

XenC

1.2.0

Generated by Doxygen 1.8.3.1

Mon Aug 19 2013 12:02:46

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Namespace Documentation	9
5.1	XenCommon Namespace Reference	9
5.1.1	Detailed Description	10
5.1.2	Function Documentation	10
5.1.2.1	flip_map	10
5.1.2.2	flip_pair	10
5.1.2.3	getStdoutFromCommand	10
5.1.2.4	toDouble	11
5.1.2.5	toInt	11
5.1.2.6	toString	11
5.1.2.7	toString0	12
5.1.2.8	wordCount	13
6	Class Documentation	15
6.1	_Options Struct Reference	15
6.1.1	Detailed Description	17
6.1.2	Member Data Documentation	17
6.1.2.1	binLM	17
6.1.2.2	bp	17
6.1.2.3	dev	17
6.1.2.4	discount	17

6.1.2.5	eval	17
6.1.2.6	fullVoc	17
6.1.2.7	inSData	18
6.1.2.8	inSLM	18
6.1.2.9	inSStem	18
6.1.2.10	inTData	18
6.1.2.11	inTLM	18
6.1.2.12	inToks	18
6.1.2.13	inTStem	18
6.1.2.14	inv	18
6.1.2.15	iPTable	18
6.1.2.16	local	18
6.1.2.17	log	18
6.1.2.18	mean	18
6.1.2.19	mode	19
6.1.2.20	mono	19
6.1.2.21	name	19
6.1.2.22	noUnklsWord	19
6.1.2.23	oPTable	19
6.1.2.24	order	19
6.1.2.25	outName	19
6.1.2.26	outSData	19
6.1.2.27	outSLM	19
6.1.2.28	outSStem	19
6.1.2.29	outTData	19
6.1.2.30	outTLM	19
6.1.2.31	outToks	20
6.1.2.32	outTStem	20
6.1.2.33	pc	20
6.1.2.34	rev	20
6.1.2.35	sampleSize	20
6.1.2.36	sim	20
6.1.2.37	simOnly	20
6.1.2.38	sLang	20
6.1.2.39	sortOnly	20
6.1.2.40	stem	20
6.1.2.41	step	20
6.1.2.42	sVocab	20
6.1.2.43	threads	21
6.1.2.44	tLang	21

6.1.2.45	toLower	21
6.1.2.46	tVocab	21
6.1.2.47	vecSize	21
6.1.2.48	version	21
6.1.2.49	wFile	21
6.2	BiXEntropy Class Reference	21
6.2.1	Detailed Description	22
6.2.2	Constructor & Destructor Documentation	22
6.2.2.1	BiXEntropy	22
6.2.2.2	~BiXEntropy	22
6.2.3	Member Function Documentation	22
6.2.3.1	launch	22
6.3	Corpus Class Reference	23
6.3.1	Detailed Description	23
6.3.2	Constructor & Destructor Documentation	23
6.3.2.1	Corpus	23
6.3.2.2	~Corpus	24
6.3.3	Member Function Documentation	24
6.3.3.1	getLang	24
6.3.3.2	getLine	24
6.3.3.3	getPrint	24
6.3.3.4	getSize	24
6.3.3.5	getWC	24
6.3.3.6	getXenFile	25
6.3.3.7	initialize	25
6.3.3.8	initialize	25
6.3.3.9	removeLine	26
6.4	CorpusPair Class Reference	26
6.4.1	Detailed Description	26
6.4.2	Constructor & Destructor Documentation	26
6.4.2.1	CorpusPair	26
6.4.2.2	~CorpusPair	26
6.4.3	Member Function Documentation	26
6.4.3.1	getPtrInCorp	26
6.4.3.2	getPtrOutCorp	27
6.5	Eval Class Reference	27
6.5.1	Detailed Description	27
6.5.2	Constructor & Destructor Documentation	27
6.5.2.1	Eval	27
6.5.2.2	Eval	27

6.5.2.3	<code>~Eval</code>	28
6.5.3	Member Function Documentation	28
6.5.3.1	<code>doBP</code>	28
6.5.3.2	<code>doEval</code>	29
6.5.3.3	<code>getDist</code>	29
6.6	LMPair Class Reference	30
6.6.1	Detailed Description	30
6.6.2	Constructor & Destructor Documentation	30
6.6.2.1	<code>LMPair</code>	30
6.6.2.2	<code>~LMPair</code>	30
6.6.3	Member Function Documentation	30
6.6.3.1	<code>getPtrInLM</code>	30
6.6.3.2	<code>getPtrOutLM</code>	31
6.7	MeanLMPair Class Reference	31
6.7.1	Detailed Description	31
6.7.2	Constructor & Destructor Documentation	31
6.7.2.1	<code>MeanLMPair</code>	31
6.7.2.2	<code>~MeanLMPair</code>	31
6.7.3	Member Function Documentation	31
6.7.3.1	<code>getPtrOutLM2</code>	31
6.7.3.2	<code>getPtrOutLM3</code>	32
6.8	MeanPPLPair Class Reference	32
6.8.1	Detailed Description	32
6.8.2	Constructor & Destructor Documentation	32
6.8.2.1	<code>MeanPPLPair</code>	32
6.8.2.2	<code>~MeanPPLPair</code>	32
6.8.3	Member Function Documentation	32
6.8.3.1	<code>getPtrOutPPL2</code>	32
6.8.3.2	<code>getPtrOutPPL3</code>	33
6.9	Mode Class Reference	33
6.9.1	Detailed Description	33
6.9.2	Constructor & Destructor Documentation	34
6.9.2.1	<code>~Mode</code>	34
6.9.3	Member Function Documentation	34
6.9.3.1	<code>extractSample</code>	34
6.9.3.2	<code>findSampleSize</code>	35
6.9.3.3	<code>launch</code>	35
6.10	MonoXEntropy Class Reference	36
6.10.1	Detailed Description	36
6.10.2	Constructor & Destructor Documentation	37

6.10.2.1	MonoXEntropy	37
6.10.2.2	~MonoXEntropy	37
6.10.3	Member Function Documentation	37
6.10.3.1	launch	37
6.11	PhraseTable Class Reference	39
6.11.1	Detailed Description	39
6.11.2	Constructor & Destructor Documentation	39
6.11.2.1	PhraseTable	39
6.11.2.2	~PhraseTable	39
6.11.3	Member Function Documentation	40
6.11.3.1	getAlignment	40
6.11.3.2	getCounts	40
6.11.3.3	getScores	40
6.11.3.4	getSize	41
6.11.3.5	getSource	41
6.11.3.6	getSrcPhrases	42
6.11.3.7	getTarget	42
6.11.3.8	getXenFile	42
6.11.3.9	initialize	42
6.11.3.10	setSrcPhrases	43
6.12	PhraseTablePair Class Reference	43
6.12.1	Detailed Description	44
6.12.2	Constructor & Destructor Documentation	44
6.12.2.1	PhraseTablePair	44
6.12.2.2	~PhraseTablePair	44
6.12.3	Member Function Documentation	44
6.12.3.1	getPtrInPT	44
6.12.3.2	getPtrOutPT	44
6.13	PPL Class Reference	44
6.13.1	Detailed Description	45
6.13.2	Constructor & Destructor Documentation	45
6.13.2.1	PPL	45
6.13.2.2	~PPL	45
6.13.3	Member Function Documentation	45
6.13.3.1	calcPPLCorpus	45
6.13.3.2	calcPPLPhraseTable	46
6.13.3.3	getCorpPPL	46
6.13.3.4	getPPL	46
6.13.3.5	getSize	47
6.13.3.6	getXE	47

6.13.3.7	initialize	47
6.13.3.8	initialize	47
6.14	PPLPair Class Reference	48
6.14.1	Detailed Description	48
6.14.2	Constructor & Destructor Documentation	48
6.14.2.1	PPLPair	48
6.14.2.2	~PPLPair	48
6.14.3	Member Function Documentation	48
6.14.3.1	getPtrInPPL	48
6.14.3.2	getPtrOutPPL	48
6.15	PTScoring Class Reference	49
6.15.1	Detailed Description	49
6.15.2	Constructor & Destructor Documentation	50
6.15.2.1	PTScoring	50
6.15.2.2	~PTScoring	50
6.15.3	Member Function Documentation	50
6.15.3.1	launch	50
6.16	Score Class Reference	52
6.16.1	Detailed Description	52
6.16.2	Constructor & Destructor Documentation	52
6.16.2.1	Score	52
6.16.2.2	~Score	52
6.16.3	Member Function Documentation	52
6.16.3.1	addScore	52
6.16.3.2	calibrate	53
6.16.3.3	getPrint	53
6.16.3.4	getScore	53
6.16.3.5	getSize	53
6.16.3.6	inverse	54
6.16.3.7	removeScore	54
6.17	ScoreHolder Class Reference	54
6.17.1	Detailed Description	54
6.17.2	Constructor & Destructor Documentation	54
6.17.2.1	ScoreHolder	54
6.17.2.2	~ScoreHolder	55
6.17.3	Member Function Documentation	55
6.17.3.1	getPtrScores	55
6.17.3.2	getPtrScSimil	55
6.17.3.3	getPtrScXenC	55
6.18	Similarity Class Reference	55

6.18.1 Detailed Description	56
6.18.2 Constructor & Destructor Documentation	56
6.18.2.1 Similarity	56
6.18.2.2 ~Similarity	56
6.18.3 Member Function Documentation	56
6.18.3.1 getSim	56
6.18.3.2 getSize	56
6.18.3.3 initialize	56
6.19 SimplePPL Class Reference	57
6.19.1 Detailed Description	57
6.19.2 Constructor & Destructor Documentation	58
6.19.2.1 SimplePPL	58
6.19.2.2 ~SimplePPL	58
6.19.3 Member Function Documentation	58
6.19.3.1 launch	58
6.20 SourcePhrase Class Reference	60
6.20.1 Detailed Description	60
6.20.2 Constructor & Destructor Documentation	60
6.20.2.1 SourcePhrase	60
6.20.2.2 ~SourcePhrase	60
6.20.3 Member Function Documentation	60
6.20.3.1 addAlignments	61
6.20.3.2 addCounts	61
6.20.3.3 addScores	61
6.20.3.4 addTarget	61
6.20.3.5 getScoresXE	61
6.20.3.6 getSource	62
6.20.3.7 getTargetSize	62
6.21 XenCommon::Splitter Class Reference	62
6.21.1 Detailed Description	63
6.21.2 Member Typedef Documentation	63
6.21.2.1 size_type	63
6.21.3 Constructor & Destructor Documentation	63
6.21.3.1 Splitter	63
6.21.3.2 Splitter	63
6.21.4 Member Function Documentation	63
6.21.4.1 operator[]	63
6.21.4.2 reset	63
6.21.4.3 size	63
6.22 StaticData Class Reference	64

6.22.1	Detailed Description	65
6.22.2	Member Function Documentation	65
6.22.2.1	deleteInstance	65
6.22.2.2	getDevCorp	65
6.22.2.3	getInstance	66
6.22.2.4	getMeanSourceLMs	66
6.22.2.5	getMeanSourcePPLs	67
6.22.2.6	getMeanTargetLMs	67
6.22.2.7	getMeanTargetPPLs	68
6.22.2.8	getPTPairs	68
6.22.2.9	getScHold	69
6.22.2.10	getSim	69
6.22.2.11	getSourceCorps	69
6.22.2.12	getSourceLMs	70
6.22.2.13	getSourcePPLs	70
6.22.2.14	getStemSourceCorps	71
6.22.2.15	getStemSourceLMs	71
6.22.2.16	getStemSourcePPLs	72
6.22.2.17	getStemTargetCorps	72
6.22.2.18	getStemTargetLMs	73
6.22.2.19	getStemTargetPPLs	73
6.22.2.20	getStemVocabs	74
6.22.2.21	getTargetCorps	74
6.22.2.22	getTargetLMs	74
6.22.2.23	getTargetPPLs	75
6.22.2.24	getVocabs	75
6.22.2.25	getWeightsFile	76
6.22.2.26	getXenResult	76
6.23	VocabPair Class Reference	77
6.23.1	Detailed Description	77
6.23.2	Constructor & Destructor Documentation	77
6.23.2.1	VocabPair	77
6.23.2.2	~VocabPair	77
6.23.3	Member Function Documentation	78
6.23.3.1	getPtrSourceVoc	78
6.23.3.2	getPtrTargetVoc	78
6.24	Wfile Class Reference	78
6.24.1	Detailed Description	78
6.24.2	Constructor & Destructor Documentation	78
6.24.2.1	Wfile	78

6.24.2.2	~Wfile	79
6.24.3	Member Function Documentation	79
6.24.3.1	getSize	79
6.24.3.2	getWeight	79
6.24.3.3	initialize	79
6.25	XenCommon::XenCEption Struct Reference	79
6.25.1	Detailed Description	80
6.25.2	Constructor & Destructor Documentation	81
6.25.2.1	XenCEption	81
6.25.2.2	~XenCEption	81
6.25.3	Member Function Documentation	81
6.25.3.1	what	81
6.25.4	Member Data Documentation	81
6.25.4.1	s	81
6.26	XenFile Class Reference	81
6.26.1	Detailed Description	82
6.26.2	Constructor & Destructor Documentation	82
6.26.2.1	XenFile	82
6.26.2.2	~XenFile	82
6.26.3	Member Function Documentation	82
6.26.3.1	getDirName	82
6.26.3.2	getExt	82
6.26.3.3	getFileName	83
6.26.3.4	getFullPath	83
6.26.3.5	getPrefix	83
6.26.3.6	initialize	83
6.26.3.7	isGZ	83
6.27	XenIO Class Reference	84
6.27.1	Detailed Description	85
6.27.2	Member Function Documentation	85
6.27.2.1	cleanCorpusBi	85
6.27.2.2	cleanCorpusMono	85
6.27.2.3	dumpSimilarity	86
6.27.2.4	read	86
6.27.2.5	readDist	87
6.27.2.6	writeBiOutput	88
6.27.2.7	writeEval	89
6.27.2.8	writeMonoOutput	90
6.27.2.9	writeNewPT	90
6.27.2.10	writeSourcePhrases	91

6.27.2.11 writeTargetPhrases	92
6.28 XenLMsri Class Reference	92
6.28.1 Detailed Description	93
6.28.2 Constructor & Destructor Documentation	93
6.28.2.1 XenLMsri	93
6.28.2.2 ~XenLMsri	94
6.28.3 Member Function Documentation	94
6.28.3.1 createLM	94
6.28.3.2 getDocumentStats	94
6.28.3.3 getFileName	95
6.28.3.4 getSentenceStats	95
6.28.3.5 initialize	95
6.28.3.6 initialize	96
6.28.3.7 initialize	96
6.28.3.8 loadLM	97
6.28.3.9 writeLM	97
6.29 XenOption Class Reference	98
6.29.1 Detailed Description	100
6.29.2 Member Function Documentation	100
6.29.2.1 deleteInstance	100
6.29.2.2 getBinLM	100
6.29.2.3 getBp	101
6.29.2.4 getDev	101
6.29.2.5 getDiscount	102
6.29.2.6 getEval	102
6.29.2.7 getFullVocab	102
6.29.2.8 getInPTable	103
6.29.2.9 getInSData	103
6.29.2.10 getInSLM	104
6.29.2.11 getInSStem	104
6.29.2.12 getInstance	105
6.29.2.13 getInstance	106
6.29.2.14 getInTData	107
6.29.2.15 getInTLM	107
6.29.2.16 getInTStem	107
6.29.2.17 getInv	108
6.29.2.18 getLocal	108
6.29.2.19 getLog	109
6.29.2.20 getMean	109
6.29.2.21 getMode	110

6.29.2.22	getMono	110
6.29.2.23	getName	111
6.29.2.24	getOrder	111
6.29.2.25	getOutName	111
6.29.2.26	getOutPTable	112
6.29.2.27	getOutSData	112
6.29.2.28	getOutSLM	113
6.29.2.29	getOutSStem	113
6.29.2.30	getOutTData	114
6.29.2.31	getOutTLM	114
6.29.2.32	getOutTStem	115
6.29.2.33	getRev	115
6.29.2.34	getSampleSize	116
6.29.2.35	getSim	116
6.29.2.36	getSimOnly	117
6.29.2.37	getSLang	117
6.29.2.38	getSortOnly	118
6.29.2.39	getStem	118
6.29.2.40	getStep	119
6.29.2.41	getSVocab	119
6.29.2.42	getThreads	120
6.29.2.43	getTLang	120
6.29.2.44	getToLower	121
6.29.2.45	getTVocab	121
6.29.2.46	getVecSize	122
6.29.2.47	getWFile	122
6.29.2.48	noUnklsWord	123
6.29.2.49	setSampleSize	123
6.29.2.50	setStep	123
6.30	XenResult Class Reference	124
6.30.1	Detailed Description	124
6.30.2	Constructor & Destructor Documentation	124
6.30.2.1	XenResult	124
6.30.2.2	~XenResult	125
6.30.3	Member Function Documentation	125
6.30.3.1	getSize	125
6.30.3.2	getSortedText	125
6.30.3.3	getTextLine	125
6.30.3.4	getXenFile	125
6.30.3.5	initialize	125

6.31	XenVocab Class Reference	126
6.31.1	Detailed Description	126
6.31.2	Constructor & Destructor Documentation	127
6.31.2.1	XenVocab	127
6.31.2.2	~XenVocab	127
6.31.3	Member Function Documentation	127
6.31.3.1	getSize	127
6.31.3.2	getVocab	127
6.31.3.3	getXenFile	127
6.31.3.4	getXenVocab	127
6.31.3.5	initialize	127
6.31.3.6	initialize	128
6.31.3.7	initialize	128
6.31.3.8	initialize	128
7	File Documentation	129
7.1	include/corpus.h File Reference	129
7.1.1	Detailed Description	130
7.2	include/eval.h File Reference	130
7.2.1	Detailed Description	131
7.2.2	Typedef Documentation	131
7.2.2.1	EvalMap	131
7.2.3	Function Documentation	131
7.2.3.1	taskEval	131
7.3	include/mode.h File Reference	132
7.3.1	Detailed Description	133
7.4	include/modes/biXEntropy.h File Reference	133
7.4.1	Detailed Description	134
7.5	include/modes/monoXEntropy.h File Reference	134
7.5.1	Detailed Description	135
7.6	include/modes/ptScoring.h File Reference	136
7.6.1	Detailed Description	136
7.7	include/modes/simplePPL.h File Reference	137
7.7.1	Detailed Description	137
7.8	include/phrasetable.h File Reference	138
7.8.1	Detailed Description	139
7.9	include/ppl.h File Reference	139
7.9.1	Detailed Description	140
7.9.2	Function Documentation	140
7.9.2.1	taskCalcPPL	140

