s'estudia el model de dades necessari per aconseguir els objectius definits. Finalment, s'explica amb profunditat els aspectes de l'implementació i el funcionament del programa desenvolupat.

5.1 ANÀLISI DE REQUERIMENTS

En aquest apartat s'analitzen les característiques que ha de complir el programari a desenvolupar. A grans trets, aquestes són fiabilitat, facilitat d'ús, rapidesa i claredat. A continuació s'explica amb més detall cadascuna d'aquestes característiques i de quina manera es preveu implementar-les a l'aplicació.

5.1.1 Fiabilitat

Queda clar que per facilitar la feina del consultor s'ha d'intentar reduir al mínim la feina que ha de fer per verificar el caràcter individual de les proves presentades pels alumnes de la *UOC*. Així, no té sentit una aplicació que proporcioni falsos positius en aquesta detecció.

En aquest context, es presenten dos inconvenients: el primer, l'aparició de l'enunciat a les respostes dels alumnes. És a dir, si tots els alumnes responen les proves en base a un mateix document, és molt probable que el programari detecti les parts corresponents a l'enunciat en totes les proves com si tots haguessin comès plagi. D'aquesta manera és imprescindible eliminar aquest enunciat de tots els documents a processar. Aquest procés d'eliminació de l'enunciat s'explica amb més detall, degut a la seva importància, a l'apartat 5.2.

El segon inconvenient és el pes dels fragments idèntics detectats. Per exemple, no ha de tenir la mateixa importància una cadena de text idèntica de longitud 5 que una de longitud 50, ja que la primera evidentment pot ser fruit de la casualitat, més encara quan el tema tractat a tots els documents és el mateix. L'aplicació tindrà també aquest fet en compte, permetent a l'usuari especificar quina serà la longitud mínima de coincidència que es tindrà en compte.

5.1.2 Facilitat d'ús

S'ha de tenir present que, encara que l'àrea d'aquest PFC correspon a les tecnologies de la informació, els usuaris finals de l'aplicació resultat no tenen perquè ser usuaris informàtics experts i de fet en la majoria de casos no ho seran.

Això implica que l'*interface* de l'aplicació ha de ser atractiu, còmode, intuïtiu i pràctic per tal que la seva usabilitat sigui l'adequada en aquest context. El més adequat en aquests casos i que és la solució adoptada en el desenvolupament de l'aplicació, és emprar una aplicació gràfica *GUI* (*Graphical User Interface*) vers una aplicació de línia de comandes. Aquest *interface* ha de proporcionar un accés ràpid i fàcil a totes les opcions disponibles.

5.1.3 Rapidesa

L'aplicació a desenvolupar permetrà als consultors de la UOC un estalvi de temps important en un primer examen de les proves per detectar plagi. Així, és necessari que aquesta aplicació tingui un temps de resposta suficientment bo com perquè arribi a ser útil, ja que en cas contrari no serà viable la seva utilització.

Per a complir aquest objectiu, s'optimitzarà al màxim l'aplicació per tal que sigui el més ràpida possible. Els punts més crítics en aquest cas seran l'accés a arxius de disc i, sobretot, la cerca de coincidències entre els documents. Per la cerca s'implementarà un o més dels algoritmes més usuals per la detecció de plagi. En cas que es faci servir més d'un mètode, s'ha de donar la opció a l'usuari d'emprar un o un altre mètode, o bé una combinació dels mateixos en funció de la rapidesa esperada.

L'algoritme emprat en la comparació és el *Running-Karp-Rabin Greedy-String-Tiling*, explicat a l'apartat 3.1 ja que dels estudiats és el que ofereix una relació flexibilitat-rapidesa més gran.

5.1.4 Claredat

Els resultats de l'aplicació han de tenir un format també molt còmode d'interpretar. La informació que s'ha de mostrar com a resultat és molta, ja que el nombre de documents a processar en principi serà alt.

Com a informe de resultats s'ha optat per una solució molt fàcil d'implementar i que proporciona una manera molt senzilla de "navegar" per la informació mostrada: un informe en format *HTML*. Així, serà possible mostrar una pàgina principal de l'informe de resultats en què es mostri a trets bàsics quines són les conclusions generals que s'han trobat en matèria de plagi i, posteriorment, tenir la possibilitat de mostrar el detall dels trets que el consultor trobi més significatius. També s'ha de tenir en compte la possibilitat de, un cop s'està mostrant el detall d'una de les coincidències, poder tornar al document principal. Això és una de les coses que també ens permet el format *HTML*, ja que des del mateix navegador on s'obri es pot tornar enrera al document original.

D'aquesta manera l'informe *HTML* constarà, bàsicament, d'un document principal amb enllaços a altres documents secundaris que mostraran les coincidències amb més detall i el mateix navegador proporcionarà la utilitat de passar d'un als altres.

5.2 PREPROCESSAMENT DEL CONTINGUT TEXTUAL

Tal i com es va explicar a l'apartat 1.2, una de les formes de plagi més usual en l'àmbit acadèmic és la col·lisió, que consisteix en compartir una feina de desenvolupament o investigació entre més d'un alumne quan se suposa que hauria de ser una feina individual. El més usual també en aquests casos és que els alumnes intentin amagar aquest fet d'alguna manera, molt sovint canviant l'ordre de paràgrafs o bé aplicant un diferent format al document perquè no sigui tan obvi que són una còpia exacta. Això inclou una ortografia diferent, afegir espais o línies en blanc, etc.. Per aquest motiu es fa necessari un preprocés dels documents abans de comparar-los.

Aquest preprocés consisteix, bàsicament, en canonitzar els documents perquè tots segueixin unes normes comunes per tal de minimitzar l'intent d'amagar el plagi. Així, la canonització implicarà eliminar espais en blanc, accents, majúscules, signes especials, etc.. A la llista 5.2.2 es pot veure quines d'aquestes característiques es tindran en compte a l'hora de canonitzar els documents.

Contingut	Símbols	Procés
Caràcter blanc o espai	“ “	S'eliminarà qualsevol d'aquests caràcters
Punt	“.”	
Coma	“,”	
Geminació	“·”	
Apòstrof	“’”	
Accent tancat i obert	“´”, “`”	
Cometes dobles	“””	
Signe d'interrogació obert i tancat	“¿”, “?”	
Signe d'admiració obert i tancat	“¡”, “!”	
Parèntesi obert i tancat	“(“, “)”	
Claudàtor obert i tancat	“[“, “]”	
Clau esquerra i dreta	“{“, “}”	
Dos punts	“:”	
Punt i coma	“;”	
Guió	“-“	
Asterisc	“*”	
Barra vertical, obliqua i invertida	“|”, “/”, “\”	
Caràcters amb accent o dièresi	“à”, “á”, “è”, “é”, “í”, “ï”, “ò”, “ó”, “ú”, “ü”	Es substituiran pel mateix caràcter sense accent o dièresi

Taula 5.2.2 Canonització dels documents

5.2.1 Eliminació de l'enunciat

Aquest és un punt important en la detecció de plagi respecte el tema tractat en aquest PFC. Com s'ha explicat a l'apartat 5.1, els alumnes de la *UOC* acostumen a respondre les proves d'avaluació continuada al mateix enunciat que se'ls ha lliurat, bé per iniciativa pròpia o bé perquè així està estipulat a la mateixa prova. El fet que el mateix

enunciat apareix a les respostes dels alumnes podria provocar que es detectés com a còpia algun fragment quan en realitat correspon a l'enunciat.

Per solucionar aquest problema, una part del processament ha de ser la eliminació de la part corresponent a l'enunciat als documents que s'han de processar. Això es farà d'una forma molt similar a la manera en què es trobaran els *matches* entre els documents, tal i com es veurà al següent punt, a l'apartat "La classe *OpenOfficeProcessor*". Bàsicament, es farà servir també l'algoritme *Greedy String-Tiling* per eliminar progressivament diferents fragments de l'enunciat fins que no en quedi cap o bé el fragment resultant sigui negligible.

5.3 MODEL DE DADES

En aquest apartat s'estudia quin és el model de dades apropiat pel correcte funcionament de l'aplicació a desenvolupar.

5.3.1 Recuperació d'informació

Com s'ha explicat a l'apartat 4.2, els documents *OpenOffice.org* són fitxers comprimits que contenen una sèrie de fitxers individuals. Els que interessen per la detecció de plagi són *meta.xml* i *content.xml*.

Així doncs, el primer que farà l'aplicació és obrir tots els documents continguts en un directori determinat i recuperar la informació dels fitxers *meta.xml* i *content.xml*.

5.3.2 Els *Handler XML*

Com s'ha vist, la informació continguda als fitxers *OpenOffice.org* està emmagatzemada en format *XML*. Per aconseguir llegir aquesta informació es fa necessari l'ús d'un *parser XML* que és, bàsicament, un "intèrpret" capaç de manegar informació en format *XML*. Aquest *parser* es pot obtenir amb la classe *org.xml.sax.XMLReader*. Això, però, no és suficient perquè el *parser* per sí mateix no sap què fer amb aquesta informació. Aquí és on entren en joc els *Handler*, que indiquen al *parser* què s'ha de fer amb les dades que ha llegit.

Per l'aplicació a desenvolupar s'haurà de desenvolupar dos *Handler*, un per les *metadades* i un altre pel contingut textual. Tots dos *Handler* heretaran del *Handler* per defecte, *DefaultHandler*, contingut al paquet *org.xml.sax.helpers*.

El primer dels *Handler*, *OpenOfficeXMLMetadadesHandler*, processa les *metadades* que es van mostrar a la taula 4.2.1. El segon, *OpenOfficeXMLContentHandler*, processa el contingut textual del fitxer, aplicant les regles explicades a l'apartat 5.2.

A la Figura 5.3.1 es pot veure un esquema del funcionament del *parser* amb els *Handler* per llegir els documents.

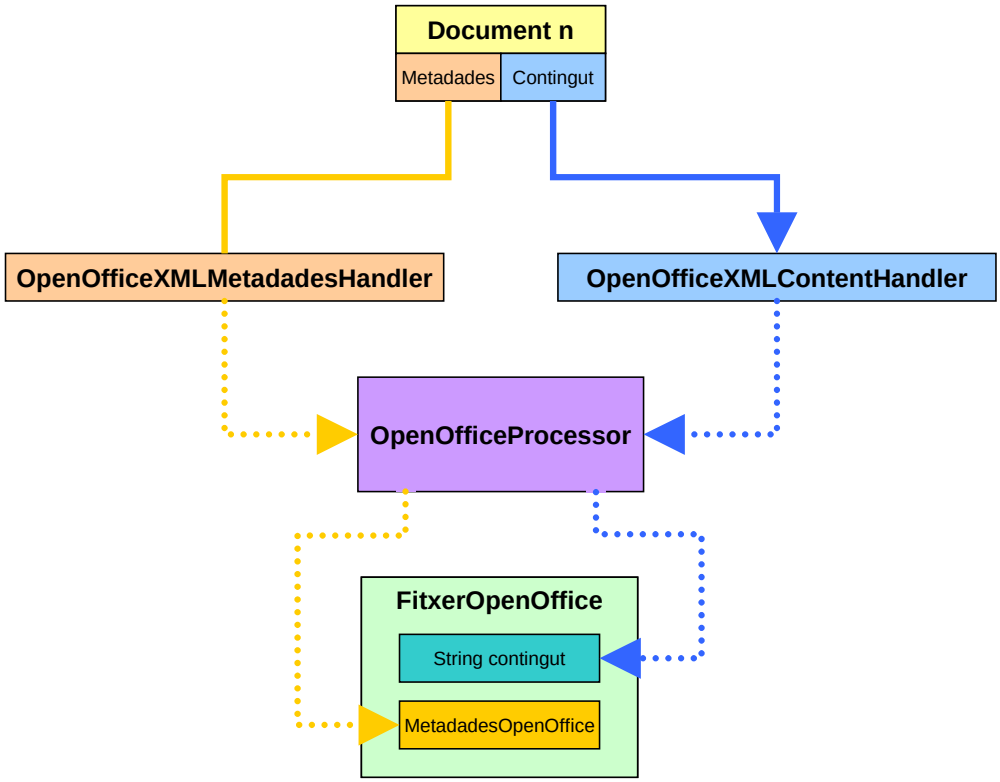


Figura 5.3.1 Funcionament del *parser* amb els *Handler*

5.3.3 Emmagatzemat de la informació

Lògicament, un cop processats els documents s'ha de guardar la informació que contenen. Per aquest motiu es crearan les classes *MetadadesOpenOffice* i *FitxerOpenOffice*.

La classe *MetadadesOpenOffice* no és res més que un parell d'objectes de la classe *Vector* anomenats *propietats* i *valors*, que emmagatzemen respectivament els noms de les propietats que hem processat i el valor corresponent, dos a dos. A més, proporcionen els típics mètodes d'accés de lectura i escriptura.

La classe *FitxerOpenOffice* conté, d'una banda, tres *String* amb el nom del document, el contingut textual original i el mateix un cop preprocessat. D'una altra banda, també conté un objecte de tipus *MetadadesOpenOffice* que s'acaba d'explicar. D'aquesta manera, un objecte *FitxerOpenOffice* proporciona tota la informació necessària per fer la posterior cerca de *matches* i similituds entre els documents.

Per últim, també se li ha afegit una propietat per poder mostrar a l'informe de resultats el text original dels *matches* trobats: el *Vector offsets*. Aquest *Vector* emmagatzema a cadascuna de les seves posicions, a mesura que es preprocessa el document, la quantitat de caràcters que s'han eliminat fins a aquella posició. D'aquesta manera, un cop es troba un *match* entre dos documents, es pot especificar a quina posició exacta

del text original es troba aquell *match* i així mostrar aquest text original a l'informe de resultats en comptes del text preprocessat, que de vegades no és gaire intel·ligible.

5.3.4 *Matches* entre documents: la classe *OpenOfficeMatch*

Aquesta classe és l'encarregada d'emmagatzemar la informació corresponent a un *match* entre dos documents *A* i *B*. Ha de donar suficients dades com per poder construir el text original del presumpte fragment copiat en el moment de crear l'informe de resultats.

Així, en aquesta classe es guarda informació com la posició inicial del *match* al text preprocessat tant del document *A* com del document *B* i la longitud del mateix. A més, també es guarda la posició inicial del *match* al text original sense preprocessar del document *A* i la seva longitud també real. D'aquesta manera es pot mostrar als resultats el text original en comptes del preprocessat, tal i com s'ha explicat al punt anterior.

5.3.5 La classe *OpenOfficeProcessor*

Es pot dir que aquesta és la classe principal de l'aplicació. S'encarrega de controlar tot el procés, des de la lectura dels documents fins a la creació de l'informe de resultats.

Aquest procés el fa en els següents passos:

1. ***recuperarDocuments( )*:** obre els documents i emmagatzema el seu document textual

2. ***eliminarPlantilla(int** enunciat**)*:** elimina de cada document el text corresponent a l'enunciat seleccionat. El paràmetre *enunciat* indica si s'ha seleccionat l'eliminació de l'enunciat (0 - Sí) o no (1 - No) a la pantalla principal

3. ***preprocessarFitxers( )*:** obre els documents, els processa i emmagatzema la informació que contenen

4. ***buscarMatches( )*:** com el seu nom indica, busca tots els *matches* entre dos documents i els emmagatzema

5. ***inicialitzarReport( )*:** prepara la carpeta de destí a on es guardarà l'informe de resultats

6. ***imprimirHTML( )*:** crea l'informe *HTML* amb els resultats del procés

La part més important d'aquest procés és la cerca dels *matches*. Aquesta és la part més crítica en quant a eficiència de l'aplicació, ja que la càrrega de feina en aquest punt serà molt gran, en funció del nombre i longitud dels documents a tractar. Per aquest motiu és on s'ha posat més èmfasi per optimitzar.