7.10	include/score.h File Reference	141
7.10.1	Detailed Description	142
7.11	include/similarity.h File Reference	142
7.11.1	Detailed Description	143
7.11.2	Typedef Documentation	144
7.11.2.1	SimMap	144
7.12	include/sourcephrase.h File Reference	144
7.12.1	Detailed Description	145
7.13	include/utils/common.h File Reference	145
7.13.1	Detailed Description	147
7.13.2	Typedef Documentation	147
7.13.2.1	LPOptions	147
7.13.2.2	Options	147
7.14	include/utils/StaticData.h File Reference	147
7.14.1	Detailed Description	148
7.15	include/utils/xenio.h File Reference	149
7.15.1	Detailed Description	149
7.16	include/wfile.h File Reference	150
7.16.1	Detailed Description	150
7.17	include/Xen.h File Reference	150
7.17.1	Detailed Description	151
7.17.2	Function Documentation	152
7.17.2.1	getOutName	152
7.17.2.2	main	154
7.17.2.3	sanityCheck	155
7.18	include/xenfile.h File Reference	157
7.18.1	Detailed Description	158
7.19	include/XenLMsri.h File Reference	158
7.19.1	Detailed Description	159
7.19.2	Macro Definition Documentation	159
7.19.2.1	MAX_CHARS	159
7.19.2.2	MAX_ORDER	160
7.19.2.3	MAX_WORDS	160
7.20	include/xenoption.h File Reference	160
7.20.1	Detailed Description	161
7.21	include/xenresult.h File Reference	161
7.21.1	Detailed Description	162
7.22	include/xenvocab.h File Reference	162
7.22.1	Detailed Description	163
7.23	src/corpus.cpp File Reference	164

7.23.1 Detailed Description	164
7.24 src/eval.cpp File Reference	164
7.24.1 Detailed Description	164
7.24.2 Function Documentation	165
7.24.2.1 taskEval	165
7.25 src/mode.cpp File Reference	165
7.25.1 Detailed Description	166
7.26 src/modes/biXEntropy.cpp File Reference	166
7.26.1 Detailed Description	166
7.27 src/modes/monoXEntropy.cpp File Reference	167
7.27.1 Detailed Description	167
7.28 src/modes/ptScoring.cpp File Reference	167
7.28.1 Detailed Description	168
7.29 src/modes/simplePPL.cpp File Reference	168
7.29.1 Detailed Description	168
7.30 src/phrasetable.cpp File Reference	169
7.30.1 Detailed Description	169
7.31 src/ppl.cpp File Reference	170
7.31.1 Detailed Description	170
7.31.2 Function Documentation	171
7.31.2.1 taskCalcPPL	171
7.32 src/score.cpp File Reference	171
7.32.1 Detailed Description	171
7.33 src/similarity.cpp File Reference	172
7.33.1 Detailed Description	172
7.34 src/sourcephrase.cpp File Reference	173
7.34.1 Detailed Description	173
7.35 src/utis/StaticData.cpp File Reference	173
7.35.1 Detailed Description	174
7.36 src/utis/xenio.cpp File Reference	174
7.36.1 Detailed Description	174
7.37 src/wfile.cpp File Reference	174
7.37.1 Detailed Description	175
7.38 src/Xen.cpp File Reference	175
7.38.1 Detailed Description	175
7.38.2 Function Documentation	176
7.38.2.1 getOutName	176
7.38.2.2 main	177
7.38.2.3 sanityCheck	178
7.38.3 Variable Documentation	180

7.38.3.1	version	180
7.39	src/xenfile.cpp File Reference	180
7.39.1	Detailed Description	180
7.40	src/XenLMsri.cpp File Reference	181
7.40.1	Detailed Description	181
7.40.2	Macro Definition Documentation	181
7.40.2.1	USE_STATS	181
7.40.2.2	USE_STATS	181
7.41	src/xenoption.cpp File Reference	181
7.41.1	Detailed Description	182
7.42	src/xenresult.cpp File Reference	182
7.42.1	Detailed Description	182
7.43	src/xenvocab.cpp File Reference	183
7.43.1	Detailed Description	183
Index		183

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

XenCommon	Namespace containing all the common functions of XenC	9
---------------------------	---	---

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

_Options	15
Corpus	23
CorpusPair	26
Eval	27
exception	
XenCommon::XenCEption	79
LMPair	30
MeanLMPair	31
MeanPPLPair	32
Mode	33
BiXEntropy	21
MonoXEntropy	36
PTScoring	49
SimplePPL	57
PhraseTable	39
PhraseTablePair	43
PPL	44
PPLPair	48
Score	52
ScoreHolder	54
Similarity	55
SourcePhrase	60
XenCommon::Splitter	62
StaticData	64
VocabPair	77
Wfile	78
XenFile	81
XenIO	84
XenLMsri	92
XenOption	98
XenResult	124
XenVocab	126

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

_Options	XenC options structure	15
BiXEntropy	Filtering mode 3: bilingual cross-entropy	21
Corpus	Corpus-related functionalities	23
CorpusPair	Tiny class holding two related Corpus	26
Eval	Evaluation system	27
LMPair	Tiny class holding two related language models	30
MeanLMPair	Tiny class holding two additional LMs for mean scoring feature	31
MeanPPLPair	Tiny class holding two additional PPL objects for mean scoring feature	32
Mode	Filtering modes interface	33
MonoXEntropy	Filtering mode 2: monolingual cross-entropy	36
PhraseTable	Class handling phrase-table related functionalities	39
PhraseTablePair	Tiny class holding the two phrase-tables	43
PPL	Perplexity/Cross-entropy computations	44
PPLPair	Tiny class holding two related PPL objects	48
PTScoring	Filtering mode 4: phrase-table cross-entropy	49
Score	Class holding the XenC scores representation	52
ScoreHolder	Tiny class holding three Score objects (global scores, similarity, cross-entropy)	54
Similarity	Class taking care of all the similarity measure computations	55
SimplePPL	Filtering mode 1: simple perplexity	57

SourcePhrase	
Class holding a merged source phrase and all associated data	60
XenCommon::Splitter	
Class defining a splitter	62
StaticData	
Class gathering all data used and generated by XenC	64
VocabPair	
Tiny class holding the two vocabularies	77
Wfile	
Class handling a file with values intended at weighting XenC scores	78
XenCommon::XenCEption	
XenC exception structure	79
XenFile	
Class providing some basic functions around files	81
XenIO	
Class handling all input/output operations of XenC	84
XenLMsri	
Class handling SRI LM estimation, loading, querying.. . . .	92
XenOption	
Singleton class handling XenC options accessors/mutators	98
XenResult	
Class handling a XenC sorted result file for evaluation/best point	124
XenVocab	
Class handling a XenC vocabulary	126

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

include/corpus.h	Class handling corpus-related functionalities	129
include/eval.h	Class handling evaluation system	130
include/mode.h	Abstract class defining the filtering modes architecture	132
include/phrasetable.h	Class handling phrase-table related functionalities	138
include/ppl.h	Class handling the perplexity/cross-entropy computations	139
include/score.h	Class holding the XenC scores representation	141
include/similarity.h	Class taking care of all the similarity measure computations	142
include/sourcephrase.h	Class holding a merged source phrase and all associated data	144
include/wfile.h	Class handling a file with values intended at weighting XenC scores	150
include/Xen.h	Main file of XenC, controls execution	150
include/xenfile.h	Class providing some basic functions around files	157
include/XenLMsri.h	Class handling SRI LM estimation, loading, querying.. . . .	158
include/xenoption.h	Singleton class handling XenC options accessors/mutators	160
include/xenresult.h	Class handling a XenC sorted result file for evaluation/best point	161
include/xenvocab.h	Class handling a XenC vocabulary	162
include/modes/biXEntropy.h	Derived class to handle filtering mode 3: bilingual cross-entropy	133
include/modes/monoXEntropy.h	Derived class to handle filtering mode 2: monolingual cross-entropy	134
include/modes/ptScoring.h	Derived class to handle filtering mode 4: phrase-table cross-entropy	136
include/modes/simplePPL.h	Derived class to handle filtering mode 1: simple perplexity	137

include/Utils/ common.h	File containing all common classes/structures/functions of many classes of XenC	145
include/Utils/ StaticData.h	File handling all data objects used by XenC in a static way	147
include/Utils/ xenio.h	Class handling all input/output operations of XenC	149
src/ corpus.cpp	Class handling corpus-related functionalities	164
src/ eval.cpp	Class handling evaluation system	164
src/ mode.cpp	Abstract class defining the filtering modes architecture	165
src/ phrasetable.cpp	Class handling phrase-table related functionalities	169
src/ ppl.cpp	Class handling the perplexity/cross-entropy computations	170
src/ score.cpp	Class holding the XenC scores representation	171
src/ similarity.cpp	Class taking care of all the similarity measure computations	172
src/ sourcephrase.cpp	Class holding a merged source phrase and all associated data	173
src/ wfile.cpp	Class handling a file with values intended at weighting XenC scores	174
src/ Xen.cpp	Main file of XenC, controls execution	175
src/ xenfile.cpp	Class providing some basic functions around files	180
src/ XenLMsri.cpp	Class handling SRI LM estimation, loading, querying..	181
src/ xenoption.cpp	Singleton class handling XenC options accessors/mutators	181
src/ xenresult.cpp	Class handling a XenC sorted result file for evaluation/best point	182
src/ xenvocab.cpp	Class handling a XenC vocabulary	183
src/modes/ biXEntropy.cpp	Derived class to handle filtering mode 3: bilingual cross-entropy	166
src/modes/ monoXEntropy.cpp	Derived class to handle filtering mode 2: monolingual cross-entropy	167
src/modes/ ptScoring.cpp	Derived class to handle filtering mode 4: phrase-table cross-entropy	167
src/modes/ simplePPL.cpp	Derived class to handle filtering mode 1: simple perplexity	168
src/Utils/ StaticData.cpp	File handling all data objects used by XenC in a static way	173
src/Utils/ xenio.cpp	Class handling all input/output operations of XenC	174

Chapter 5

Namespace Documentation

5.1 XenCommon Namespace Reference

Namespace containing all the common functions of XenC.

Classes

- struct [XenCEption](#)
XenC exception structure.
- class [Splitter](#)
Class defining a splitter.

Functions

- template<typename T >
std::string [toString](#) (const T &Value)
Template converting a value into a string with a precision of 20.
- template<typename T >
std::string [toString0](#) (const T &Value)
Template converting a value into a string with no precision.
- template<typename T >
int [toInt](#) (const T &Value)
Template converting a value (generally a string) into an integer.
- template<typename T >
double [toDouble](#) (const T &Value)
Template converting a value (generally a string) into a double.
- template<typename A , typename B >
std::pair< B, A > [flip_pair](#) (const std::pair< A, B > &p)
Template flipping a pair key type with value type.
- template<typename A , typename B >
std::multimap< B, A,
std::greater< B > > [flip_map](#) (const std::map< A, B > &src)
Template flipping a multimap with descending order keys with values.
- int [wordCount](#) (const std::string &str)
Computes the word count of a string.
- std::string [getStdoutFromCommand](#) (std::string cmd)
Executes a system command and returns the output.

5.1.1 Detailed Description

Namespace containing all the common functions of XenC.

5.1.2 Function Documentation

5.1.2.1 `template<typename A , typename B > std::multimap<B, A, std::greater > XenCommon::flip_map (const std::map< A, B > & src)`

Template flipping a multimap with descending order keys with values.

Template Parameters

<code>&src</code>	: the multimap to flip
-----------------------	------------------------

Returns

flipped multimap with descending order

5.1.2.2 `template<typename A , typename B > std::pair<B, A> XenCommon::flip_pair (const std::pair< A, B > & p)`

Template flipping a pair key type with value type.

Template Parameters

<code>&p</code>	: the map pair<A, B> to flip
---------------------	------------------------------

Returns

flipped pair<B, A>

5.1.2.3 `std::string XenCommon::getStdoutFromCommand (std::string cmd) [inline]`

Executes a system command and returns the output.

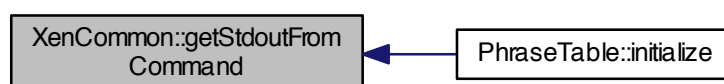
Parameters

<code>cmd</code>	: the command to execute
------------------	--------------------------

Returns

the output of the executed command

Here is the caller graph for this function:



5.1.2.4 `template<typename T > double XenCommon::toDouble (const T & Value)`

Template converting a value (generally a string) into an double.

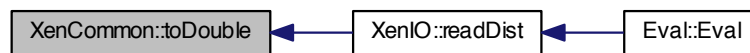
Template Parameters

<i>&Value</i>	: the value to convert
-------------------	------------------------

Returns

string containing the converted value

Here is the caller graph for this function:



5.1.2.5 `template<typename T > int XenCommon::toInt (const T & Value)`

Template converting a value (generally a string) into an integer.

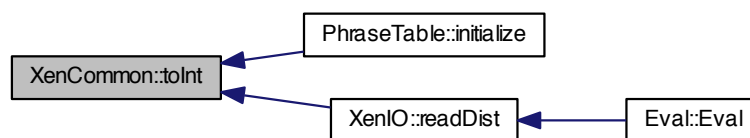
Template Parameters

<i>&Value</i>	: the value to convert
-------------------	------------------------

Returns

string containing the converted value

Here is the caller graph for this function:



5.1.2.6 `template<typename T > std::string XenCommon::toString (const T & Value)`

Template converting a value into a string with a precision of 20.

<i>&Value</i>	: the value to convert
-------------------	------------------------

string containing the converted value

```

graph TD
    main --> sanityCheck
    main --> getOutName
    main --> XenLMsri_createLM[XenLMsri::createLM]
    main --> XenIO_writeEval[XenIO::writeEval]
    main --> PTScoring_launch[PTScoring::launch]
    sanityCheck --> taskEval
    getOutName --> XenCommon_toString[XenCommon::toString]
    XenLMsri_createLM --> XenIO_writeNewPT[XenIO::writeNewPT]
    PTScoring_launch --> XenIO_writeMonoOutput[XenIO::writeMonoOutput]
    XenIO_writeEval --> XenIO_writeBIOutput[XenIO::writeBIOutput]
    XenIO_writeNewPT --> XenIO_writeMonoOutput
    XenIO_writeMonoOutput --> XenCommon_toString
    XenIO_writeBIOutput --> XenCommon_toString
    MonoXEntropy_launch[MonoXEntropy::launch] --> XenIO_writeBIOutput
    MonoXEntropy_launch --> XenIO_writeMonoOutput
    SimplePPL_launch[SimplePPL::launch] --> XenIO_writeMonoOutput
    BiXEntropy_launch[BiXEntropy::launch] --> Mode_extractSample[Mode::extractSample]
    BiXEntropy_launch --> XenIO_writeMonoOutput
    Mode_extractSample --> XenCommon_toString
    Eval_doEval[Eval::doEval] --> taskEval
    Eval_doBP[Eval::doBP] --> taskEval
    Eval_doBP --> XenCommon_toString
  
```

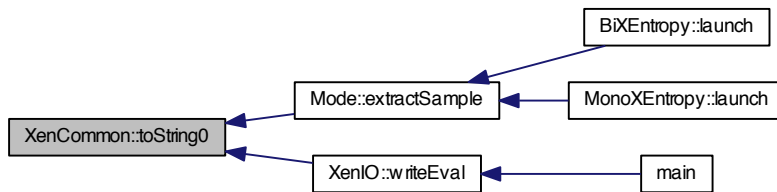
Template converting a value into a string with no precision.

<i>&Value</i>	: the value to convert
-------------------	------------------------

Returns

string containing the converted value

Here is the caller graph for this function:



5.1.2.8 int XenCommon::wordCount (const std::string & str) [inline]

Computes the word count of a string.

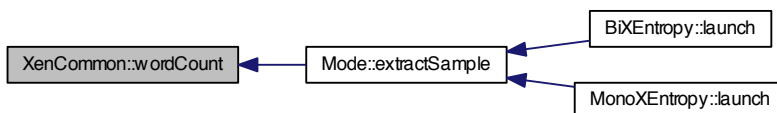
Parameters

<code>&str</code>	: the string to count the words
-----------------------	---------------------------------

Returns

the number of words of the string

Here is the caller graph for this function:



Chapter 6

Class Documentation

6.1 _Options Struct Reference

XenC options structure.

```
#include "utils/common.h"
```

Public Attributes

- std::string [sLang](#)
The source language.
- std::string [tLang](#)
The target language.
- std::string [inSData](#)
The in-domain source corpus.
- std::string [outSData](#)
The out-of-domain source corpus.
- std::string [inTData](#)
The in-domain target corpus.
- std::string [outTData](#)
The out-of-domain target corpus.
- std::string [inSStem](#)
The in-domain source stem corpus.
- std::string [outSStem](#)
The out-of-domain source stem corpus.
- std::string [inTStem](#)
The in-domain target stem corpus.
- std::string [outTStem](#)
The out-of-domain source stem corpus.
- std::string [iPTable](#)
The in-domain phrase-table.
- std::string [oPTable](#)
The out-of-domain phrase-table.
- int [mode](#)
The filtering mode.
- bool [mean](#)
Indicates mean computation.
- bool [sim](#)

- Indicates similarity computation.*
 - bool `simOnly`
Indicates similarity computation only.
 - int `vecSize`
The vector size for similarity.
 - std::string `sVocab`
The source language vocabulary.
 - std::string `tVocab`
The target language vocabulary.
 - bool `fullVoc`
Indicates if a global vocabulary is requested (instead of only in-domain)
 - std::string `inSLM`
The in-domain source language model.
 - std::string `outSLM`
The out-of-domain source language model.
 - std::string `inTLM`
The in-domain target language model.
 - std::string `outTLM`
The out-of-domain target language model.
 - std::string `wFile`
The weight file.
 - std::string `dev`
The development corpus (for evaluation)
 - int `order`
The order for language models estimation.
 - int `discount`
The discounting method for language models estimation.
 - bool `toLower`
Indicates lower case vocabulary mapping for LM estimation.
 - bool `noUnklsWord`
Indicates if <unk> is considered as a word or not.
 - int `binLM`
The language models output format (0 = ARPA, 1 = binary)
 - int `sampleSize`
The sample size for the out-of-domain corpus.
 - bool `log`
Indicates if the weights are given in the log domain.
 - bool `rev`
Indicates if a reversed output is requested (descending order)
 - bool `inv`
Indicates if an inverse output is requested (1 - score)
 - bool `mono`
Indicates if monolingual data is being filtered or not.
 - bool `stem`
Indicates stem computation.
 - bool `local`
Indicates local scores computation for phrase table filtering.
 - bool `eval`
Indicates evaluation mode.
 - bool `bp`
Indicates best-point evaluation mode.

- int `step`
The step size for evaluation and best-point.
- int `pc`
The current percentage being evaluated.
- int `inToks`
The number of in-domain tokens.
- int `outToks`
The number of out-of-domain tokens.
- std::string `outName`
The output file name.
- std::string `name`
The program name.
- bool `version`
The program version.
- int `threads`
The number of threads.
- bool `sortOnly`
Indicated outputting only the "sorted" file (not the "scored" one)

6.1.1 Detailed Description

XenC options structure.

6.1.2 Member Data Documentation

6.1.2.1 int _Options::binLM

The language models output format (0 = ARPA, 1 = binary)

6.1.2.2 bool _Options::bp

Indicates best-point evaluation mode.

6.1.2.3 std::string _Options::dev

The development corpus (for evaluation)

6.1.2.4 int _Options::discount

The discounting method for language models estimation.

6.1.2.5 bool _Options::eval

Indicates evaluation mode.

6.1.2.6 bool _Options::fullVoc

Indicates if a global vocabulary is requested (instead of only in-domain)

6.1.2.7 std::string _Options::inSData

The in-domain source corpus.

6.1.2.8 std::string _Options::inSLM

The in-domain source language model.

6.1.2.9 std::string _Options::inSStem

The in-domain source stem corpus.

6.1.2.10 std::string _Options::inTData

The in-domain target corpus.

6.1.2.11 std::string _Options::inTLM

The in-domain target language model.

6.1.2.12 int _Options::inToks

The number of in-domain tokens.

6.1.2.13 std::string _Options::inTStem

The in-domain target stem corpus.

6.1.2.14 bool _Options::inv

Indicates if an inverse output is requested (1 - score)

6.1.2.15 std::string _Options::iPTable

The in-domain phrase-table.

6.1.2.16 bool _Options::local

Indicates local scores computation for phrase table filtering.

6.1.2.17 bool _Options::log

Indicates if the weights are given in the log domain.

6.1.2.18 bool _Options::mean

Indicates mean computation.

6.1.2.19 int _Options::mode

The filtering mode.

6.1.2.20 bool _Options::mono

Indicates if monolingual data is being filtered or not.

6.1.2.21 std::string _Options::name

The program name.

6.1.2.22 bool _Options::noUnklsWord

Indicates if <unk> is considered as a word or not.

6.1.2.23 std::string _Options::oPTable

The out-of-domain phrase-table.

6.1.2.24 int _Options::order

The order for language models estimation.

6.1.2.25 std::string _Options::outName

The output file name.

6.1.2.26 std::string _Options::outSData

The out-of-domain source corpus.

6.1.2.27 std::string _Options::outSLM

The out-of-domain source language model.

6.1.2.28 std::string _Options::outSStem

The out-of-domain source stem corpus.

6.1.2.29 std::string _Options::outTData

The out-of-domain target corpus.

6.1.2.30 std::string _Options::outTLM

The out-of-domain target language model.

6.1.2.31 int _Options::outToks

The number of out-of-domain tokens.

6.1.2.32 std::string _Options::outTStem

The out-of-domain source stem corpus.

6.1.2.33 int _Options::pc

The current percentage being evaluated.

6.1.2.34 bool _Options::rev

Indicates if a reversed output is requested (descending order)

6.1.2.35 int _Options::sampleSize

The sample size for the out-of-domain corpus.

6.1.2.36 bool _Options::sim

Indicates similarity computation.

6.1.2.37 bool _Options::simOnly

Indicates similarity computation only.

6.1.2.38 std::string _Options::sLang

The source language.

6.1.2.39 bool _Options::sortOnly

Indicated outputting only the "sorted" file (not the "scored" one)

6.1.2.40 bool _Options::stem

Indicates stem computation.

6.1.2.41 int _Options::step

The step size for evaluation and best-point.

6.1.2.42 std::string _Options::sVocab

The source language vocabulary.

6.1.2.43 int _Options::threads

The number of threads.

6.1.2.44 std::string _Options::tLang

The target language.

6.1.2.45 bool _Options::toLower

Indicates lower case vocabulary mapping for LM estimation.

6.1.2.46 std::string _Options::tVocab

The target language vocabulary.

6.1.2.47 int _Options::vecSize

The vector size for similarity.

6.1.2.48 bool _Options::version

The program version.

6.1.2.49 std::string _Options::wFile

The weight file.

The documentation for this struct was generated from the following file:

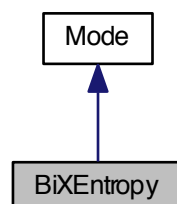
- include/utils/[common.h](#)

6.2 BiXEntropy Class Reference

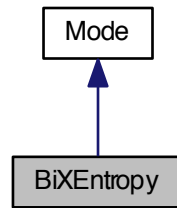
Filtering mode 3: bilingual cross-entropy.

```
#include <biXEntropy.h>
```

Inheritance diagram for BiXEntropy:



Collaboration diagram for BiXEntropy:



Public Member Functions

- [BiXEntropy](#) ()
Default constructor.
- [~BiXEntropy](#) ()
Default destructor.
- `int` [launch](#) ()
Function in charge of launching the filtering mode.

Additional Inherited Members

6.2.1 Detailed Description

Filtering mode 3: bilingual cross-entropy.

This class derived from [Mode](#) handles the third filtering mode: bilingual cross-entropy

6.2.2 Constructor & Destructor Documentation

6.2.2.1 `BiXEntropy::BiXEntropy ()`

Default constructor.

6.2.2.2 `BiXEntropy::~~BiXEntropy ()`

Default destructor.

6.2.3 Member Function Documentation

6.2.3.1 `int BiXEntropy::launch ()` `[virtual]`

Function in charge of launching the filtering mode.

Returns

0 if the filtering succeeds

Implements [Mode](#).

The documentation for this class was generated from the following files:

- include/modes/[biXEntropy.h](#)
- src/modes/[biXEntropy.cpp](#)

6.3 Corpus Class Reference

Corpus-related functionalities.

```
#include <corpus.h>
```

Public Member Functions

- [Corpus](#) ()
Default constructor.
- void [initialize](#) (boost::shared_ptr< [XenFile](#) > ptrData, std::string lg)
Initialization function from an already instanciated [XenFile](#).
- void [initialize](#) (std::string filePath, std::string lg)
Initialization function from a string containing a valid path/file name.
- [~Corpus](#) ()
Default destructor.
- boost::shared_ptr< [XenFile](#) > [getXenFile](#) () const
Accessor to the [XenFile](#) associated to the [Corpus](#).
- std::string [getLine](#) (int line)
Accessor to the lines of text from the [Corpus](#).
- unsigned int [getSize](#) () const
Accessor to the size of the [Corpus](#).
- std::string [getLang](#) () const
Accessor to the language of the [Corpus](#).
- bool [getPrint](#) (int line)
Accessor to the printing status of a line.
- int [getWC](#) () const
Accessor to the number of tokens of the [Corpus](#).
- void [removeLine](#) (int line)
Put the printing status of a line to false.

6.3.1 Detailed Description

Corpus-related functionalities.

This class handles the corpus used in XenC, providing means to get lines of text, size, language, token counts...

6.3.2 Constructor & Destructor Documentation

6.3.2.1 [Corpus::Corpus](#) ()

Default constructor.

6.3.2.2 Corpus::~~Corpus ()

Default destructor.

6.3.3 Member Function Documentation

6.3.3.1 std::string Corpus::getLang () const

Accessor to the language of the [Corpus](#).

Returns

string containing the language

6.3.3.2 std::string Corpus::getLine (int *line*)

Accessor to the lines of text from the [Corpus](#).

Parameters

<i>line</i>	: integer representing the line number
-------------	--

Returns

string containing the text line

6.3.3.3 bool Corpus::getPrint (int *line*)

Accessor to the printing status of a line.

Parameters

<i>line</i>	: integer representing the line number
-------------	--

Returns

true if the line can be printed

6.3.3.4 unsigned int Corpus::getSize () const

Accessor to the size of the [Corpus](#).

Returns

unsigned int representing the size

6.3.3.5 int Corpus::getWC () const

Accessor to the number of tokens of the [Corpus](#).

Returns

integer representing the token count

Here is the caller graph for this function:

**6.3.3.6 `boost::shared_ptr< XenFile > Corpus::getXenFile () const`**

Accessor to the [XenFile](#) associated to the [Corpus](#).

Returns

shared pointer to the [XenFile](#)

6.3.3.7 `void Corpus::initialize (boost::shared_ptr< XenFile > ptrData, std::string lg)`

Initialization function from an already instanciated [XenFile](#).

Parameters

<i>ptrData</i>	: shared pointer on a XenFile representing the corpus on disk
<i>lg</i>	: language of the corpus

Here is the caller graph for this function:

**6.3.3.8 `void Corpus::initialize (std::string filePath, std::string lg)`**

Initialization function from a string containing a valid path/file name.

Parameters

<i>filePath</i>	: string containing a valid path/file name
<i>lg</i>	: language of the corpus

6.3.3.9 void Corpus::removeLine (int *line*)

Put the printing status of a line to false.

Parameters

<i>line</i>	: integer representing the line number
-------------	--

The documentation for this class was generated from the following files:

- include/corpus.h
- src/corpus.cpp

6.4 CorpusPair Class Reference

Tiny class holding two related [Corpus](#).

```
#include <StaticData.h>
```

Public Member Functions

- [CorpusPair](#) ()
Default constructor.
- [~CorpusPair](#) ()
Default destructor.
- boost::shared_ptr< [Corpus](#) > [getPtrInCorp](#) () const
Accessor to the in-domain corpus.
- boost::shared_ptr< [Corpus](#) > [getPtrOutCorp](#) () const
Accessor to the out-of-domain corpus.

6.4.1 Detailed Description

Tiny class holding two related [Corpus](#).

6.4.2 Constructor & Destructor Documentation

6.4.2.1 CorpusPair::CorpusPair () [inline]

Default constructor.

6.4.2.2 CorpusPair::~~CorpusPair () [inline]

Default destructor.

6.4.3 Member Function Documentation

6.4.3.1 boost::shared_ptr< [Corpus](#) > CorpusPair::getPtrInCorp () const [inline]

Accessor to the in-domain corpus.

Returns

the in-domain corpus

6.4.3.2 `boost::shared_ptr< Corpus > CorpusPair::getPtrOutCorp () const` `[inline]`

Accessor to the out-of-domain corpus.

Returns

the out-of-domain corpus

The documentation for this class was generated from the following file:

- `include/utills/StaticData.h`

6.5 Eval Class Reference

Evaluation system.

```
#include <eval.h>
```

Public Member Functions

- `Eval ()`
Default constructor.
- `Eval (std::string distFile)`
Constructor from a string.
- `~Eval ()`
Default destructor.
- `void doEval (int high, int low)`
Computes an evaluation bount by the high and low integers (in percentage)
- `void doBP ()`
Computes the best theoretical point based on current evaluation.
- `boost::shared_ptr< EvalMap > getDist () const`
Accessor to the evaluation distribution map.

6.5.1 Detailed Description

Evaluation system.

This class handles the evaluation procedure in XenC, providing mean to perform eval, best point, and getting the results. It uses threads extensively, so please watch your memory usage since there is some memory leaks in SRILM.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 `Eval::Eval ()`

Default constructor.

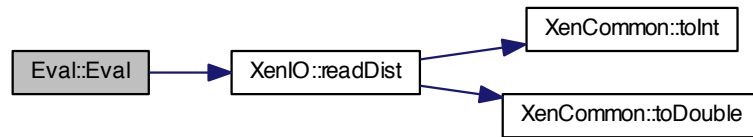
6.5.2.2 `Eval::Eval (std::string distFile)`

Constructor from a string.

Parameters

<i>distFile</i>	: string containing a valid path to the evaluation (*.dist) file, usually used when doing BP
-----------------	--

Here is the call graph for this function:



6.5.2.3 Eval::~Eval ()

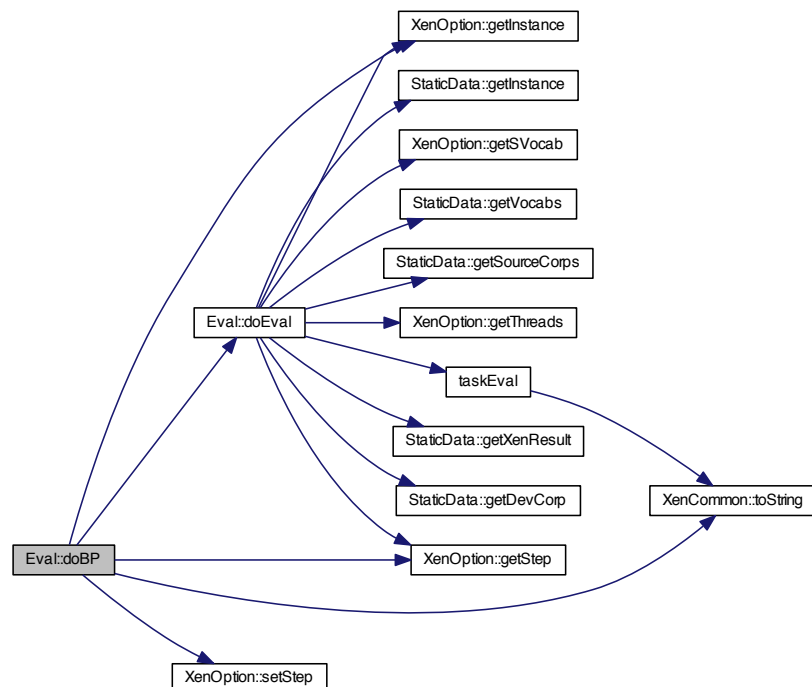
Default destructor.

6.5.3 Member Function Documentation

6.5.3.1 void Eval::doBP ()

Computes the best theoretical point based on current evaluation.

Here is the call graph for this function:



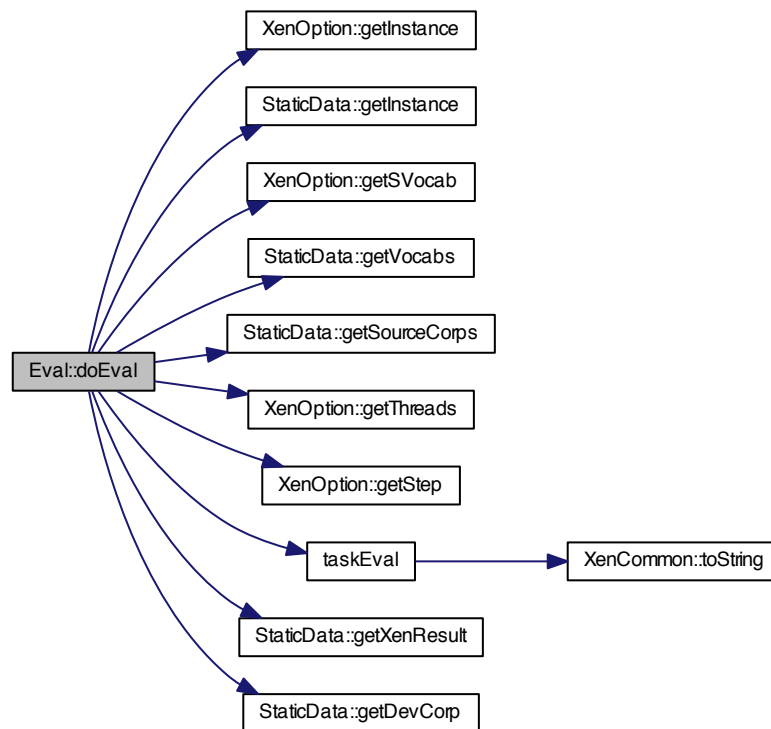
6.5.3.2 void Eval::doEval (int *high*, int *low*)

Computes an evaluation bound by the high and low integers (in percentage)

Parameters

<i>high</i>	: integer representing the upper bound for evaluation
<i>low</i>	: integer representing the lower bound for evaluation

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.3.3 boost::shared_ptr< EvalMap > Eval::getDist () const

Accessor to the evaluation distribution map.

Returns

shared pointer on EvalMap containing all the evaluation results

The documentation for this class was generated from the following files:

- include/eval.h
- src/eval.cpp

6.6 LMPair Class Reference

Tiny class holding two related language models.

```
#include <StaticData.h>
```

Public Member Functions

- [LMPair](#) ()
Default constructor.
- [~LMPair](#) ()
Default destructor.
- boost::shared_ptr< [XenLMsri](#) > [getPtrInLM](#) () const
Accessor to the in-domain language model.
- boost::shared_ptr< [XenLMsri](#) > [getPtrOutLM](#) () const
Accessor to the out-of-domain language model.

6.6.1 Detailed Description

Tiny class holding two related language models.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 LMPair::LMPair () [inline]

Default constructor.

6.6.2.2 LMPair::~~LMPair () [inline]

Default destructor.

6.6.3 Member Function Documentation

6.6.3.1 boost::shared_ptr< XenLMsri > LMPair::getPtrInLM () const [inline]

Accessor to the in-domain language model.

Returns

the in-domain language model

6.6.3.2 `boost::shared_ptr< XenLMsri > LMPair::getPtrOutLM () const` `[inline]`

Accessor to the out-of-domain language model.

Returns

the out-of-domain language model

The documentation for this class was generated from the following file:

- `include/Utils/StaticData.h`

6.7 MeanLMPair Class Reference

Tiny class holding two additional LMs for mean scoring feature.

```
#include <StaticData.h>
```

Public Member Functions

- `MeanLMPair ()`
Default constructor.
- `~MeanLMPair ()`
Default Destructor.
- `boost::shared_ptr< XenLMsri > getPtrOutLM2 () const`
Accessor to the second out-of-domain language model.
- `boost::shared_ptr< XenLMsri > getPtrOutLM3 () const`
Accessor to the third out-of-domain language model.

6.7.1 Detailed Description

Tiny class holding two additional LMs for mean scoring feature.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 `MeanLMPair::MeanLMPair ()` `[inline]`

Default constructor.

6.7.2.2 `MeanLMPair::~~MeanLMPair ()` `[inline]`

Default Destructor.

6.7.3 Member Function Documentation

6.7.3.1 `boost::shared_ptr< XenLMsri > MeanLMPair::getPtrOutLM2 () const` `[inline]`

Accessor to the second out-of-domain language model.

Returns

the second out-of-domain language model

6.7.3.2 `boost::shared_ptr<XenLMsri> MeanLMPair::getPtrOutLM3 () const` `[inline]`

Accessor to the third out-of-domain language model.