Bàsicament, la classe té tres components principals: l'objecte *documents* de la classe *Vector*, l'objecte *allMatches* també de la classe *Vector* i *mml* que és una variable de tipus *int*.

documents és un *Vector* que emmagatzema els objectes *FitxerOpenOffice*. El paràmetre *mml* és un enter que el pot especificar l'usuari i que indica la longitud mínima que ha de tenir una cadena coincident entre dos documents per tal que sigui considerada.

Per últim, *allMatches* és el *Vector* on s'emmagatzemen totes les coincidències entre documents. Cada posició d'aquest *Vector* conté el que es mostra a la Figura 5.3.2.

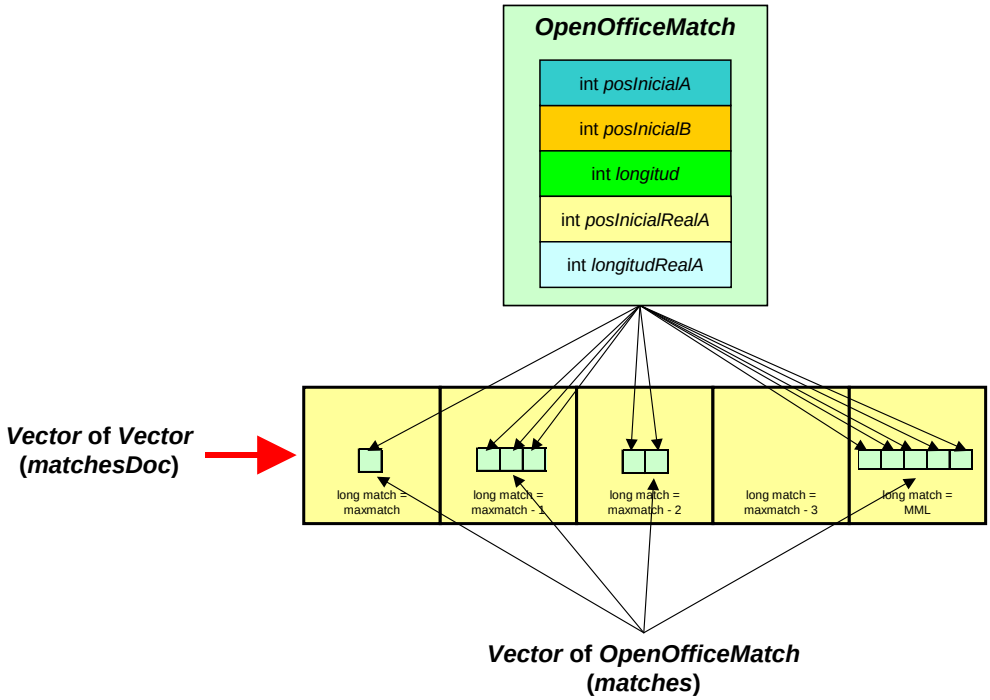


Figura 5.3.2 Contingut de cada posició del *Vector allMatches*

Com es veu, el que es guarda en cada posició del *Vector allMatches* és un altre *Vector*, anomenat *matchesDoc*. *matchesDoc* representa les coincidències existents entre dos documents determinats. Aquest *Vector matchesDoc* conté al seu torn, a cadascuna de les seves posicions, un altre *Vector*, en aquest cas d'objectes *OpenOfficeMatch*, començant pels que tenen longitud *maxmatch* (longitud màxima de cadena coincident) i acabant pels que tenen longitud *mml*. Així, és obvi que el *Vector matchesDoc* tindrà *maxmatch* – *mml* + 1 posicions, havent la possibilitat de que alguna d'aquestes posicions estigui buida si no hi ha coincidències amb aquella longitud, tal i com es mostra a la Figura 5.3.2 per a longitud *maxmatch* - 3.

S'ha considerat la possibilitat d'utilitzar objectes de la classe *Hashtable* en comptes dels *Vector*, ja que els *Hashtable* proporcionen un accés indexat i per tant més ràpid a les dades que emmagatzemen. En aquest cas no està justificat, ja que les dades es guarden als *Vector* exactament en el mateix ordre en què s'han de recollir després per elaborar l'informe de resultats. Així doncs, com que l'accés que es necessita és precisament seqüencial, no cal fer servir més memòria amb la utilització dels *Hashtable*.

La implementació de l'algoritme *RKR-GST* en la funció *scanpattern* compta amb la classe auxiliar *OpenOfficeRKR*, que a cada iteració del bucle principal serveix per retornar un enter amb la longitud màxima trobada i un *Vector* amb els *matches* d'aquella longitud.

També es disposa de la classe *OpenOfficeTile*, que serveix per emmagatzemar cada un dels fragments de longitud *n* d'un determinat document, anomenats *tiles*, necessaris per la implementació de l'algoritme *RKR-GST*.

5.3.6 Optimització del *Vector allMatches*

Com s'ha explicat, el *Vector allMatches* conté les coincidències trobades entre cada parell de documents. Així, si *n* és el nombre de documents a processar, a primera vista es pot pensar que *allMatches* hauria de ser una matriu de *n* * *n* posicions, ja que s'ha de comparar cada document amb tota la resta.

S'ha de tenir en compte, però, un parell de fets que ajuden a reduir el nombre de posicions d'aquesta matriu. Primer, un document no s'ha de comparar amb sí mateix. I segon, si es compara un document *A* amb un document *B*, després no cal tornar a comparar el document *B* amb el document *A*. Això es mostra gràficament a la Figura 5.3.3.

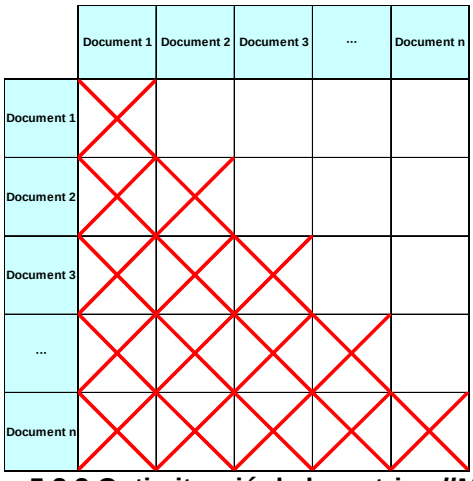


Figura 5.3.3 Optimització de la matriu *allMatches*

A la Figura anterior es veu que en realitat el nombre de posicions necessàries no és n^2 sinó que s'han d'eliminar les posicions corresponents a la diagonal de la matriu i les que queden per sota de la mateixa. Així, les posicions necessàries seran $n^2 - \left(\frac{n(n-1)}{2} + n\right)$. Per tant, en el cas de l'exemple anterior el nombre de posicions serà igual a 10, per *n* = 5.

Per treballar còmodament amb aquesta suposada "matriu", *allMatches* serà en realitat un *Vector* amb la matriu "desplegada" en una única fila, tal i com es mostra a la Figura 5.3.4. Així, donats *n* i dos documents Doc_a i Doc_b, la posició del *Vector* a on s'emmagatzemen les coincidències entre aquests dos documents es calcularà amb la següent fórmula:

$$pos = a \cdot n - \left(\frac{a(a-1)}{2} + a\right) - (n-b)$$

tenint en compte que *a* ha de ser menor que *b*.