Returns

the third out-of-domain language model

The documentation for this class was generated from the following file:

- `include/Utils/StaticData.h`

6.8 MeanPPLPair Class Reference

Tiny class holding two additional [PPL](#) objects for mean scoring feature.

```
#include <StaticData.h>
```

Public Member Functions

- [MeanPPLPair](#) ()
Default constructor.
- [~MeanPPLPair](#) ()
Default Destructor.
- `boost::shared_ptr<PPL> getPtrOutPPL2 () const`
Accessor to the second out-of-domain PPL object.
- `boost::shared_ptr<PPL> getPtrOutPPL3 () const`
Accessor to the third out-of-domain PPL object.

6.8.1 Detailed Description

Tiny class holding two additional [PPL](#) objects for mean scoring feature.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 `MeanPPLPair::MeanPPLPair ()` `[inline]`

Default constructor.

6.8.2.2 `MeanPPLPair::~~MeanPPLPair ()` `[inline]`

Default Destructor.

6.8.3 Member Function Documentation

6.8.3.1 `boost::shared_ptr<PPL> MeanPPLPair::getPtrOutPPL2 () const` `[inline]`

Accessor to the second out-of-domain [PPL](#) object.

Returns

the second out-of-domain [PPL](#) object

6.8.3.2 `boost::shared_ptr< PPL > MeanPPLPair::getPtrOutPPL3 () const` `[inline]`

Accessor to the third out-of-domain [PPL](#) object.

Returns

the third out-of-domain [PPL](#) object

The documentation for this class was generated from the following file:

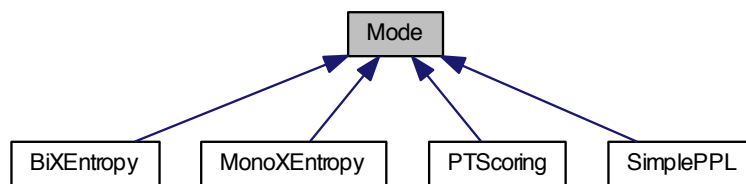
- `include/utis/StaticData.h`

6.9 Mode Class Reference

Filtering modes interface.

```
#include <mode.h>
```

Inheritance diagram for Mode:



Public Member Functions

- virtual int [launch](#) ()=0
Virtual function in charge of launching the implemented mode.
- virtual [~Mode](#) ()=0
Pure virtual destructor.

Static Protected Member Functions

- static int [findSampleSize](#) (boost::shared_ptr< [Corpus](#) > idCorp, boost::shared_ptr< [Corpus](#) > oodCorp)
Finds the optimal sample size for the OOD [Corpus](#).
- static [Corpus](#) [extractSample](#) (boost::shared_ptr< [Corpus](#) > ptrCorp, int sSize, bool mean)
Extracts a random sample from a give [Corpus](#).

6.9.1 Detailed Description

Filtering modes interface.

This class takes the role of an interface to the various XenC filtering modes.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Mode::~~Mode () [pure virtual]

Pure virtual destructor.

6.9.3 Member Function Documentation

6.9.3.1 Corpus Mode::extractSample (boost::shared_ptr< Corpus > ptrCorp, int sSize, bool mean) [static], [protected]

Extracts a random sample from a give [Corpus](#).

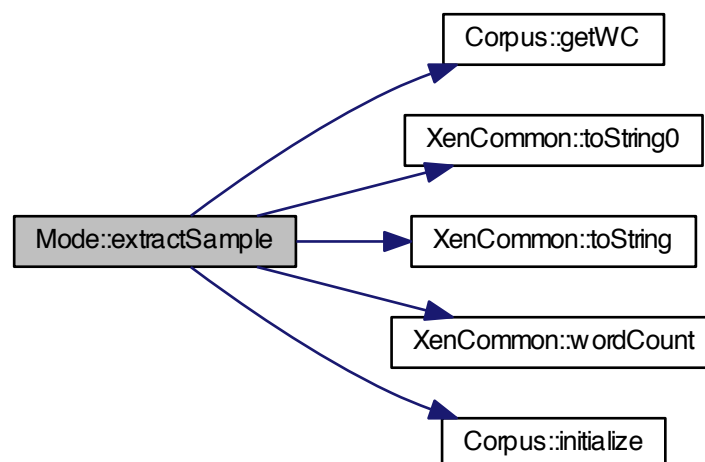
Parameters

<i>ptrCorp</i>	: Corpus from which the sample should be extracted
<i>sSize</i>	: size of the sample to extract
<i>mean</i>	: true if we are in "mean" mode (not the same Corpus filename)

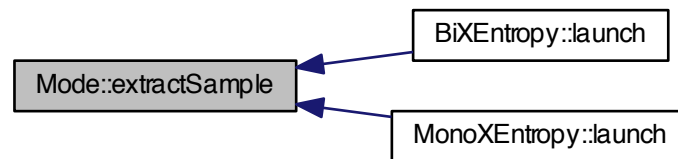
Returns

extracted [Corpus](#) sample

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.3.2 `int Mode::findSampleSize (boost::shared_ptr< Corpus > idCorp, boost::shared_ptr< Corpus > oodCorp)`
`[static], [protected]`

Finds the optimal sample size for the OOD [Corpus](#).

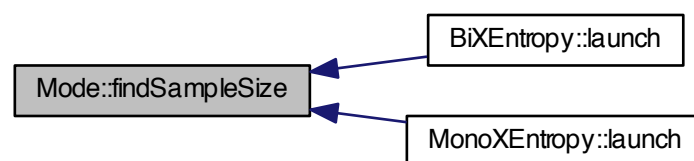
Parameters

<i>idCorp</i>	: in-domain Corpus
<i>oodCorp</i>	: out-of-domain Corpus

Returns

size of the required [Corpus](#) sample in percentage of the whole one

Here is the caller graph for this function:



6.9.3.3 `int Mode::launch ()` `[pure virtual]`

Virtual function in charge of launching the implemented mode.

Implemented in [BiXEntropy](#), [MonoXEntropy](#), [PTScoring](#), and [SimplePPL](#).

The documentation for this class was generated from the following files:

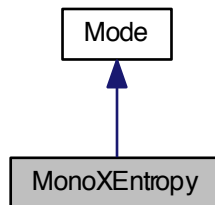
- [include/mode.h](#)
- [src/mode.cpp](#)

6.10 MonoXEntropy Class Reference

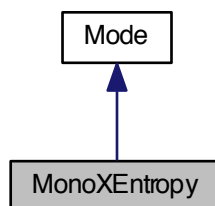
Filtering mode 2: monolingual cross-entropy.

```
#include <monoXEntropy.h>
```

Inheritance diagram for MonoXEntropy:



Collaboration diagram for MonoXEntropy:



Public Member Functions

- [MonoXEntropy](#) ()
Default constructor.
- [~MonoXEntropy](#) ()
Default destructor.
- int [launch](#) ()
Function in charge of launching the filtering mode.

Additional Inherited Members

6.10.1 Detailed Description

Filtering mode 2: monolingual cross-entropy.

This class derived from [Mode](#) handles the second filtering mode: monolingual cross-entropy

6.10.2 Constructor & Destructor Documentation

6.10.2.1 MonoXEntropy::MonoXEntropy ()

Default constructor.

6.10.2.2 MonoXEntropy::~~MonoXEntropy ()

Default destructor.

6.10.3 Member Function Documentation

6.10.3.1 int MonoXEntropy::launch () [virtual]

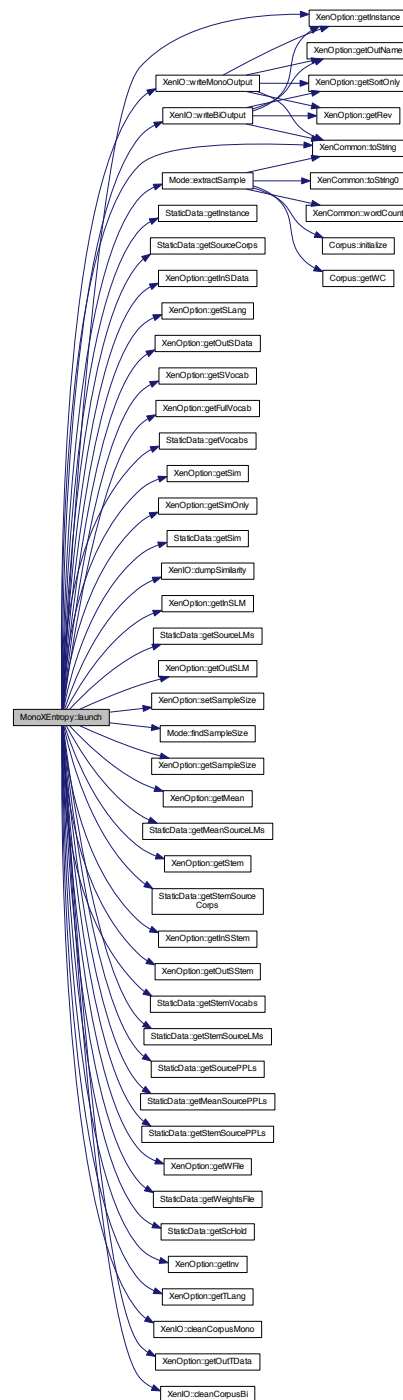
Function in charge of launching the filtering mode.

Returns

0 if the filtering succeeds

Implements [Mode](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- [include/modes/monoXEntropy.h](#)
- [src/modes/monoXEntropy.cpp](#)

6.11 PhraseTable Class Reference

Class handling phrase-table related functionalities.

```
#include <phrasetable.h>
```

Public Member Functions

- [PhraseTable](#) ()
Default constructor.
- void [initialize](#) (boost::shared_ptr< [XenFile](#) > ptrData)
Initialization function from an already instantiated [XenFile](#).
- [~PhraseTable](#) ()
Default destructor.
- boost::shared_ptr< [XenFile](#) > [getXenFile](#) () const
Accessor to the [XenFile](#) associated to the [PhraseTable](#).
- std::string [getSource](#) (int n)
*Accessor to the *nth* source phrase.*
- std::string [getTarget](#) (int n)
*Accessor to the *nth* target phrase.*
- std::string [getScores](#) (int n)
*Accessor to the *nth* scores for the source/target phrase pair.*
- std::string [getAlignment](#) (int n)
*Accessor to the *nth* alignments for the source/target phrase pair.*
- std::string [getCounts](#) (int ph)
*Accessor to the *nth* counts for the source/target phrase pair.*
- std::vector< [SourcePhrase](#) > [getSrcPhrases](#) ()
Accessor to the vector of merged source phrases.
- void [setSrcPhrases](#) (std::vector< [SourcePhrase](#) > vSP)
Mutator to the vector of merged source phrases.
- unsigned int [getSize](#) () const
Accessor to the size of the [PhraseTable](#).

6.11.1 Detailed Description

Class handling phrase-table related functionalities.

This class handles all phrase-table related functionalities and is used in the fourth filtering mode

6.11.2 Constructor & Destructor Documentation

6.11.2.1 [PhraseTable::PhraseTable](#) ()

Default constructor.

6.11.2.2 [PhraseTable::~~PhraseTable](#) ()

Default destructor.

6.11.3 Member Function Documentation

6.11.3.1 `std::string PhraseTable::getAlignment (int n)`

Accessor to the *n*th alignments for the source/target phrase pair.

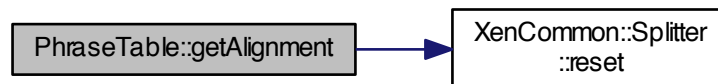
Parameters

<i>n</i>	: integer representing the phrase number
----------	--

Returns

string containing the alignment

Here is the call graph for this function:



6.11.3.2 `std::string PhraseTable::getCounts (int n)`

Accessor to the *n*th counts for the source/target phrase pair.

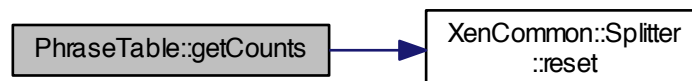
Parameters

<i>n</i>	: integer representing the phrase number
----------	--

Returns

string containing the counts

Here is the call graph for this function:



6.11.3.3 `std::string PhraseTable::getScores (int n)`

Accessor to the *n*th scores for the source/target phrase pair.

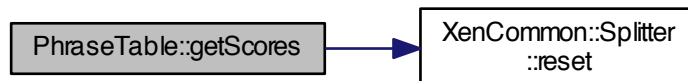
Parameters

n	: integer representing the phrase number
-----	--

Returns

string containing the scores

Here is the call graph for this function:

6.11.3.4 `unsigned int PhraseTable::getSize () const`

Accessor to the size of the [PhraseTable](#).

Returns

unsigned int representing the size

6.11.3.5 `std::string PhraseTable::getSource (int n)`

Accessor to the n th source phrase.

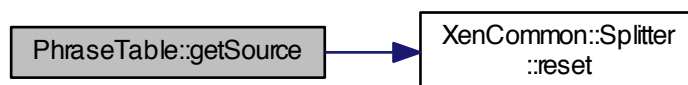
Parameters

n	: integer representing the phrase number
-----	--

Returns

string containing the source phrase

Here is the call graph for this function:



6.11.3.6 `std::vector< SourcePhrase > PhraseTable::getSrcPhrases ()`

Accessor to the vector of merged source phrases.

Returns

vector of merged [SourcePhrase](#)

6.11.3.7 `std::string PhraseTable::getTarget (int n)`

Accessor to the *n*th target phrase.

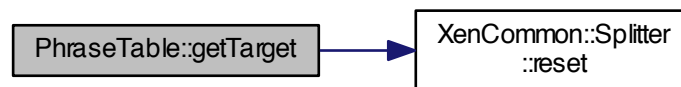
Parameters

<i>n</i>	: integer representing the phrase number
----------	--

Returns

string containing the target phrase

Here is the call graph for this function:



6.11.3.8 `boost::shared_ptr< XenFile > PhraseTable::getXenFile () const`

Accessor to the [XenFile](#) associated to the [PhraseTable](#).

Returns

shared pointer to the [XenFile](#)

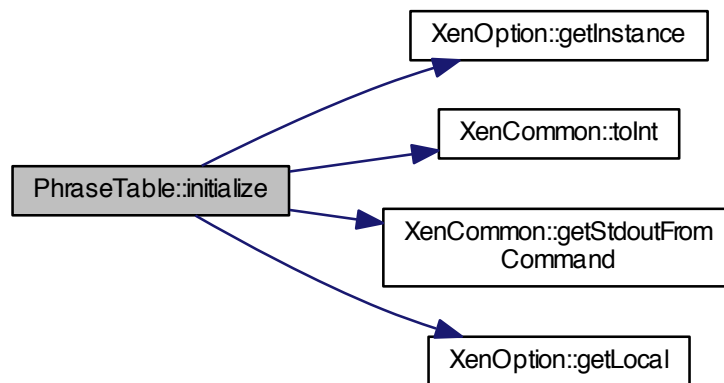
6.11.3.9 `void PhraseTable::initialize (boost::shared_ptr< XenFile > ptrData)`

Initialization function from an already instantiated [XenFile](#).

Parameters

<i>ptrData</i>	: shared pointer on a XenFile representing the PhraseTable on disk
----------------	--

Here is the call graph for this function:



6.11.3.10 `void PhraseTable::setSrcPhrases (std::vector< SourcePhrase > vSP)`

Mutator to the vector of merged source phrases.

Parameters

<code>vSP</code>	: vector of SourcePhrase
------------------	--

The documentation for this class was generated from the following files:

- [include/phrasetable.h](#)
- [src/phrasetable.cpp](#)

6.12 PhraseTablePair Class Reference

Tiny class holding the two phrase-tables.

```
#include <StaticData.h>
```

Public Member Functions

- [PhraseTablePair](#) ()
Default constructor.
- [~PhraseTablePair](#) ()
Default destructor.
- `boost::shared_ptr< PhraseTable > getPtrInPT () const`
Accessor to the in-domain phrase-table.
- `boost::shared_ptr< PhraseTable > getPtrOutPT () const`
Accessor to the out-of-domain phrase-table.

6.12.1 Detailed Description

Tiny class holding the two phrase-tables.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 PhraseTablePair::PhraseTablePair () [inline]

Default constructor.

6.12.2.2 PhraseTablePair::~~PhraseTablePair () [inline]

Default destructor.

6.12.3 Member Function Documentation

6.12.3.1 boost::shared_ptr< PhraseTable > PhraseTablePair::getPtrInPT () const [inline]

Accessor to the in-domain phrase-table.

Returns

the in-domain phrase-table

6.12.3.2 boost::shared_ptr< PhraseTable > PhraseTablePair::getPtrOutPT () const [inline]

Accessor to the out-of-domain phrase-table.

Returns

the out-of-domain phrase-table

The documentation for this class was generated from the following file:

- include/utills/[StaticData.h](#)

6.13 PPL Class Reference

Perplexity/Cross-entropy computations.

```
#include <ppl.h>
```

Public Member Functions

- [PPL](#) ()
Default constructor.
- void [initialize](#) (boost::shared_ptr< [Corpus](#) > ptrCorp, boost::shared_ptr< [XenLMsri](#) > ptrLM)
Initialization function from a [Corpus](#) and a Language Model object.
- void [initialize](#) (boost::shared_ptr< [PhraseTable](#) > ptrPT, boost::shared_ptr< [XenLMsri](#) > ptrLM, bool source)
Initialization function from a [PhraseTable](#) and a Language Model object.

- `~PPL ()`

Default destructor.

- `unsigned int getSize () const`

Accessor to the size of the perplexity/cross-entropy vector.

- `double getPPL (int n)`

Accessor to the n th perplexity score.

- `double getXE (int n)`

Accessor to the n th cross-entropy score.

- `double getCorpPPL ()`

Accessor to the document-level perplexity score.

- `void calcPPLCorpus ()`

Computes the perplexity of a [Corpus](#) sentence by sentence.

- `void calcPPLPhraseTable ()`

Computes the perplexity of a [PhraseTable](#) phrase by phrase.

6.13.1 Detailed Description

Perplexity/Cross-entropy computations.

This class handles the perplexity/cross-entropy computations in XenC. It uses threads extensively to compute scores simultaneously.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 `PPL::PPL ()`

Default constructor.

6.13.2.2 `PPL::~~PPL ()`

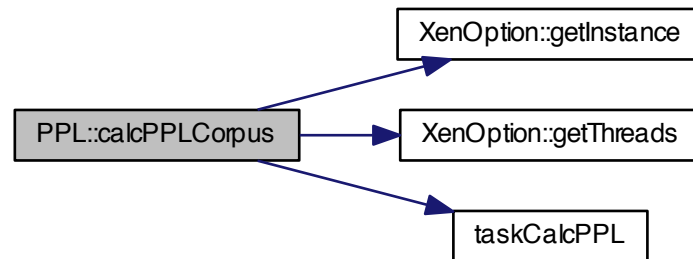
Default destructor.

6.13.3 Member Function Documentation

6.13.3.1 `void PPL::calcPPLCorpus ()`

Computes the perplexity of a [Corpus](#) sentence by sentence.