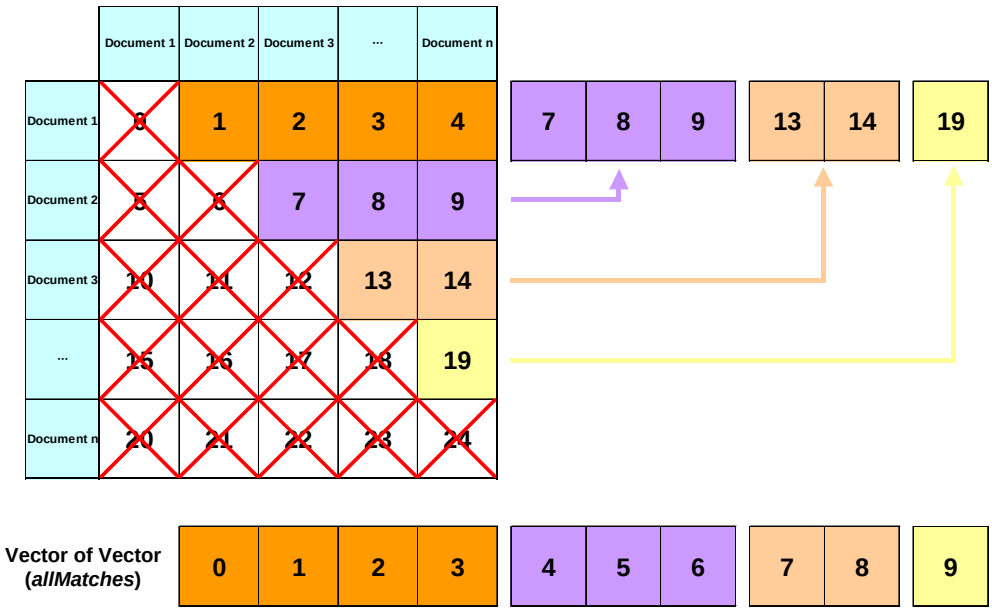


Figura 5.3.4 "Desplegament" de la matriu *allMatches* en un *Vector*

Per exemple, si el total de documents és 5, la posició a on es guardaran les coincidències entre els documents 2 i 4 serà:

$$pos = 2 \cdot 5 - \left(\frac{2(2-1)}{2} + 2\right) - (5-4) = 10 - 3 - 1 = 6$$

i de manera similar, per als documents 4 i 5 serà:

$$pos = 4 \cdot 5 - \left(\frac{4(4-1)}{2} + 4 \right) - (5-5) = 20 - 10 - 0 = 10$$

En aquest punt, però, s'ha de tenir en compte dos fets. Un, que a la posició 0 del *Vector documents* s'emmagatzema l'enunciat de la prova. Així, si *n* és la mida del *Vector documents*, s'haurà de computar com a *n-1* per no tenir en compte l'enunciat. L'altre punt és que la primera posició d'un *Vector* sempre és la posició 0 en comptes de la posició 1. Així, al resultat final se li haurà de restar 1. D'aquesta manera, la fórmula pel càlcul de la posició del *Vector* quedarà, de cara al desenvolupament, com segueix:

$$pos = a \cdot (n-1) - \left(\frac{a(a-1)}{2} + a \right) - ((n-1) - b) - 1$$

5.3.7 Control d'errors

Tota aplicació ha de disposar d'un control d'errors i, en cas d'error, ha de proporcionar una informació intel·ligible a l'usuari. Per aquesta fi s'han creat les classes *OpenOfficeException* i *OpenOfficeMessage*.

OpenOfficeException simplement hereta de la classe bàsica d'error *Exception* per tal de proporcionar un tipus d'error específic per aquesta aplicació. Quan s'invoca aquesta excepció, es passa com a paràmetre el missatge d'error que es desitja mostrar a l'usuari.

En aquest moment es crea un objecte de tipus *OpenOfficeMessage*. Quan es crea una instància de la classe *OpenOfficeMessage*, es mostra, mitjançant la classe *JoptionPane* de la plataforma *Java*, el missatge que es passa com a paràmetre al constructor. D'aquesta manera, a més, si algun dia es desitja canviar l'aparença de l'aplicació només s'hauria de canviar aquesta classe perquè es mostressin els missatges a l'usuari d'alguna altra manera.

5.4 DESENVOLUPAMENT

Bàsicament, l'aplicació a desenvolupar efectuarà quatre passos necessaris per tal de determinar si es produeix plagi entre documents o no. Aquest passos són els següents:

1. Tal i com s'ha vist al capítol 4, els documents *OpenOffice.org* són un conjunt de subdocuments, cadascun amb un contingut específic. Així, caldrà que l'eina a desenvolupar sigui capaç de descomprimir l'arxiu principal per poder accedir al contingut necessari

2. Un cop s'ha aconseguit l'accés als arxius, s'ha de llegir el contingut en format *XML* i emmagatzemar-lo pel posterior procés

3. A continuació, el contingut textual dels documents s'ha de preprocessar per tal de reduir el possible marge d'error en la comparació dels mateixos

4. El següent pas serà la comparació dels documents en sí mateixa. Aquesta s'efectuarà entre cada parell de documents i es guardarà la informació obtinguda

5. Finalment, amb aquesta informació es mostrarà un resum de resultats entre documents per tal que l'usuari de l'aplicació pugui, a simple vista, determinar quins dels documents efectivament cometen plagi i quins d'ells s'haurien d'estudiar més detingudament per determinar si ho fan o no

A continuació es detalla el procés de desenvolupament de l'aplicació.

5.4.1 L'*interface* de l'aplicació: la classe *scrMain*

Un dels requeriments que té l'aplicació, explicats a l'apartat 5.1, és la facilitat d'ús. En aquest sentit, s'ha optat per desenvolupar l'aplicació amb l'ajuda del paquet de classes *javax.Swing*.

javax.Swing és un conjunt de classes especialitzades en el desenvolupament d'interfícies gràfiques. En concret, la classe *scrMain*, que és la pantalla principal de l'aplicació, estén la classe *JFrame* que pertany a aquell paquet.

A la Figura 5.4.1 es mostra aquesta pantalla principal, amb totes les seves opcions.

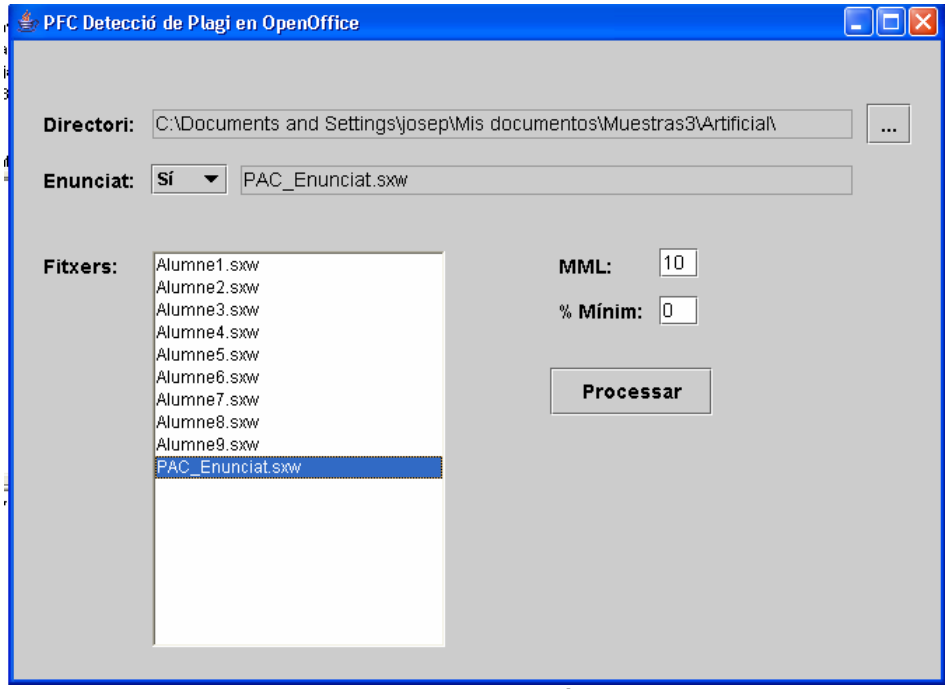


Figura 5.4.1 Pantalla principal de l'aplicació

En aquesta pantalla es veuen les possibilitats de l'aplicació:

- **Directori:** Es mostra quin serà el directori a on estan ubicats els documents que s'han de processar

- **Enunciat:** Hi ha la possibilitat d'indicar si es vol utilitzar un document com a enunciat o no. En cas afirmatiu es pot seleccionar el document enunciat de la prova que conté les parts que s'eliminaran de la resta de documents. Aquest enunciat, per tant, estarà situat a la mateixa carpeta que la resta de documents a comparar i es podrà seleccionar polsant el botó esquerra del ratolí a sobre del fitxer enunciat a la llista de fitxers del directori de treball

- **Fitxers:** Llista amb tots els fitxers *OpenOffice.org* continguts a la carpeta seleccionada com a directori de treball

- ***MML:*** Tal i com s'explica a l'apartat 3.1, correspon al *Minimum Match Length* de l'algoritme *Greedy String Tiling*

- **% Mínim:** Indica el percentatge mínim de similitud entre documents per tal que siguin inclosos a l'informe final de resultats

El directori de treball es pot seleccionar mitjançant un diàleg *JFileChooser*, classe també inclosa al paquet *javax.Swing*, tal i com es veu a la figura 5.4.2.

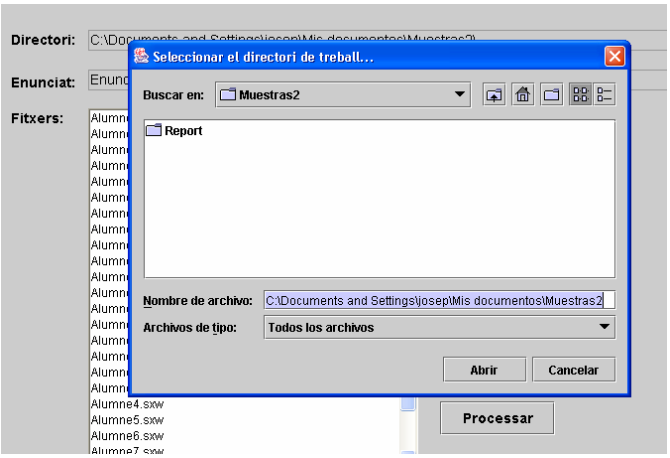


Figura 5.4.2 Diàleg de selecció de directori de treball

Lògicament, aquest diàleg per seleccionar el directori de treball només permet seleccionar directoris. Per si és necessari en línies futures de treball, a més, s'ha creat la classe *OpenOfficeFileFilter*, que el que fa és filtrar els fitxers inclosos en un directori perquè només es mostrin els que tenen extensió ".sxw".

Finalment, el botó *Processar* iniciarà el procés de detecció de plagi.

Per facilitar més l'ús de l'aplicació, s'han afegit etiquetes d'ajuda que apareixen quan es situa el cursor durant un segon o més a sobre d'algun dels elements de la pantalla.

5.4.2 Descompressió dels documents

Com es va veure a l'apartat 4.2, els documents *OpenOffice.org* són fitxers comprimits que contenen una sèrie de subdocuments, com el contingut textual del document o les *metadades* del mateix. El primer pas per poder processar aquests documents serà la descompressió dels documents i posterior recuperació dels subdocuments necessaris.

L'entorn de desenvolupament de *Java* proporciona, dins del paquet *java.util.Zip*, la classe *ZipFile*. Aquesta classe permet tractar fitxers comprimits en el format *zip*, com és el cas que aquí es tracta. Un cop obert el document, la classe *ZipEntry*, inclosa al mateix paquet, permet "navegar" pel contingut del fitxer comprimit, retornant cadascun dels subdocuments.

5.4.3 Lectura del contingut *XML*

L'entorn *Java* també proporciona una sèrie de classes especialitzades en el procés de documents *XML*. En aquest cas, s'emprarà la classe *XMLReader*, inclosa al paquet *org.xml.sax.* Aquesta classe ens proporciona els mètodes necessaris per fer el *parse* o lectura dels documents *XML*.

Una de les propietats que s'han d'assignar a aquest *parser* és el *ContentHandler* que farà servir, tal i com s'explica a l'apartat 5.3. En cas que el subdocument a processar sigui *content.xml*, corresponent al contingut textual del document, es farà servir el *ContentHandler OpenOfficeXMLContentHandler*. En cas que el subdocument correspongui a les *metadades,* es farà servir el *OpenOfficeXMLMetadadesHandler*.

S'ha de tenir en compte, però, que els documents *XML OpenOffice.org*, com la majoria de documents *XML*, especifiquen un document "patró" que han de seguir per tal d'ésser vàlids. Això pot implicar alguns problemes, tal i com es veurà seguidament.

5.4.4 L'*Entity Resolver*: la classe *OpenOfficeNoDTD*

Usualment, com s'acaba d'explicar, els documents *XML* han de seguir un patró de dades definit en un document *DTD* (*Data Type Definition*) i aquí es presenta un problema. El *DTD* és un document físic amb extensió ".dtd" i als documents *XML* s'especifica a on està situat aquest document per tal de validar el contingut *XML*. Així, tots els usuaris que utilitzen aquest document *XML* haurien de tenir accés també al document *DTD*, en cas contrari rebran un error. El problema és que si el document *DTD* es troba a la *web*, és possible que un usuari no hi tingui accés en un moment determinat. O si el document *DTD* està situat en un *path* concret de l'ordinador d'un usuari, aquest document no té perquè estar ubicat al mateix *path* de l'ordinador d'un altre usuari, per exemple en el cas d'aquest PFC, ja que els usuaris poden estar fent servir diferents versions d'*OpenOffice.org* i el nom del *path* en aquest cas seria diferent.

La manera més pràctica de solucionar aquest problema en aquest cas és no utilitzar el *DTD*, ja que es dóna per suposat que tots els documents que s'han de processar estan creats amb el programari *OpenOffice.org* i per tant obligatòriament segueixen el seu *DTD*.

Així, es fa necessari que cada cop que un document *XML* necessita un *DTD*, l'aplicació sigui capaç de respondre amb un *DTD* buit. Això s'aconsegueix mitjançant el que s'anomena *Entity Resolver*, en aquest cas la classe *OpenOfficeNoDTD*. El que fa aquesta classe és precisament això, tornar un *DTD* buit cada cop que en fa falta un. D'aquesta manera es soluciona el problema esmentat i no fa falta tenir disponible el *DTD* en base al qual es va crear els documents *XML* que s'han de processar.

5.4.5 Preprocés dels documents

El preprocés dels documents és totalment necessari per tal d'eliminar el major número possible d'intents d'amagar el plagi, tal i com es va explicar a l'apartat 5.2. Aquest preprocés, però, s'efectua al mateix temps que es fa la lectura del document *XML*, ja que és el mateix *ContentHandler* el que especifica com s'ha de tractar el contingut que s'està llegint.

Així, quan l'aplicació està construint la cadena corresponent al contingut textual del document, ja té en compte aquest preprocessament i retorna directament la cadena de text canonitzada segons les regles definides al *ContentHandler*.

5.4.6 Comparació dels documents

A l'apartat 5.3 ja es va veure de quina manera s'efectua la cerca dels *matches* entre documents. Aquesta és la part central de l'aplicació i, com es va veure, la més crítica ja que és a on s'utilitzen més recursos de maquinari. D'aquesta manera, s'ha intentat optimitzar al màxim per tal que l'execució de l'aplicació sigui suficientment ràpida per tal que sigui útil a l'usuari.

L'algoritme utilitzat en la cerca és el *Running-Karp-Rabin Greedy-String-Tiling*, explicat a l'apartat 3.1. Com es va veure, és una optimització de l'algoritme *Greedy-String-Tiling* que el fa molt més ràpid i eficient.