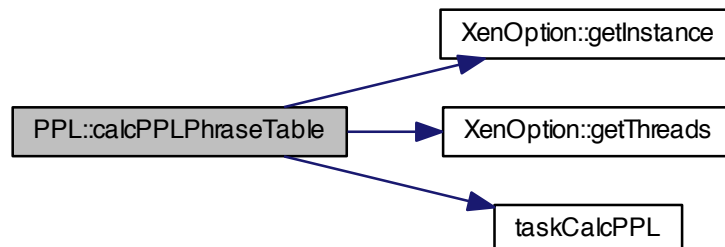
Here is the call graph for this function:



6.13.3.2 void PPL::calcPPLPhraseTable ()

Computes the perplexity of a [PhraseTable](#) phrase by phrase.

Here is the call graph for this function:



6.13.3.3 double PPL::getCorpPPL ()

Accessor to the document-level perplexity score.

Returns

double representing the document perplexity score

6.13.3.4 double PPL::getPPL (int n)

Accessor to the nth perplexity score.

Parameters

<i>n</i>	: integer indicating the position of the score to return
----------	--

Returns

double representing the nth perplexity score

6.13.3.5 unsigned int PPL::getSize () const

Accessor to the size of the perplexity/cross-entropy vector.

Returns

unsigned int representing the size

6.13.3.6 double PPL::getXE (int *n*)

Accessor to the nth cross-entropy score.

Parameters

<i>n</i>	: integer indicating the position of the score to return
----------	--

Returns

double representing the nth cross-entropy score

6.13.3.7 void PPL::initialize (boost::shared_ptr< Corpus > *ptrCorp*, boost::shared_ptr< XenLMsri > *ptrLM*)

Initialization function from a [Corpus](#) and a Language Model object.

Parameters

<i>ptrCorp</i>	: shared pointer on a Corpus to compute perplexity for
<i>ptrLM</i>	: shared pointer on a XenLMsri object to compute perplexity from

6.13.3.8 void PPL::initialize (boost::shared_ptr< PhraseTable > *ptrPT*, boost::shared_ptr< XenLMsri > *ptrLM*, bool *source*)

Initialization function from a [PhraseTable](#) and a Language Model object.

Parameters

<i>ptrPT</i>	: shared pointer on a PhraseTable to compute perplexity for
<i>ptrLM</i>	: shared pointer on a XenLMsri object to compute perplexity from
<i>source</i>	: boolean indicating if we are on source (true) or target (false) side of the PhraseTable

The documentation for this class was generated from the following files:

- [include/ppl.h](#)
- [src/ppl.cpp](#)

6.14 PPLPair Class Reference

Tiny class holding two related [PPL](#) objects.

```
#include <StaticData.h>
```

Public Member Functions

- [PPLPair](#) ()
Default constructor.
- [~PPLPair](#) ()
Default destructor.
- `boost::shared_ptr< PPL > getPtrInPPL () const`
Accessor to the in-domain [PPL](#) object.
- `boost::shared_ptr< PPL > getPtrOutPPL () const`
Accessor to the out-of-domain [PPL](#) object.

6.14.1 Detailed Description

Tiny class holding two related [PPL](#) objects.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 `PPLPair::PPLPair ()` `[inline]`

Default constructor.

6.14.2.2 `PPLPair::~~PPLPair ()` `[inline]`

Default destructor.

6.14.3 Member Function Documentation

6.14.3.1 `boost::shared_ptr< PPL > PPLPair::getPtrInPPL () const` `[inline]`

Accessor to the in-domain [PPL](#) object.

Returns

the in-domain [PPL](#) object

6.14.3.2 `boost::shared_ptr< PPL > PPLPair::getPtrOutPPL () const` `[inline]`

Accessor to the out-of-domain [PPL](#) object.

Returns

the out-of-domain [PPL](#) object

The documentation for this class was generated from the following file:

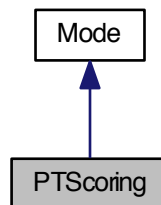
- `include/utis/StaticData.h`

6.15 PTScoring Class Reference

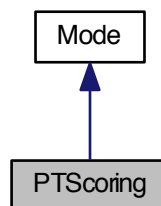
Filtering mode 4: phrase-table cross-entropy.

```
#include <ptScoring.h>
```

Inheritance diagram for PTScoring:



Collaboration diagram for PTScoring:



Public Member Functions

- [PTScoring](#) ()
Default constructor.
- [~PTScoring](#) ()
Default destructor.
- `int` [launch](#) ()
Function in charge of launching the filtering mode.

Additional Inherited Members

6.15.1 Detailed Description

Filtering mode 4: phrase-table cross-entropy.

This class derived from [Mode](#) handles the fourth filtering mode: phrase-table cross-entropy – WARNING: experimental

6.15.2 Constructor & Destructor Documentation

6.15.2.1 PTScoring::PTScoring ()

Default constructor.

6.15.2.2 PTScoring::~~PTScoring ()

Default destructor.

6.15.3 Member Function Documentation

6.15.3.1 int PTScoring::launch () `[virtual]`

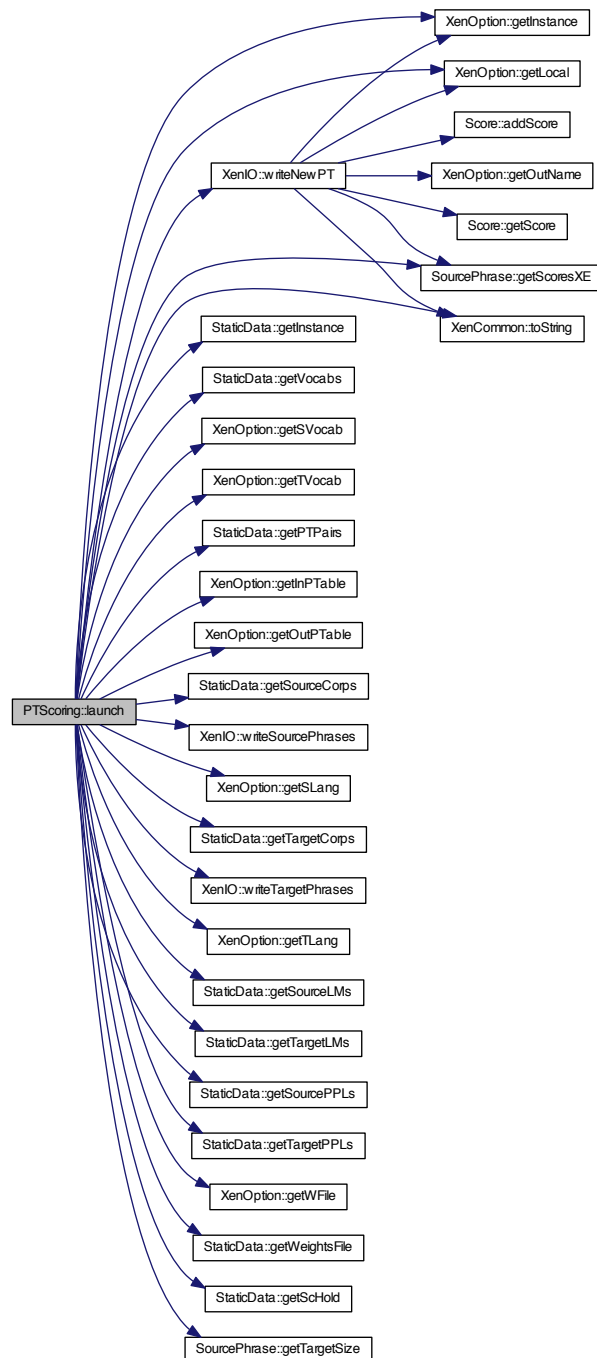
Function in charge of launching the filtering mode.

Returns

0 if the filtering succeeds

Implements [Mode](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- [include/modes/ptScoring.h](#)
- [src/modes/ptScoring.cpp](#)

6.16 Score Class Reference

Class holding the XenC scores representation.

```
#include <score.h>
```

Public Member Functions

- [Score](#) ()
Default constructor.
- [~Score](#) ()
Default destructor.
- void [addScore](#) (double sc)
Adds a score to the vector of doubles.
- void [removeScore](#) (int n)
Removes the nth score from the vector of doubles.
- double [getScore](#) (int n) const
Accessor to the nth score.
- bool [getPrint](#) (int n) const
Accessor to the output status of the nth score.
- unsigned int [getSize](#) () const
Accessor to the size of the scores vector.
- void [calibrate](#) ()
Calibrates the scores distribution between 0 and 1.
- void [inverse](#) ()
Inverts the calibrated score distribution (1 - score)

6.16.1 Detailed Description

Class holding the XenC scores representation.

This class holds the representation of XenC scores. Can add/remove scores and provides access to them.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 [Score::Score](#) ()

Default constructor.

6.16.2.2 [Score::~~Score](#) ()

Default destructor.

6.16.3 Member Function Documentation

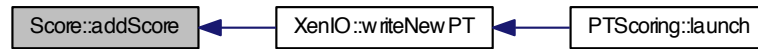
6.16.3.1 [void Score::addScore](#) (double sc)

Adds a score to the vector of doubles.

Parameters

sc	: score to add to the Score holder
--------------------	--

Here is the caller graph for this function:



6.16.3.2 void Score::calibrate ()

Calibrates the scores distribution between 0 and 1.

6.16.3.3 bool Score::getPrint (int *n*) const

Accessor to the output status of the *n*th score.

Parameters

<i>n</i>	: position of the printing status to get
----------	--

Returns

true if the score should be outputted

6.16.3.4 double Score::getScore (int *n*) const

Accessor to the *n*th score.

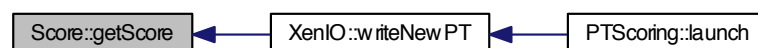
Parameters

<i>n</i>	: position of the score to return
----------	-----------------------------------

Returns

double representing the requested score

Here is the caller graph for this function:



6.16.3.5 unsigned int Score::getSize () const

Accessor to the size of the scores vector.

Returns

unsigned int representing the size

6.16.3.6 void Score::inverse ()

Inverts the calibrated score distribution (1 - score)

6.16.3.7 void Score::removeScore (int *n*)

Removes the *n*th score from the vector of doubles.

Parameters

<i>n</i>	: position of the score to remove in the vector
----------	---

The documentation for this class was generated from the following files:

- include/score.h
- src/score.cpp

6.17 ScoreHolder Class Reference

Tiny class holding three [Score](#) objects (global scores, similarity, cross-entropy)

```
#include <StaticData.h>
```

Public Member Functions

- [ScoreHolder](#) ()
Default constructor.
- [~ScoreHolder](#) ()
Default Destructor.
- boost::shared_ptr< [Score](#) > [getPtrScores](#) () const
Accessor to the global [Score](#) object.
- boost::shared_ptr< [Score](#) > [getPtrScSimil](#) () const
Accessor to the similarity measures [Score](#) object.
- boost::shared_ptr< [Score](#) > [getPtrScXenC](#) () const
Accessor to the cross-entropy [Score](#) object.

6.17.1 Detailed Description

Tiny class holding three [Score](#) objects (global scores, similarity, cross-entropy)

6.17.2 Constructor & Destructor Documentation**6.17.2.1 ScoreHolder::ScoreHolder () [inline]**

Default constructor.

6.17.2.2 ScoreHolder::~~ScoreHolder () [inline]

Default Destructor.

6.17.3 Member Function Documentation

6.17.3.1 boost::shared_ptr< Score > ScoreHolder::getPtrScores () const [inline]

Accessor to the global [Score](#) object.

Returns

the global [Score](#) object

6.17.3.2 boost::shared_ptr< Score > ScoreHolder::getPtrScSimil () const [inline]

Accessor to the similarity measures [Score](#) object.

Returns

the similarity measures [Score](#) object

6.17.3.3 boost::shared_ptr< Score > ScoreHolder::getPtrScXenC () const [inline]

Accessor to the cross-entropy [Score](#) object.

Returns

the cross-entropy [Score](#) object

The documentation for this class was generated from the following file:

- include/utils/[StaticData.h](#)

6.18 Similarity Class Reference

Class taking care of all the similarity measure computations.

```
#include <similarity.h>
```

Public Member Functions

- [Similarity](#) ()
Default constructor.
- void [initialize](#) (boost::shared_ptr< [Corpus](#) > ptrInCorp, boost::shared_ptr< [Corpus](#) > ptrOutCorp, boost::shared_ptr< [XenVocab](#) > ptrVocab)
Initialization function from two [Corpus](#) (in and out-of-domain) and a vocabulary ([XenVocab](#))
- [~Similarity](#) ()
Default destructor.
- float [getSim](#) (int n)
*Accessor to the *n*th sentence similarity measure.*
- unsigned int [getSize](#) () const
Accessor to the size of the similarity map.

6.18.1 Detailed Description

Class taking care of all the similarity measure computations.

This class computes similarity scores between two [Corpus](#) given a vocabulary. It determines the optimal vector for both [Corpus](#), and uses it for similarity computation. WARNING: this feature is still experimental

6.18.2 Constructor & Destructor Documentation

6.18.2.1 `Similarity::Similarity ()`

Default constructor.

6.18.2.2 `Similarity::~~Similarity ()`

Default destructor.

6.18.3 Member Function Documentation

6.18.3.1 `float Similarity::getSim (int n)`

Accessor to the *n*th sentence similarity measure.

Parameters

<i>n</i>	: integer representing the number of the sentence
----------	---

Returns

float representing the similarity measure of the *n*th sentence

6.18.3.2 `unsigned int Similarity::getSize () const`

Accessor to the size of the similarity map.

Returns

unsigned int representing the size

6.18.3.3 `void Similarity::initialize (boost::shared_ptr< Corpus > ptrInCorp, boost::shared_ptr< Corpus > ptrOutCorp, boost::shared_ptr< XenVocab > ptrVocab)`

Initialization function from two [Corpus](#) (in and out-of-domain) and a vocabulary ([XenVocab](#))

Parameters

<i>ptrInCorp</i>	: shared pointer on the in-domain Corpus
<i>ptrOutCorp</i>	: shared pointer on the out-of-domain Corpus
<i>ptrVocab</i>	: shared pointer on the common XenVocab (usually the in-domain one)

The documentation for this class was generated from the following files:

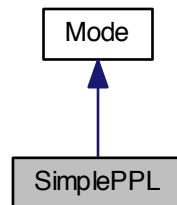
- include/[similarity.h](#)
- src/[similarity.cpp](#)

6.19 SimplePPL Class Reference

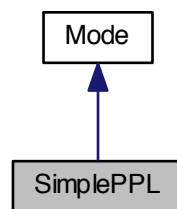
Filtering mode 1: simple perplexity.

```
#include <simplePPL.h>
```

Inheritance diagram for SimplePPL:



Collaboration diagram for SimplePPL:



Public Member Functions

- [SimplePPL \(\)](#)
Default constructor.
- [~SimplePPL \(\)](#)
Default destructor.
- `int` [launch \(\)](#)
Function in charge of launching the filtering mode.

Additional Inherited Members

6.19.1 Detailed Description

Filtering mode 1: simple perplexity.

This class derived from [Mode](#) handles the first filtering mode: simple perplexity

6.19.2 Constructor & Destructor Documentation

6.19.2.1 SimplePPL::SimplePPL ()

Default constructor.

6.19.2.2 SimplePPL::~~SimplePPL ()

Default destructor.

6.19.3 Member Function Documentation

6.19.3.1 int SimplePPL::launch () [virtual]

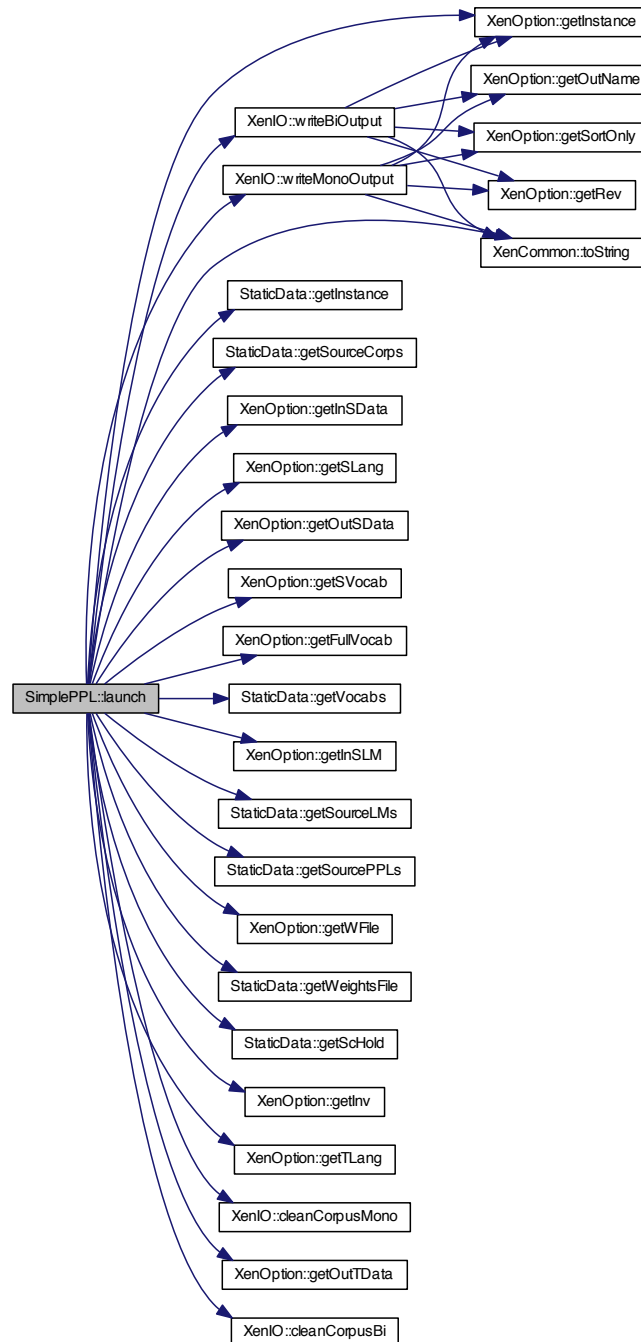
Function in charge of launching the filtering mode.

Returns

0 if the filtering succeeds

Implements [Mode](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- `include/modes/simplePPL.h`
- `src/modes/simplePPL.cpp`

6.20 SourcePhrase Class Reference

Class holding a merged source phrase and all associated data.

```
#include <sourcephrase.h>
```

Public Member Functions

- [SourcePhrase](#) (std::string src)
Constructor from a string.
- [~SourcePhrase](#) ()
Default destructor.
- std::string [getSource](#) () const
Accessor to the source phrase.
- unsigned int [getTargetSize](#) () const
Accessor to the size of the target phrases associated to the source phrase.
- boost::shared_ptr< [Score](#) > [getScoresXE](#) () const
Accessor to the vector of cross-entropy scores for the target phrases.
- void [addTarget](#) (std::string s)
Associates a target phrase to the source phrase.
- void [addScores](#) (std::string s)
Associates a phrase table scores sequence to the source phrase.
- void [addAlignments](#) (std::string s)
Associates alignments to the source phrase.
- void [addCounts](#) (std::string s)
Associates counts to the source phrase.