A l'apartat 5.3 es va justificar el fet d'emprar objectes *Vector* en compte dels *Hashtable*, encara que aquests són molt més eficients. En aquest cas, però, sí que està totalment justificada la utilització dels *Hashtable*, ja que la comparació ha de fer múltiples passades pels fitxers per trobar els *matches* adequats, tal i com es veu a l'algoritme presentat anteriorment.

A més s'ha de tenir en compte que el programa desenvolupat admet moltes millores, que s'explicaran més endavant al capítol 7. La més important d'aquestes podria ser l'aplicació de més d'un algoritme i mostrar el resultat de la comparació en funció de tots dos.

5.4.7 Informe de resultats

Per l'usuari, aquest és un punt crucial pel que fa a utilitat de l'aplicació, ja que en aquest informe és on s'ha de poder "navegar" per la informació resultant de tal manera que sigui ràpid i fàcil trobar qualsevol indici de plagi que s'hagi pogut trobar.

L'informe de resultats es crea automàticament al mateix directori de treball que s'ha seleccionat al principi del procés. L'aplicació crea, en aquest directori, una nova carpeta anomenada "*Report*". En aquesta carpeta es crea un document *HTML* amb el nom "Report.html", que és el document principal de l'informe de resultats i del qual es pot veure una mostra a la Figura 5.4.3.

Les dades que proporciona aquest informe, com es veu a l'esmentada Figura, són:

- **Fitxer A:** Primer dels documents que s'ha comparat
- **Fitxer B:** Segon dels documents, que s'ha comparat amb el Fitxer A
- **% Contingut:** Percentatge de similitud entre el documents A i el document B
- **MaxMatch:** Longitud màxima de *match* trobada entre ambdós documents
- **Metadades:** Resum d'algunes característiques de les *metadades* que podrien ser útils en la detecció de plagi

Comentario: Es podria ordernar el contingut d'aquest fitxer en funció del percentatge de similitud.

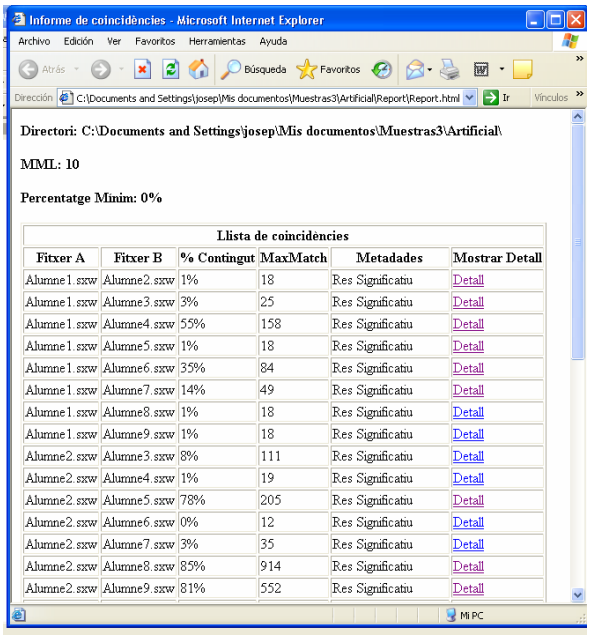
S'ha intentat confeccionar aquest informe amb la major senzillesa possible per tal que l'usuari pugui a simple vista valorar quins dels documents s'han de revisar més a fons. Així, a més, es pot veure que a la dreta de cada parell de documents hi ha un enllaç anomenat "Detall". Mitjançant aquest enllaç es pot accedir al detall de la comparació d'aquell parell de documents.

A la Figura 5.4.4 es pot veure una mostra del detall entre un parell de documents. En aquest document detall es mostra la següent informació:

- **Longitud:** Longitud de la cadena coincident entre els dos documents
- **Text:** Mostra la cadena de text que s'ha trobat a tots dos documents

A més, per proporcionar més utilitat a l'informe, s'ha afegit a la capçalera dos enllaços més que permeten obrir qualsevol dels dos documents implicats en el presumpte plagi. En cas de prémer algun d'aquests enllaços, el document en qüestió s'obrirà en una nova finestra.

Per tornar al document principal de l'informe des del document de detall es pot prémer el botó "Enrere" del mateix navegador.



Informe de coincidències - Microsoft Internet Explorer

Archivo Edición Ver Favoritos Herramientas Ayuda

Atrás Búsqueda Favoritos

Dirección C:\Documents and Settings\josep\Mis documentos\Muestras3\Artificial\Report\Report.html Ir Vínculos

Directori: C:\Documents and Settings\josep\Mis documentos\Muestras3\Artificial

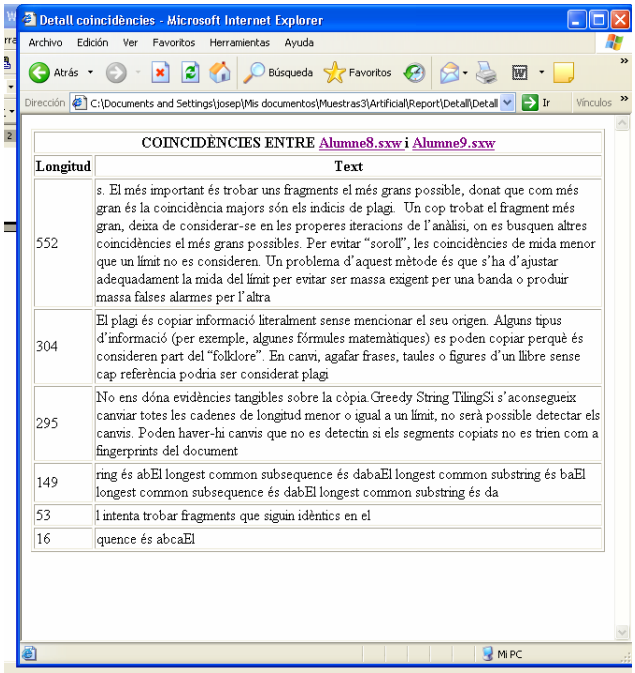
MML: 10

Percentatge Mínim: 0%

Llista de coincidències					
Fitxer A	**Fitxer B**	**% Contingut**	**MaxMatch**	**Metadades**	**Mostrar Detall**
Alumne1.sxw	Alumne2.sxw	1%	18	Res Significatiu	Detall
Alumne1.sxw	Alumne3.sxw	3%	25	Res Significatiu	Detall
Alumne1.sxw	Alumne4.sxw	55%	158	Res Significatiu	Detall
Alumne1.sxw	Alumne5.sxw	1%	18	Res Significatiu	Detall
Alumne1.sxw	Alumne6.sxw	35%	84	Res Significatiu	Detall
Alumne1.sxw	Alumne7.sxw	14%	49	Res Significatiu	Detall
Alumne1.sxw	Alumne8.sxw	1%	18	Res Significatiu	Detall
Alumne1.sxw	Alumne9.sxw	1%	18	Res Significatiu	Detall
Alumne2.sxw	Alumne3.sxw	8%	111	Res Significatiu	Detall
Alumne2.sxw	Alumne4.sxw	1%	19	Res Significatiu	Detall
Alumne2.sxw	Alumne5.sxw	78%	205	Res Significatiu	Detall
Alumne2.sxw	Alumne6.sxw	0%	12	Res Significatiu	Detall
Alumne2.sxw	Alumne7.sxw	3%	35	Res Significatiu	Detall
Alumne2.sxw	Alumne8.sxw	85%	914	Res Significatiu	Detall
Alumne2.sxw	Alumne9.sxw	81%	552	Res Significatiu	Detall

Mi PC

Figura 5.4.3 Document principal de l'informe de resultats



Detall coincidències - Microsoft Internet Explorer

Archivo Edición Ver Favoritos Herramientas Ayuda

Atrás Búsqueda Favoritos

Dirección C:\Documents and Settings\josep\Mis documentos\Muestras3\Artificial\Report\Detall\Detall Ir Vínculos

COINCIDÈNCIES ENTRE Alumne8.sxw i Alumne9.sxw	
Longitud	**Text**
552	s. El més important és trobar uns fragments el més grans possible, donat que com més gran és la coincidència majors són els indicis de plagi. Un cop trobat el fragment més gran, deixa de considerar-se en les properes iteracions de l'anàlisi, on es busquen altres coincidències el més grans possibles. Per evitar "soroll", les coincidències de mida menor que un límit no es consideren. Un problema d'aquest mètode és que s'ha d'ajustar adequadament la mida del límit per evitar ser massa exigent per una banda o produir massa falses alarmes per l'altra
304	El plagi és copiar informació literalment sense mencionar el seu origen. Alguns tipus d'informació (per exemple, algunes fórmules matemàtiques) es poden copiar perquè és consideren part del "folklore". En canvi, agafar frases, taules o figures d'un llibre sense cap referència podria ser considerat plagi
295	No ens dóna evidències tangibles sobre la còpia.Greedy String TilingSi s'aconsegueix canviar totes les cadenes de longitud menor o igual a un límit, no serà possible detectar els canvis. Poden haver-hi canvis que no es detectin si els segments copiats no es trien com a fingerprints del document
149	ring és abEl longest common subsequence és dabaEl longest common substring és baEl longest common subsequence és dabEl longest common substring és da
53	l intenta trobar fragments que siguin idèntics en el
16	quence és abcaEl

Mi PC

Figura 5.4.4 Detall de la comparació

5.5 JOC DE PROVES

El joc de proves utilitzat per posar a prova l'efectivitat de l'aplicació desenvolupada ha constat de quatre parts diferenciades. D'una banda, s'ha fet servir documents de proves reals proporcionats pel consultor del PFC. D'altra, documents amb fragments plagiats també proporcionats pel consultor del PFC. També s'ha fet servir un joc de proves amb documents molt extensos per posar a prova l'eficiència i el temps d'execució de l'aplicació. Per acabar, s'ha fet servir un últim joc de proves petit però preparat pel mateix desenvolupador per tal de provar aspectes molt específics de l'aplicació i afinar-la el màxim possible.

Amb aquest PFC es lliura a més un dels jocs de proves utilitzats amb resultats de plagi tant positius com negatius perquè l'avaluador el tingui al seu abast i així poder provar l'aplicació desenvolupada.

Al quadre 5.5.1 es pot veure un resum de temps d'execució de l'aplicació amb un *MML* de 20 i un percentatge mínim de 10. S'ha tingut en compte el temps que es tardava en una primera etapa de desenvolupament, quan l'algoritme emprat era simplement el *Greedy String-Tiling*, i posteriorment quan es va implementar la millora *Running Karp-Rabin*. Es pot veure que l'algoritme *Running Karp-Rabin* suposa una millora important sobre el temps d'execució.

	GST	RKR-GST
Joc de proves adjunt (9 fitxers + enunciat)	4 segons	2 segons
Joc de proves adjunt copiat 10 vegades (99 fitxers + enunciat)	244 segons	132 segons

Taula 5.5.1 Comparació temps d'execució

6. LÍNIES FUTURES DE TREBALL

Com és normal, en desenvolupar un projecte com aquest es plantegen moltes millores que es poden aplicar però que no dóna temps de desenvolupar en un únic semestre. En aquest apartat es recull una llista de les millores més significatives que es podrien tenir en compte per tal de facilitar la tasca del consultor o bé millorar l'eficiència i rapidesa de l'aplicació.

- Calcular prèviament la memòria disponible i la que faria falta per guardar a memòria els *hash* de tots els documents implicats en la comparació i, si és viable, fer-ho d'aquesta manera. Això comporta que només s'hauria de processar un cop cada fitxer per computar els seus *hash*, ja que ara alguns dels fitxers s'han de processar més d'un cop

- Pantalla de configuració de l'usuari amb la opció de guardar. Permetre obrir informe automàticament en acabar de processar, *MML* per defecte, % mínim per defecte, etc.

- En el preprocessament, eliminar també preposicions, articles, etc.. En definitiva, qualsevol paraula que no aporta res significatiu al contingut textual del document. En aquest cas, però, s'ha de tenir en compte que els documents poden estar en castellà o en català i aquestes paraules no són iguals en tots dos idiomes

- Poder ordenar d'alguna manera els resultats a l'informe, de manera que l'usuari pugui triar com els vol visualitzar

- Fer un altre procés paral·lel en què es pugui comptar el número d'aparicions de paraules en cada frase en el mateix ordre. És una mètrica que pot ser interessant de cara a detectar *paraphrasing* o substitució de paraules per sinònims

- En alguns casos, les proves d'avaluació continuada tenen un domini molt concret. Així, es pot donar a l'usuari la possibilitat de crear un diccionari amb les paraules més usuals d'aquest domini i que aquestes no es tinguin en compte a l'hora d'efectuar la comparació, és a dir, que s'eliminin dels documents mitjançant el preprocessament

- Implementar l'execució de l'aplicació mitjançant un *thread*. D'aquesta manera es pot mostrar missatges informatius per pantalla en temps real, pausar l'execució del programa, etc.

CONCLUSIONS

En el desenvolupament d'aquest PFC s'ha estudiat a fons el format de document *OpenOffice.org* i com emmagatzema el seu contingut en format *XML*, assimilant tota la informació necessària per la seva elaboració.

Aquests documents són molt fàcils de manipular degut precisament a la seva forma d'emmagatzematge i a que *OpenOffice.org* és un projecte *open-source* i, per tant, existeix molta informació a *Internet* sobre el seu funcionament intern.

L'aplicació desenvolupada executa correctament tots els jocs de proves als que ha estat sotmesa, encara que com s'explica a l'apartat *"Línies Futures de Treball"* s'hi podrien implementar moltes millores per perfeccionar la seva eficiència i utilitat.

Posteriorment, la millora efectuada a l'algoritme emprat, *RKR-GST*, ha demostrat la seva eficiència tal i com es mostra al quadre resum de l'apartat 5.5.

En definitiva, s'han assolit els objectius inicialment establerts, que es van recollir al Capítol 0.

GLOSSARI

Col·lisió: Veure Collusion

Collusion: Forma de plagi que es pot donar de les següents maneres: 1) Una part o la totalitat d'un treball ha estat realitzat per una altra persona diferent al suposat autor 2) L'autor copia una part o la totalitat del treball d'una altra persona amb el seu consentiment 3) Algú dóna consentiment a un altre per copiar una part o la totalitat d'un treball sabent que el presentarà com a propi 4) Dos o més autors treballen conjuntament per desenvolupar un treball però s'atribueixen el treball com a propi i individual

Content.xml: Un dels documents XML contingut a un document principal *OpenOffice.org* que bàsicament correspon al text

---

Diff: Eina dels sistemes operatius *UNIX* que reporta els canvis necessaris per convertir un fitxer A en un altre fitxer B

Distància de Levenshtein: Veure Levenshtein, Distància de

Distància d'edició: Veure Levenshtein, Distància de

Flesch Reading Ease: Mètrica basada en una escala de 0 a 100 que indica en quin grau un document textual és intel·ligible. Mentre més gran és la puntuació, més fàcil és entendre el document

---

Glatt Plagiarism Teaching Program: Tutorial que mostra com diferenciar què és plagi del que no ho és i ajuda a evitar-ho

Greedy String Tiling: Algoritme per detecció de còpia textual en parell de documents, mitjançant subcadenes dels mateixos anomenades *tiles*

Gunning Fog: Índex que indica el nombre d'anys d'educació escolar que suposadament necessita una persona per entendre un determinat document

---

Hash: En informàtica, representa una *clau* que identifica unívocament un document, registre, arxiu, etc.. També anomenat *Funció Hash* o *Funció Resum*. Aquesta *clau* permet un accés molt més ràpid al contingut que identifica