6.20.1 Detailed Description

Class holding a merged source phrase and all associated data.

This class holds a merged source phrase from a [PhraseTable](#), along with target phrases, scores, alignments and counts.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 SourcePhrase::SourcePhrase (std::string src)

Constructor from a string.

Parameters

<code>src</code>	: string representing the source phrase
------------------	---

6.20.2.2 SourcePhrase::~SourcePhrase ()

Default destructor.

6.20.3 Member Function Documentation

6.20.3.1 void SourcePhrase::addAlignments (std::string s)

Associates alignments to the source phrase.

Parameters

s	: the alignments to add to the source phrase
---	--

6.20.3.2 void SourcePhrase::addCounts (std::string s)

Associates counts to the source phrase.

Parameters

s	: the counts to add to the source phrase
---	--

6.20.3.3 void SourcePhrase::addScores (std::string s)

Associates a phrase table scores sequence to the source phrase.

Parameters

s	: the scores sequence to add to the source phrase
---	---

6.20.3.4 void SourcePhrase::addTarget (std::string s)

Associates a target phrase to the source phrase.

Parameters

s	: the target phrase to add to the source phrase
---	---

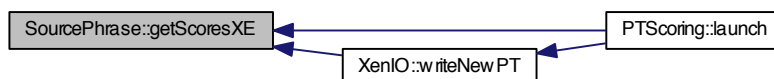
6.20.3.5 boost::shared_ptr< Score > SourcePhrase::getScoresXE () const

Accessor to the vector of cross-entropy scores for the target phrases.

Returns

a shared pointer on a [Score](#) object containing the scores

Here is the caller graph for this function:



6.20.3.6 `std::string SourcePhrase::getSource () const`

Accessor to the source phrase.

Returns

the source phrase

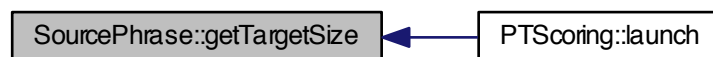
6.20.3.7 `unsigned int SourcePhrase::getTargetSize () const`

Accessor to the size of the target phrases associated to the source phrase.

Returns

the size of the target phrases vector

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- [include/sourcephrase.h](#)
- [src/sourcephrase.cpp](#)

6.21 `XenCommon::Splitter` Class Reference

Class defining a splitter.

```
#include <common.h>
```

Public Types

- `typedef std::vector
< std::string >::size_type size_type`

Public Member Functions

- [Splitter](#) ()
- [Splitter](#) (const std::string &src, const std::string &delim)
- std::string & [operator\[\]](#) (size_type i)
- [size_type size](#) () const
- void [reset](#) (const std::string &src, const std::string &delim)

6.21.1 Detailed Description

Class defining a splitter.

Class to split a string into vector of string, given a potentially multi-character delimiter (like "|||" in a phrase table for instance)

6.21.2 Member Typedef Documentation

6.21.2.1 `typedef std::vector<std::string>::size_type XenCommon::Splitter::size_type`

6.21.3 Constructor & Destructor Documentation

6.21.3.1 `XenCommon::Splitter::Splitter ()` `[inline]`

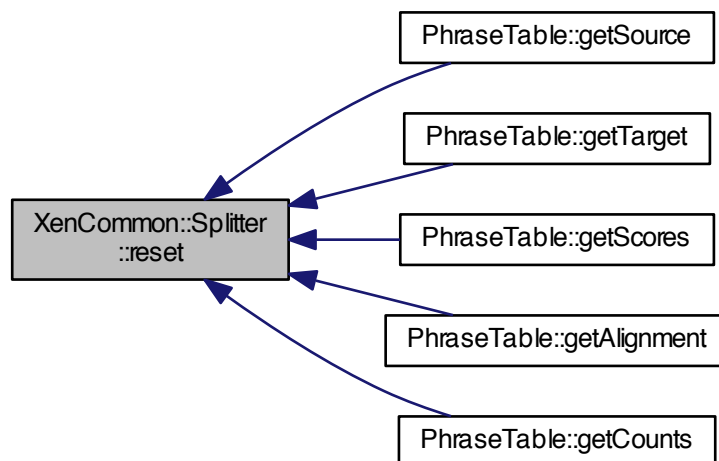
6.21.3.2 `XenCommon::Splitter::Splitter (const std::string & src, const std::string & delim)` `[inline]`

6.21.4 Member Function Documentation

6.21.4.1 `std::string& XenCommon::Splitter::operator[] (size_type i)` `[inline]`

6.21.4.2 `void XenCommon::Splitter::reset (const std::string & src, const std::string & delim)` `[inline]`

Here is the caller graph for this function:



6.21.4.3 `size_type XenCommon::Splitter::size () const` `[inline]`

The documentation for this class was generated from the following file:

- `include/utis/common.h`

6.22 StaticData Class Reference

Class gathering all data used and generated by XenC.

```
#include <StaticData.h>
```

Static Public Member Functions

- static [StaticData](#) * [getInstance](#) ()
Accessor to the instance of the singleton [StaticData](#) object.
- static void [deleteInstance](#) ()
Deletes the unique instance of the [StaticData](#) singleton.
- static boost::shared_ptr
 < [CorpusPair](#) > [getSourceCorps](#) ()
Accessor to the source language [Corpus](#) Pair.
- static boost::shared_ptr
 < [CorpusPair](#) > [getTargetCorps](#) ()
Accessor to the target language [Corpus](#).
- static boost::shared_ptr< [LMPair](#) > [getSourceLMs](#) ()
Accessor to the source language models.
- static boost::shared_ptr< [LMPair](#) > [getTargetLMs](#) ()
Accessor to the target language models.
- static boost::shared_ptr
 < [VocabPair](#) > [getVocabs](#) ()
Accessor to the vocabularies.
- static boost::shared_ptr< [PPLPair](#) > [getSourcePPLs](#) ()
Accessor to the source language [PPL](#) objects.
- static boost::shared_ptr< [PPLPair](#) > [getTargetPPLs](#) ()
Accessor to the target language [PPL](#) objects.
- static boost::shared_ptr
 < [PhraseTablePair](#) > [getPTPairs](#) ()
Accessor to the phrase-tables.
- static boost::shared_ptr
 < [MeanLMPair](#) > [getMeanSourceLMs](#) ()
Accessor to the mean source language models.
- static boost::shared_ptr
 < [MeanLMPair](#) > [getMeanTargetLMs](#) ()
Accessor to the mean target language models.
- static boost::shared_ptr
 < [MeanPPLPair](#) > [getMeanSourcePPLs](#) ()
Accessor to the mean source [PPL](#) objects.
- static boost::shared_ptr
 < [MeanPPLPair](#) > [getMeanTargetPPLs](#) ()
Accessor to the mean target [PPL](#) objects.
- static boost::shared_ptr
 < [CorpusPair](#) > [getStemSourceCorps](#) ()
Accessor to the source language stem [Corpus](#) Pair.
- static boost::shared_ptr
 < [CorpusPair](#) > [getStemTargetCorps](#) ()
Accessor to the target language stem [Corpus](#) Pair.
- static boost::shared_ptr< [LMPair](#) > [getStemSourceLMs](#) ()
Accessor to the source language stem language models.

- static boost::shared_ptr< [LMPair](#) > [getStemTargetLMs](#) ()
Accessor to the target language stem language models.
- static boost::shared_ptr< [VocabPair](#) > [getStemVocabs](#) ()
Accessor to the stem vocabularies.
- static boost::shared_ptr< [PPLPair](#) > [getStemSourcePPLs](#) ()
Accessor to the source language stem [PPL](#) objects.
- static boost::shared_ptr< [PPLPair](#) > [getStemTargetPPLs](#) ()
Accessor to the target language stem [PPL](#) objects.
- static boost::shared_ptr< [Similarity](#) > [getSim](#) ()
Accessor to the [Similarity](#) measures object.
- static boost::shared_ptr< [ScoreHolder](#) > [getScHold](#) ()
Accessor to the [ScoreHolder](#) object.
- static boost::shared_ptr< [Wfile](#) > [getWeightsFile](#) ()
Accessor to the weights file.
- static boost::shared_ptr< [XenResult](#) > [getXenResult](#) ()
Accessor to the filtering result file.
- static boost::shared_ptr< [Corpus](#) > [getDevCorp](#) ()
Accessor to the development [Corpus](#).

6.22.1 Detailed Description

Class gathering all data used and generated by XenC.

6.22.2 Member Function Documentation

6.22.2.1 void StaticData::deleteInstance () [static]

Deletes the unique instance of the [StaticData](#) singleton.

Here is the caller graph for this function:



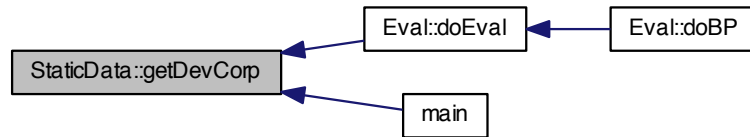
6.22.2.2 boost::shared_ptr< [Corpus](#) > StaticData::getDevCorp () [static]

Accessor to the development [Corpus](#).

Returns

the development [Corpus](#)

Here is the caller graph for this function:



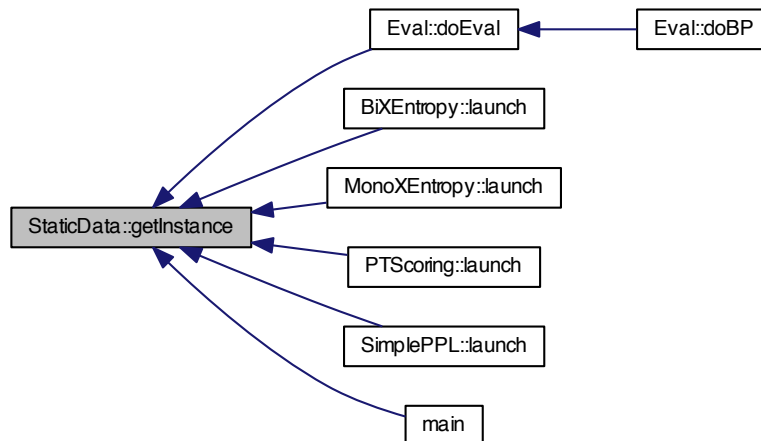
6.22.2.3 `StaticData * StaticData::getInstance () [static]`

Accessor to the instance of the singleton [StaticData](#) object.

Returns

the [StaticData](#) unique instance

Here is the caller graph for this function:



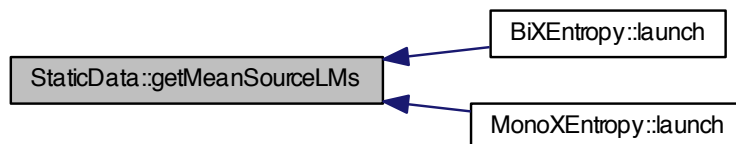
6.22.2.4 `boost::shared_ptr< MeanLMPair > StaticData::getMeanSourceLMs () [static]`

Accessor to the mean source language models.

Returns

the mean source language models

Here is the caller graph for this function:

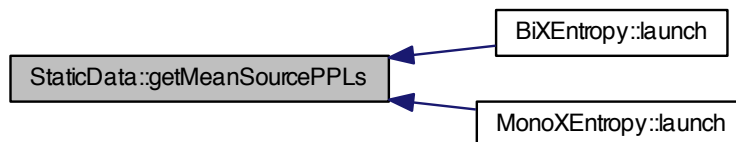
**6.22.2.5** `boost::shared_ptr< MeanPPLPair > StaticData::getMeanSourcePPLs () [static]`

Accessor to the mean source [PPL](#) objects.

Returns

the mean source [PPL](#) objects

Here is the caller graph for this function:

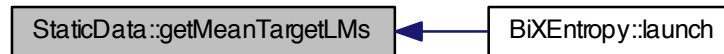
**6.22.2.6** `boost::shared_ptr< MeanLMPair > StaticData::getMeanTargetLMs () [static]`

Accessor to the mean target language models.

Returns

the mean target language models

Here is the caller graph for this function:



6.22.2.7 `boost::shared_ptr< MeanPPLPair > StaticData::getMeanTargetPPLs () [static]`

Accessor to the mean target [PPL](#) objects.

Returns

the mean target [PPL](#) objects

Here is the caller graph for this function:



6.22.2.8 `boost::shared_ptr< PhraseTablePair > StaticData::getPTPairs () [static]`

Accessor to the phrase-tables.

Returns

the phrase-tables

Here is the caller graph for this function:



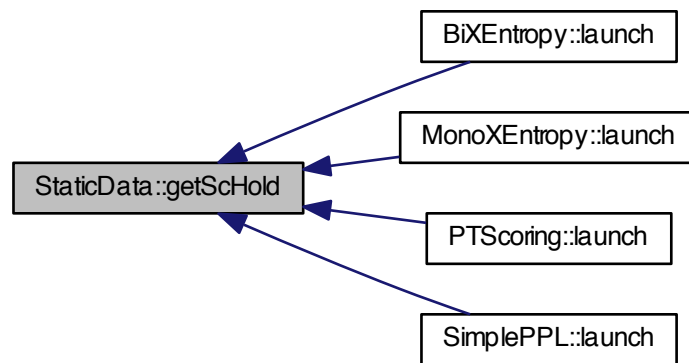
6.22.2.9 `boost::shared_ptr< ScoreHolder > StaticData::getScHold () [static]`

Accessor to the [ScoreHolder](#) object.

Returns

the [ScoreHolder](#) object

Here is the caller graph for this function:



6.22.2.10 `boost::shared_ptr< Similarity > StaticData::getSim () [static]`

Accessor to the [Similarity](#) measures object.

Returns

the [Similarity](#) measures object

Here is the caller graph for this function:



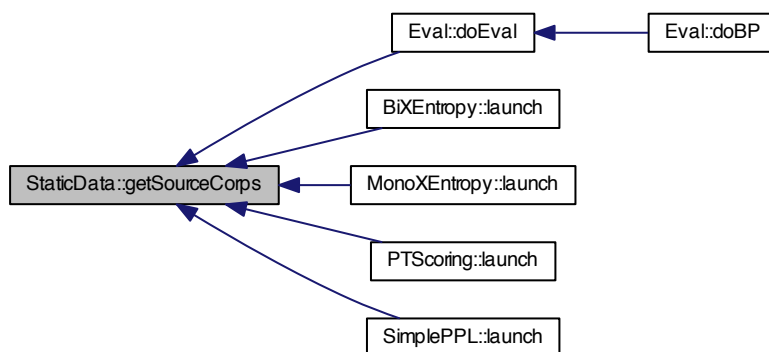
6.22.2.11 `boost::shared_ptr< CorpusPair > StaticData::getSourceCorps () [static]`

Accessor to the source language [Corpus](#) Pair.

Returns

the source language [Corpus](#) Pair

Here is the caller graph for this function:



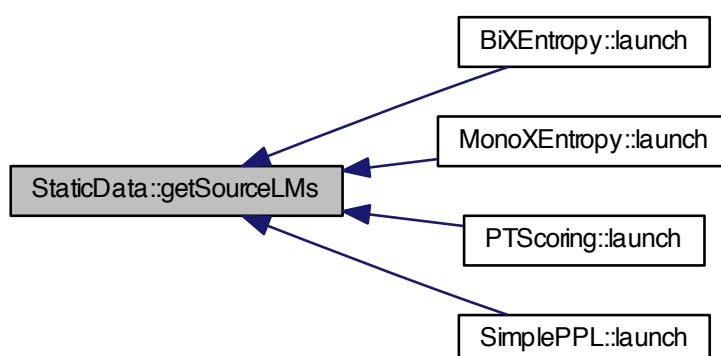
6.22.2.12 `boost::shared_ptr< LMPair > StaticData::getSourceLMs () [static]`

Accessor to the source language models.

Returns

the source language models

Here is the caller graph for this function:



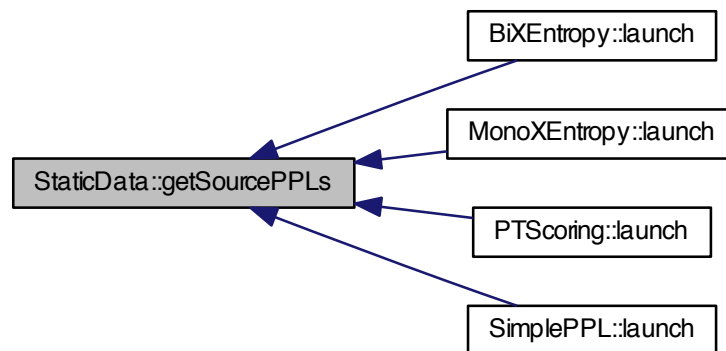
6.22.2.13 `boost::shared_ptr< PPLPair > StaticData::getSourcePPLs () [static]`

Accessor to the source language [PPL](#) objects.

Returns

the source language [PPL](#) objects

Here is the caller graph for this function:



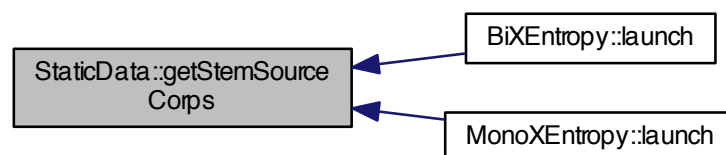
6.22.2.14 `boost::shared_ptr< CorpusPair > StaticData::getStemSourceCorps () [static]`

Accessor to the source language stem [Corpus](#) Pair.

Returns

the source language stem [Corpus](#) Pair

Here is the caller graph for this function:



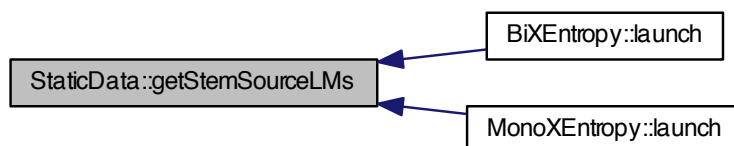
6.22.2.15 `boost::shared_ptr< LMPair > StaticData::getStemSourceLMs () [static]`

Accessor to the source language stem language models.

Returns

the source language stem language models

Here is the caller graph for this function:



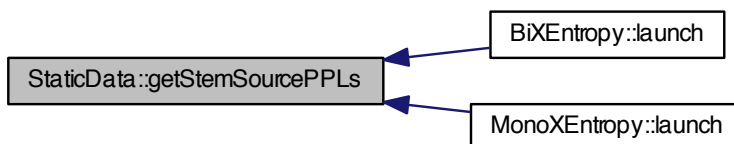
6.22.2.16 `boost::shared_ptr< PPLPair > StaticData::getStemSourcePPLs () [static]`

Accessor to the source language stem [PPL](#) objects.

Returns

the source language stem [PPL](#) objects

Here is the caller graph for this function:



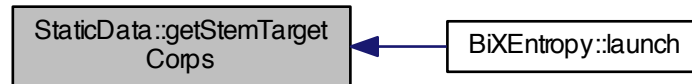
6.22.2.17 `boost::shared_ptr< CorpusPair > StaticData::getStemTargetCorps () [static]`

Accessor to the target language stem [Corpus](#) Pair.

Returns

the target language stem [Corpus](#) Pair

Here is the caller graph for this function:



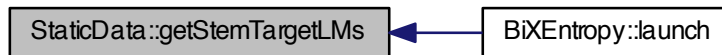
6.22.2.18 `boost::shared_ptr< LMPair > StaticData::getStemTargetLMs () [static]`

Accessor to the target language stem language models.

Returns

the target language stem language models

Here is the caller graph for this function:



6.22.2.19 `boost::shared_ptr< PPLPair > StaticData::getStemTargetPPLs () [static]`

Accessor to the target language stem [PPL](#) objects.

Returns

the target language stem [PPL](#) objects

Here is the caller graph for this function:



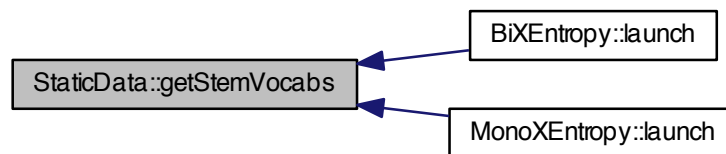
6.22.2.20 `boost::shared_ptr< VocabPair > StaticData::getStemVocabs () [static]`

Accessor to the stem vocabularies.

Returns

the stem vocabularies

Here is the caller graph for this function:



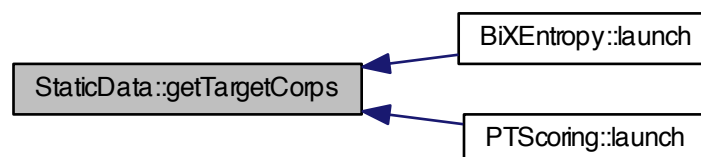
6.22.2.21 `boost::shared_ptr< CorpusPair > StaticData::getTargetCorps () [static]`

Accessor to the target language [Corpus](#).

Returns

the target language [Corpus](#) Pair

Here is the caller graph for this function:



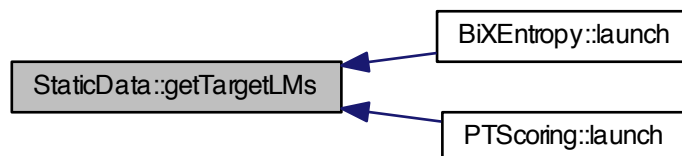
6.22.2.22 `boost::shared_ptr< LMPair > StaticData::getTargetLMs () [static]`

Accessor to the target language models.

Returns

the target language models

Here is the caller graph for this function:



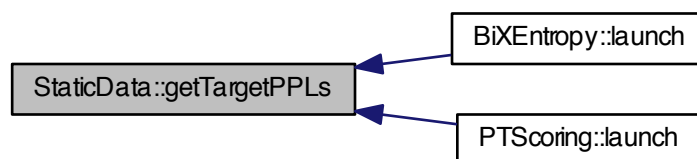
6.22.2.23 `boost::shared_ptr< PPLPair > StaticData::getTargetPPLs () [static]`

Accessor to the target language PPL objects.

Returns

the target language PPL objects

Here is the caller graph for this function:



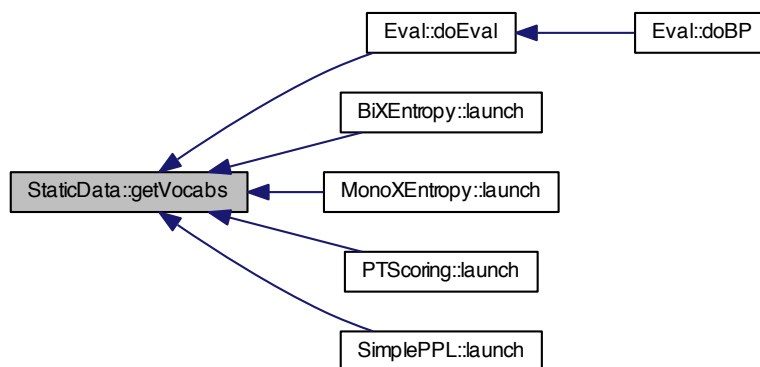
6.22.2.24 `boost::shared_ptr< VocabPair > StaticData::getVocabs () [static]`

Accessor to the vocabularies.

Returns

the vocabularies

Here is the caller graph for this function:



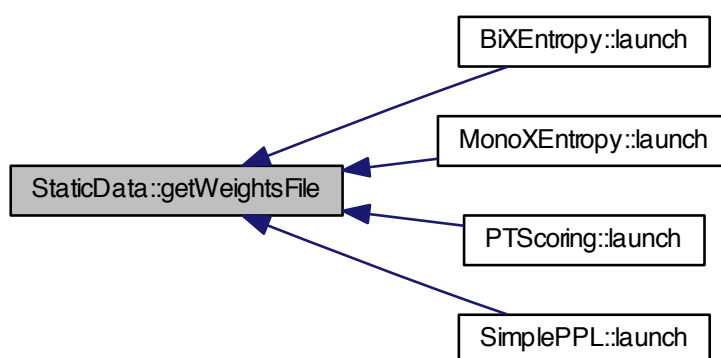
6.22.2.25 `boost::shared_ptr< Wfile > StaticData::getWeightsFile () [static]`

Accessor to the weights file.

Returns

the weights file

Here is the caller graph for this function:



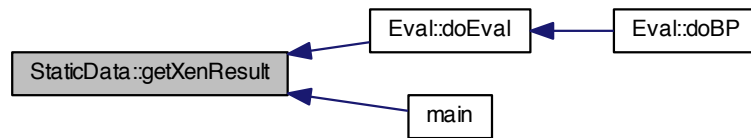
6.22.2.26 `boost::shared_ptr< XenResult > StaticData::getXenResult () [static]`

Accessor to the filtering result file.

Returns

the filtering result file

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- include/utils/[StaticData.h](#)
- src/utils/[StaticData.cpp](#)

6.23 VocabPair Class Reference

Tiny class holding the two vocabularies.

```
#include <StaticData.h>
```

Public Member Functions

- [VocabPair](#) ()
Default constructor.
- [~VocabPair](#) ()
Default destructor.
- `boost::shared_ptr< XenVocab > getPtrSourceVoc () const`
Accessor to the source vocabulary.
- `boost::shared_ptr< XenVocab > getPtrTargetVoc () const`
Accessor to the target vocabulary.

6.23.1 Detailed Description

Tiny class holding the two vocabularies.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 VocabPair::VocabPair () [inline]

Default constructor.

6.23.2.2 VocabPair::~~VocabPair () [inline]

Default destructor.

6.23.3 Member Function Documentation

6.23.3.1 `boost::shared_ptr< XenVocab > VocabPair::getPtrSourceVoc () const` `[inline]`

Accessor to the source vocabulary.

Returns

the source vocabulary

6.23.3.2 `boost::shared_ptr< XenVocab > VocabPair::getPtrTargetVoc () const` `[inline]`

Accessor to the target vocabulary.

Returns

the target vocabulary

The documentation for this class was generated from the following file:

- `include/utils/StaticData.h`

6.24 Wfile Class Reference

Class handling a file with values intended at weighting XenC scores.

```
#include <wfile.h>
```

Public Member Functions

- `Wfile ()`
Default constructor.
- `void initialize (boost::shared_ptr< XenFile > ptrFile)`
Initialization function from an already instanciated `XenFile`.
- `~Wfile ()`
Default destructor.
- `double getWeight (int n)`
Accessor to the n th weight of the file.
- `unsigned int getSize () const`
Accessor to the size of the weights file.

6.24.1 Detailed Description

Class handling a file with values intended at weighting XenC scores.

The values file should contain one value per line, these values can also be in the log domain.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 `Wfile::Wfile ()`

Default constructor.

6.24.2.2 Wfile::~~Wfile ()

Default destructor.

6.24.3 Member Function Documentation

6.24.3.1 unsigned int Wfile::getSize () const

Accessor to the size of the weights file.

Returns

the size of the weights file

6.24.3.2 double Wfile::getWeight (int *n*)

Accessor to the *n*th weight of the file.

Parameters

<i>n</i>	: the number of the weight line in the file
----------	---

Returns

the requested weight

6.24.3.3 void Wfile::initialize (boost::shared_ptr< XenFile > *ptrFile*)

Initialization function from an already instanciated [XenFile](#).

Parameters

<i>ptrFile</i>	: the weights file
----------------	--------------------

The documentation for this class was generated from the following files:

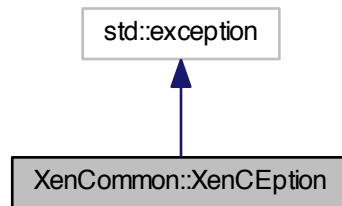
- include/[wfile.h](#)
- src/[wfile.cpp](#)

6.25 XenCommon::XenCEption Struct Reference

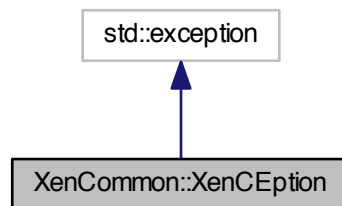
XenC exception structure.

```
#include "utils/common.h"
```

Inheritance diagram for `XenCommon::XenCEption`:



Collaboration diagram for `XenCommon::XenCEption`:



Public Member Functions

- [XenCEption](#) (`std::string ss`)
Exception constructor.
- virtual [~XenCEption](#) () throw ()
Exception destructor.
- const char * [what](#) () const throw ()
Accessor to the exception message.

Public Attributes

- `std::string s`
The exception message.

6.25.1 Detailed Description

XenC exception structure.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 XenCommon::XenCEption::XenCEption (std::string ss) [inline]

Exception constructor.

6.25.2.2 XenCommon::XenCEption::~~XenCEption () throw () [inline],[virtual]

Exception desctructor.

6.25.3 Member Function Documentation

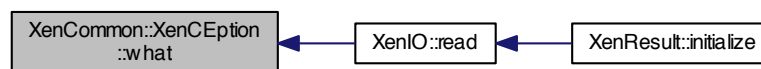
6.25.3.1 const char * XenCommon::XenCEption::what () const throw () [inline]

Accessor to the exception message.

Returns

the exception message

Here is the caller graph for this function:



6.25.4 Member Data Documentation

6.25.4.1 std::string XenCommon::XenCEption::s

The exception message.

The documentation for this struct was generated from the following file:

- include/utils/[common.h](#)

6.26 XenFile Class Reference

Class providing some basic functions around files.

```
#include <xenfile.h>
```

Public Member Functions

- [XenFile](#) ()
Default constructor.
- void [initialize](#) (std::string name)
Initialization function from a string.

- `~XenFile ()`
Default destructor.
- `std::string getFileName () const`
Accessor to the file name.
- `std::string getPrefix ()`
Accessor to the prefix of the file name (before the dot)
- `std::string getExt ()`
Accessor to the extension of the file name (after the dot)
- `std::string getDirName () const`
Accessor to the file's directory name.
- `std::string getFullPath () const`
Accessor to the file's full path.
- `bool isGZ () const`
Tries to guess if the file is a gzip or plain text.

6.26.1 Detailed Description

Class providing some basic functions around files.

This class handles the files used in XenC and provides some basic functionalities around them used widely in XenC.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 `XenFile::XenFile ()`

Default constructor.

6.26.2.2 `XenFile::~~XenFile ()`

Default destructor.

6.26.3 Member Function Documentation

6.26.3.1 `std::string XenFile::getDirName () const`

Accessor to the file's directory name.

Returns

the file's directory name

6.26.3.2 `std::string XenFile::getExt ()`

Accessor to the extension of the file name (after the dot)

Returns

the file name's extension

6.26.3.3 `std::string XenFile::getFileName () const`

Accessor to the file name.

Returns

the file name (and only the file name)

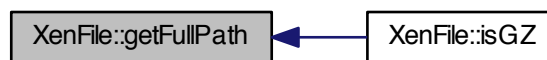
6.26.3.4 `std::string XenFile::getFullPath () const`

Accessor to the file's full path.

Returns

the file's full path

Here is the caller graph for this function:



6.26.3.5 `std::string XenFile::getPrefix ()`

Accessor to the prefix of the file name (before the dot)

Returns

the file name's prefix

6.26.3.6 `void XenFile::initialize (std::string name)`

Initialization function from a string.

Parameters

<i>name</i>	: the file to handle
-------------	----------------------

6.26.3.7 `bool XenFile::isGZ () const`

Tries to guess if the file is a gzip or plain text.

Returns

true if the file is gzipped

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- include/xenfile.h
- src/xenfile.cpp

6.27 XenIO Class Reference

Class handling all input/output operations of XenC.

```
#include <xenio.h>
```

Static Public Member Functions

- static void [cleanCorpusMono](#) (boost::shared_ptr< [Corpus](#) > ptrCorp, boost::shared_ptr< [Score](#) > ptrScore)
Monolingual corpus cleaning (ensures no empty lines)
- static void [cleanCorpusBi](#) (boost::shared_ptr< [Corpus](#) > ptrCorpSource, boost::shared_ptr< [Corpus](#) > ptrCorpTarget, boost::shared_ptr< [Score](#) > ptrScore)
Bilingual corpus cleaning (ensures no empty lines)
- static void [writeMonoOutput](#) (boost::shared_ptr< [Corpus](#) > ptrCorp, boost::shared_ptr< [Score](#) > ptrScore)
Writes monolingual scored/sorted result files.
- static void [writeBiOutput](#) (boost::shared_ptr< [Corpus](#) > ptrCorpSource, boost::shared_ptr< [Corpus](#) > ptrCorpTarget, boost::shared_ptr< [Score](#) > ptrScore)
Writes bilingual scored/sorted result files.
- static void [writeNewPT](#) (boost::shared_ptr< [PhraseTable](#) > ptrPT, boost::shared_ptr< [Score](#) > ptrScore)
Writes a new rescored phrase-table.
- static std::string [writeSourcePhrases](#) (boost::shared_ptr< [PhraseTable](#) > ptrPT)
Writes a phrase-table's source phrases.
- static std::string [writeTargetPhrases](#) (boost::shared_ptr< [PhraseTable](#) > ptrPT)
Writes a phrase-table's target phrases.
- static void [writeEval](#) (boost::shared_ptr< [EvalMap](#) > ptrEvalMap, std::string distName)
Writes an evaluation/best point distribution file.
- static void [dumpSimilarity](#) (boost::shared_ptr< [Corpus](#) > ptrCorp, boost::shared_ptr< [Similarity](#) > ptrSim)
Dumps the [Similarity](#) measures of a [Corpus](#).
- static std::vector< std::string > [read](#) (boost::shared_ptr< [XenFile](#) > ptrFile)
Reads a file (plain text/gzipped)
- static boost::shared_ptr< [EvalMap](#) > [readDist](#) (std::string distFile)
Reads a evaluation/best point distribution file.

6.27.1 Detailed Description

Class handling all input/output operations of XenC.

This class handles file reading/writing (plain text or compressed), corpus cleaning, phrases and phrase-tables writing, similarity dumping...

6.27.2 Member Function Documentation

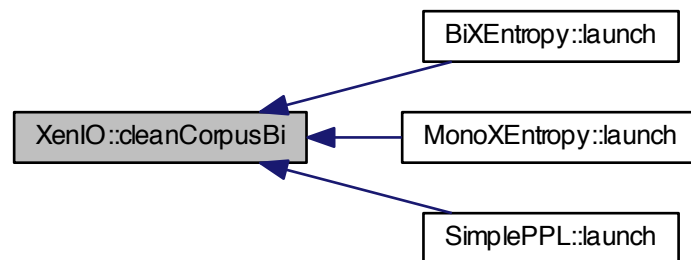
6.27.2.1 `void XenIO::cleanCorpusBi (boost::shared_ptr< Corpus > ptrCorpSource, boost::shared_ptr< Corpus > ptrCorpTarget, boost::shared_ptr< Score > ptrScore) [static]`

Bilingual corpus cleaning (ensures no empty lines)

Parameters

<i>ptrCorpSource</i>	: the source language Corpus to clean
<i>ptrCorpTarget</i>	: the target language Corpus to clean
<i>ptrScore</i>	: the associated Score object to clean

Here is the caller graph for this function:



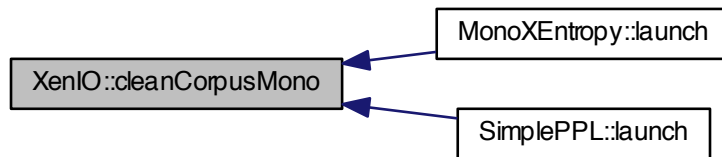
6.27.2.2 `void XenIO::cleanCorpusMono (boost::shared_ptr< Corpus > ptrCorp, boost::shared_ptr< Score > ptrScore) [static]`

Monolingual corpus cleaning (ensures no empty lines)

Parameters

<i>ptrCorp</i>	: the Corpus to clean
<i>ptrScore</i>	: the associated Score object to clean

Here is the caller graph for this function:



6.27.2.3 `void XenIO::dumpSimilarity (boost::shared_ptr< Corpus > ptrCorp, boost::shared_ptr< Similarity > ptrSim)`
`[static]`

Dumps the [Similarity](#) measures of a [Corpus](#).

Parameters

<i>ptrCorp</i>	: the Corpus from which the Similarity measures are dumped
<i>ptrSim</i>	: the Similarity measures to dump

Here is the caller graph for this function:



6.27.2.4 `std::vector< std::string > XenIO::read (boost::shared_ptr< XenFile > ptrFile)` `[static]`

Reads a file (plain text/gzipped)

Parameters

<i>ptrFile</i>	: the file to read
----------------	--------------------

Returns

a vector of strings containing the read file's lines

Here is the call graph for this function:



Here is the caller graph for this function:



6.27.2.5 `boost::shared_ptr< EvalMap > XenIO::readDist (std::string distFile) [static]`

Reads a evaluation/best point distribution file.