---

HTML: Acrònim d'Hypertext Markup Language, llenguatge de marcatge hipertextual. Metallenguatge dissenyat per estructurar texts i presentar-los en format d'hipertext, estàndar de les pàgines *Web*

JPLAG: Eina de detecció de plagi utilitzada principalment per la detecció en codi font de programes, encara que també suporta el llenguatge natural

Levenshtein, Distància de: Mesura de similitud entre dos cadenes de text. Indica la quantitat d'esborrats, insercions o substitucions que s'han de dur a terme per convertir una cadena de text *A* en una altra cadena de text *B*

Llenguatge natural: Llenguatge utilitzat pels humans per comunicar-se entre sí, en contraposició a llenguatges artificials creats amb finalitats específiques com per exemple els llenguatges de programació

Match: En aquest context, coincidència d'una cadena de text d'una determinada longitud entre dos documents

Metadada: Dada que descriu un determinat conjunt de dades. En general, les *metadades* intenten respondre preguntes com qui, què, quan, com, on i perquè sobre les dades que descriuen

Meta.xml: Un dels documents *XML* contingut a un document principal *OpenOffice.org* que bàsicament correspon a les *metadades* del document

Minimum Match Length: Veure MML

MML: En l'algoritme *Greedy String-Tiling*, *l*ongitud mínima de coincidència de cadenes entre dos documents per tal que siguin considerades com a plagi

MOSS: Acrònim de *Measure Of Software Similarity*. Eina orientada a la cerca de plagi en fitxers de codi font de programes

---

NLP: Acrònim de *Natural Language Processing*. Subcamp de la intel·ligència artificial i la lingüística que estudia el procés i la manipulació del llenguatge natural orientats principalment al seu enteniment per part de les computadores

---

OMPI: Acrònim d'*Organización Mundial de la Propiedad Intelectual*. Agència especialitzada de les Nacions Unides creada al 1967 que promou l'activitat creativa, protegint la propietat intel·lectual arreu del món

OpenOffice.org: Conjunt de programari gratuït i de codi obert, disponible per diferents plataformes, que competeix directament amb el conjunt de programari *Microsoft Office* però que és totalment compatible amb aquest. Inclou un processador de texts, un full de càlcul, una utilitat per confeccionar presentacions, una utilitat de dibuix i un gestor de bases de dades

Open-source: Es refereix a programari per computadores i la disponibilitat pública del seu codi font sota una llicència especial, anomenada també *llicència open-source*. Aquest tipus de programari es crea per estudiar, canviar i millorar el seu disseny

---

Paraphrasing: Una de les formes de plagi. Consisteix en explicar quelcom amb unes altres paraules o d'una altra manera. Sovint és difícil de detectar

---

Parser: Programa informàtic que efectua un *parsing* sobre una seqüència d'entrada

Parsing: En informàtica, procés d'analitzar una seqüència d'entrada, per exemple des de teclat o des d'un fitxer, amb la finalitat de determinar l'estructura gramatical respecte una determinada gramàtica formal

Plagi: Acte de presentar la totalitat o una part d'un treball com a propi quan no ho és

Plagiarism.org: Eina *online* que permet preveure el plagi, evitar l'engany escolar en la presentació de treballs i incrementar l'ètica acadèmica

Preprocessament: En aquest context, procés d'una cadena de text per tal de deixar-la canonitzada segons unes determinades regles per facilitar la posterior tasca de detecció de plagi

Running Karp Rabin: Optimització de l'algoritme *Greedy String Tiling* que millora la seva rapidesa i efectivitat

SGAE: Acrònim de *Sociedad General de Autores y Editores*. Entitat espanyola de gestió col·lectiva dedicada a la defensa dels drets d'autor dels seus socis, entre els quals es troben tota mena d'artistes i empresaris del negoci de la cultura

SMOG: Índex que determina la facilitat de lectura d'un document textual

Token: En aquest context, part mínima d'un document textual que és susceptible de ser comparada amb un altre document

WINNOWING: Algoritme utilitzat per la detecció de plagi, similar al *Running Karp Rabin*, que també utilitza funcions de *hash* amb aquesta finalitat

XML: Acrònim d'*eXtensible Markup Language*, llenguatge de marcatge extensible. Un dels seus objectius principals és substituir el llenguatge *HTML* ja que hi aporta semàntica fent una estructura molt comprensible pels éssers humans, encara que té altres aplicacions. Un dels usos més importants que se li dóna és la seva utilització com a estàndard per l'intercanvi de dades entre diversos programaris

YAP: Acrònim de *Yet Another Plague*. Eina informàtica per la detecció de plagi en fitxers de codi font de programari

BIBLIOGRAFIA

[1] Louis Bloomfield: *"The Importance of Writing"*. Referència *on-line*, 12 d'abril de 2004.
http://plagiarism.phys.virginia.edu/essays/The%20Importance%20of%20Writing.html

[2] Paul Clough: *"Plagiarism in Natural and Programming Languages: An Overview of Current Tools and Technologies"*. Document electrònic, juny de 2000.
http://ir.shef.ac.uk/cloughie/papers/plagiarism2000.pdf

[3] Writing Tutorial Services, Indiana University: *"Plagiarism: What it is and how to Recognize and Avoid it"*. Referència *on-line*, 27 d'abril de 2004.
http://www.indiana.edu/~wts/pamphlets/plagiarism.shtml

[4] Organización Mundial de la Propiedad Intelectual: *"El derecho de autor y los derechos conexos"*. Referència *on-line*, octubre de 2005.
http://www.ompi.int/about-ip/es/copyright.html

[5] *Microsoft Corporation*: "*Natural Language Processing"*. Referència *on-line*, octubre de 2005.
http://research.microsoft.com/nlp/

[7] Michael J. Wise, Dept. of CS, University of Sidney: *"Detection of Similarities via Greedy String Tiling and Running Karp-Rabin matching"*. Document electrònic, desembre de 1993.
ftp://ftp.cs.su.oz.au/michaelw/doc/RKR_GST.ps

[8] Lutz Prechelt, Guido Malpohl, Michael Phlippsen: *"JPlag: Finding plagiarisms among a set of programs"*. Document electrònic, 28 de març de 2000.
http://page.mi.fu-berlin.de/~prechelt/Biblio/jplagTR.pdf

[9] R. M. Karp, M. O. Rabin, IBM J. of Research and Development: *"Efficient Randomized Pattern-Matching Algorithms"*. Document electrònic, març de 1987.
http://www.research.ibm.com/journal/rd/312/ibmrd3102P.pdf

[10] Alex Aiken: *"MOSS: A System for Detecting Software Plagiarism"*. Referència *on-line*, 19 d'octubre de 2005.
http://www.cs.berkeley.edu/~aiken/moss.html

[11] Kevin W. Bowyer, Lawrence O. Hall: *"Experience Using 'MOSS' to Detect Cheating On Programming Assignments"*. Document electrònic, sense data.
http://www.cse.nd.edu/~kwb/nsf-ufe/1110.pdf

[12] Saul Schleimer, Daniel S. Wilkerson, Alex Aiken: *"Winnowing: Local Algorithms for Document Fingerprinting"*. Document electrònic, sense data.
http://theory.stanford.edu/~aiken/publications/papers/sigmod03.pdf

[13] Michael J. Wise, Department of Computer Science, University of Sidney, Australia: *"YAP3: Improved Detection of Similarities in Computer Program and Other Texts"*. Document electrònic, març de 1996.
http://www.bio.cam.ac.uk/~mw263/ftp/doc/yap3.ps

[14] Michael Gilleland: *"Levenshtein Distance, in Three Flavors*". Referència *on-line*, juny de 2003.
http://www.merriampark.com/ld.htm

[15] *OpenOffice.org* : "*OpenOffice.org XML* file format". Referència *on-line*, sense data.
http://xml.OpenOffice.org/