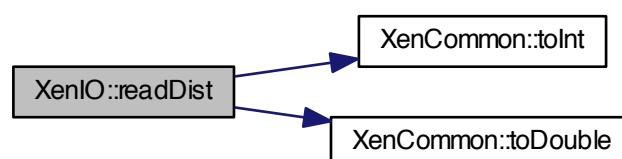
Parameters

<i>distFile</i>	: file to read
-----------------	----------------

Returns

an EvalMap containing the already computed scores

Here is the call graph for this function:



Here is the caller graph for this function:



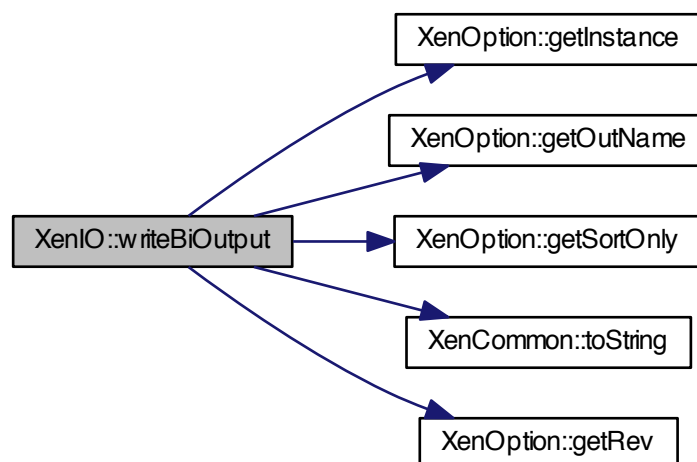
6.27.2.6 `void XenIO::writeBiOutput (boost::shared_ptr< Corpus > ptrCorpSource, boost::shared_ptr< Corpus > ptrCorpTarget, boost::shared_ptr< Score > ptrScore) [static]`

Writes bilingual scored/sorted result files.

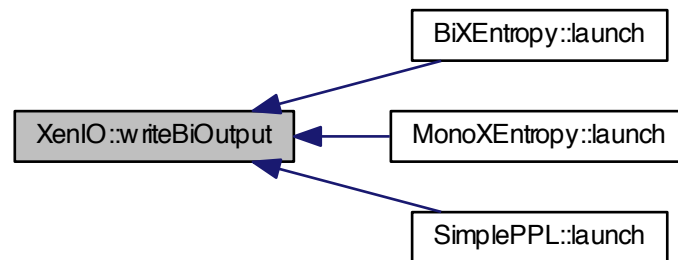
Parameters

<i>ptrCorpSource</i>	: the source language Corpus to write
<i>ptrCorpTarget</i>	: the target language Corpus to write
<i>ptrScore</i>	: the associated Score object to write

Here is the call graph for this function:



Here is the caller graph for this function:



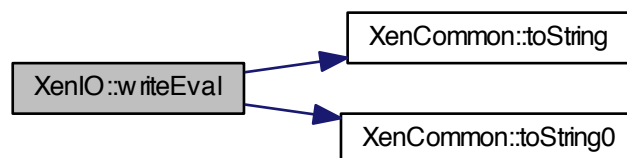
6.27.2.7 `void XenIO::writeEval (boost::shared_ptr< EvalMap > ptrEvalMap, std::string distName) [static]`

Writes an evaluation/best point distribution file.

Parameters

<i>ptrEvalMap</i>	: the EvalMap containing the scores to write
<i>distName</i>	: the distribution file name

Here is the call graph for this function:



Here is the caller graph for this function:



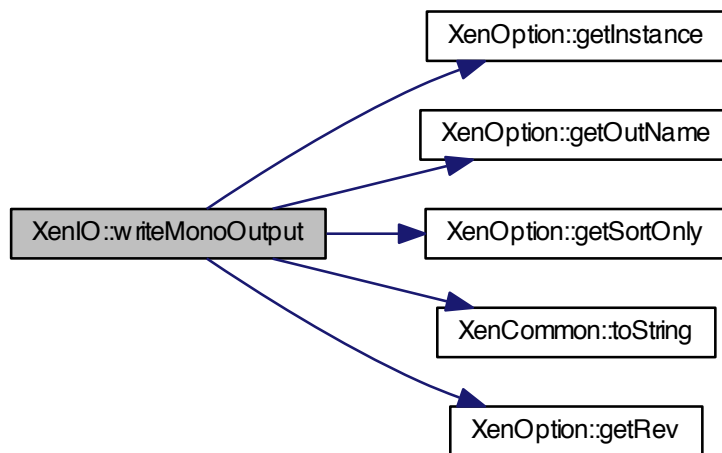
6.27.2.8 `void XenIO::writeMonoOutput (boost::shared_ptr< Corpus > ptrCorp, boost::shared_ptr< Score > ptrScore)`
`[static]`

Writes monolingual scored/sorted result files.

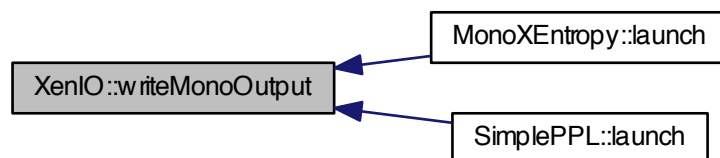
Parameters

<i>ptrCorp</i>	: the Corpus to write
<i>ptrScore</i>	: the associated Score object to write

Here is the call graph for this function:



Here is the caller graph for this function:



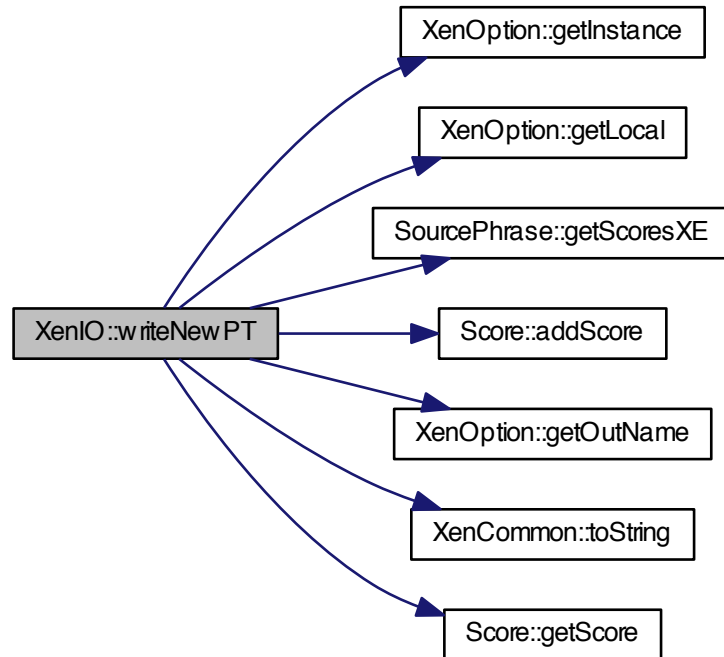
6.27.2.9 `void XenIO::writeNewPT (boost::shared_ptr< PhraseTable > ptrPT, boost::shared_ptr< Score > ptrScore)`
`[static]`

Writes a new rescored phrase-table.

Parameters

<i>ptrPT</i>	: the new phrase-table to write
<i>ptrScore</i>	: the associated Score object to write

Here is the call graph for this function:



Here is the caller graph for this function:



6.27.2.10 `std::string XenIO::writeSourcePhrases (boost::shared_ptr< PhraseTable > ptrPT) [static]`

Writes a phrase-table's source phrases.

Parameters

<i>ptrPT</i>	: the phrase-table to write the source phrases from
--------------	---

Returns

the written source phrases file name

Here is the caller graph for this function:



6.27.2.11 `std::string XenIO::writeTargetPhrases (boost::shared_ptr< PhraseTable > ptrPT) [static]`

Writes a phrase-table's target phrases.

Parameters

<i>ptrPT</i>	: the phrase-table to write the target phrases from
--------------	---

Returns

the written target phrases file name

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- include/utis/[xenio.h](#)
- src/utis/[xenio.cpp](#)

6.28 XenLMsri Class Reference

Class handling SRI LM estimation, loading, querying...

```
#include <XenLMsri.h>
```

Public Member Functions

- [XenLMsri](#) ()

Default constructor.

- void [initialize](#) (boost::shared_ptr< [Corpus](#) > ptrCorp, boost::shared_ptr< [XenVocab](#) > ptrVoc)

Initialization function from a [Corpus](#) and a vocabulary ([XenVocab](#))

- void [initialize](#) (boost::shared_ptr< [XenFile](#) > ptrFile, boost::shared_ptr< [XenVocab](#) > ptrVoc)

Initialization function from an already existing LM file and a vocabulary ([XenVocab](#))

- void [initialize](#) (boost::shared_ptr< [XenResult](#) > ptrXenRes, boost::shared_ptr< [XenVocab](#) > ptrVoc, int pc, std::string name="")

Initialization function from a [XenC](#) filtering result file and a vocabulary.

- [~XenLMsri](#) ()

Default destructor.

- int [createLM](#) ()

Estimates a language model based on the provided data.

- int [loadLM](#) ()

Loads a language model from a provided file name.

- int [writeLM](#) ()

Writes an (arpa or binary) estimated language model on disk.

- std::string [getFileName](#) () const

Accessor to the language model file name.

- TextStats [getSentenceStats](#) (std::string sent)

Computes the SRILM stats of a given sentence.

- TextStats [getDocumentStats](#) (boost::shared_ptr< [Corpus](#) > ptrCorp)

Computes the SRILM stats of a [Corpus](#) at a document level.

6.28.1 Detailed Description

Class handling SRI LM estimation, loading, querying...

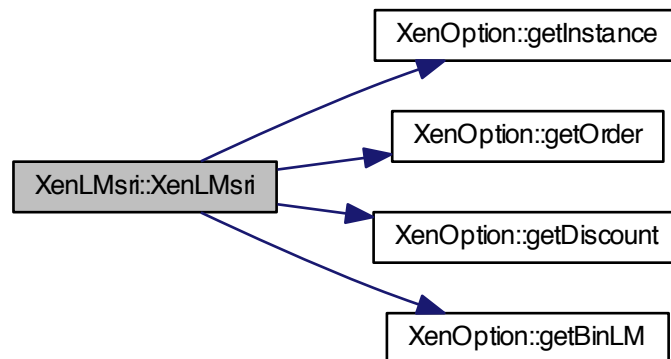
This class is in charge of handling all SRILM-related operations. Be aware that due to some memory leaks in SRILM, memory usage in [XenC](#) can grow up very fast.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 [XenLMsri::XenLMsri](#) ()

Default constructor.

Here is the call graph for this function:



6.28.2.2 `XenLMsri::~~XenLMsri ()`

Default destructor.

6.28.3 Member Function Documentation

6.28.3.1 `int XenLMsri::createLM ()`

Estimates a language model based on the provided data.

Returns

0 if all goes well

Here is the call graph for this function:



6.28.3.2 `TextStats XenLMsri::getDocumentStats (boost::shared_ptr< Corpus > ptrCorp)`

Computes the SRILM stats of a [Corpus](#) at a document level.

Parameters

<i>ptrCorp</i>	: the Corpus to compute the stats from
----------------	--

Returns

the computed document-level SRILM stats

Here is the call graph for this function:



6.28.3.3 `std::string XenLMsri::getFileName () const`

Accessor to the language model file name.

Returns

the language model file name

6.28.3.4 `TextStats XenLMsri::getSentenceStats (std::string sent)`

Computes the SRILM stats of a given sentence.

Parameters

<i>sent</i>	: the sentence to compute the stats from
-------------	--

Returns

the computed SRILM stats

Here is the caller graph for this function:



6.28.3.5 `void XenLMsri::initialize (boost::shared_ptr< Corpus > ptrCorp, boost::shared_ptr< XenVocab > ptrVoc)`

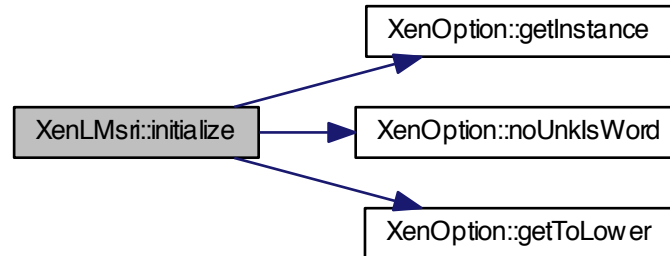
Initialization function from a [Corpus](#) and a vocabulary ([XenVocab](#))

Parameters

<i>ptrCorp</i>	: the corpus to estimate the LM from
<i>ptrVoc</i>	: the vocabulary used to estimate the LM

< No use for a [XenResult](#) here

Here is the call graph for this function:



6.28.3.6 `void XenLMsri::initialize (boost::shared_ptr< XenFile > ptrFile, boost::shared_ptr< XenVocab > ptrVoc)`

Initialization function from an already existing LM file and a vocabulary ([XenVocab](#))

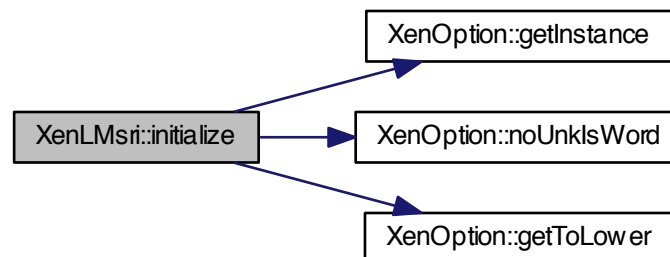
Parameters

<i>ptrFile</i>	: the LM file to load
<i>ptrVoc</i>	: the vocabulary used to estimate the LM

< No use for a corpus here

< No use for a [XenResult](#) here

Here is the call graph for this function:



6.28.3.7 `void XenLMsri::initialize (boost::shared_ptr< XenResult > ptrXenRes, boost::shared_ptr< XenVocab > ptrVoc, int pc, std::string name = " ")`

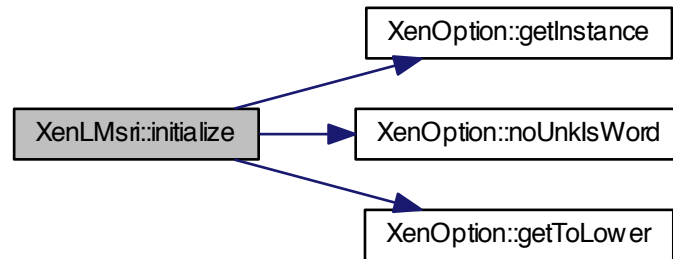
Initialization function from a XenC filtering result file and a vocabulary.

Parameters

<i>ptrXenRes</i>	: the XenC filtering result file
<i>ptrVoc</i>	: the vocabulary used to estimate the LM
<i>pc</i>	: the percentage of the corpus in ptrXenRes to use
<i>name</i>	: optionnal file name of the LM

< No use for a corpus here

Here is the call graph for this function:



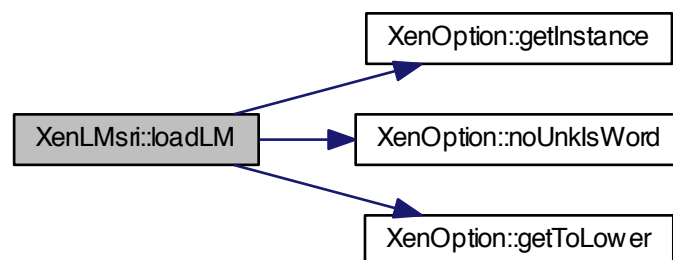
6.28.3.8 int XenLMsri::loadLM ()

Loads a language model from a provided file name.

Returns

0 if all goes well

Here is the call graph for this function:



6.28.3.9 int XenLMsri::writeLM ()

Writes an (arpa or binary) estimated language model on disk.

Returns

0 if all goes well

The documentation for this class was generated from the following files:

- include/XenLMsri.h
- src/XenLMsri.cpp

6.29 XenOption Class Reference

Singleton class handling XenC options accessors/mutators.

```
#include <xenoption.h>
```

Public Member Functions

- std::string [getSLang](#) () const
Accessor to the source language.
- std::string [getTLang](#) () const
Accessor to the target language.
- boost::shared_ptr< [XenFile](#) > [getInSData](#) () const
Accessor to the source language in-domain data file.
- boost::shared_ptr< [XenFile](#) > [getOutSData](#) () const
Accessor to the source language out-of-domain data file.
- boost::shared_ptr< [XenFile](#) > [getInTData](#) () const
Accessor to the target language in-domain data file.
- boost::shared_ptr< [XenFile](#) > [getOutTData](#) () const
Accessor to the target language out-of-domain data file.
- boost::shared_ptr< [XenFile](#) > [getInSStem](#) () const
Accessor to the source language in-domain stem data file.
- boost::shared_ptr< [XenFile](#) > [getOutSStem](#) () const
Accessor to the source language out-of-domain stem data file.
- boost::shared_ptr< [XenFile](#) > [getInTStem](#) () const
Accessor to the target language in-domain stem data file.
- boost::shared_ptr< [XenFile](#) > [getOutTStem](#) () const
Accessor to the target language out-of-domain stem data file.
- boost::shared_ptr< [XenFile](#) > [getInPTable](#) () const
Accessor to the in-domain phrase-table file.
- boost::shared_ptr< [XenFile](#) > [getOutPTable](#) () const
Accessor to the out-of-domain phrase-table file.
- bool [getMono](#) () const
Accessor to the monolingual or bilingual execution state.
- int [getMode](#) () const
Accessor to the filtering mode.
- bool [getMean](#) () const
Accessor to the mean execution state.
- bool [getSim](#) () const
Accessor to the similarity measures execution state.
- bool [getSimOnly](#) () const
Accessor to the similarity measures ONLY execution state.

- int [getVecSize](#) () const
Accessor to the similarity measures vector size.
- boost::shared_ptr< [XenFile](#) > [getSVocab](#) () const
Accessor to the source language vocabulary file.
- boost::shared_ptr< [XenFile](#) > [getTVocab](#) () const
Accessor to the target language vocabulary file.
- bool [getFullVocab](#) () const
Accessor to the global vocabulary execution state.
- boost::shared_ptr< [XenFile](#) > [getInSLM](#) () const
Accessor to the source language in-domain language model file.
- boost::shared_ptr< [XenFile](#) > [getOutSLM](#) () const
Accessor to the source language out-of-domain language model file.
- boost::shared_ptr< [XenFile](#) > [getInTLM](#) () const
Accessor to the target language in-domain language model file.
- boost::shared_ptr< [XenFile](#) > [getOutTLM](#) () const
Accessor to the target language out-of-domain language model file.
- boost::shared_ptr< [XenFile](#) > [getWFile](#) () const
Accessor to the weights file.
- boost::shared_ptr< [XenFile](#) > [getDev](#) () const
Accessor to the development corpus file.
- int [getOrder](#) () const
Accessor to the order for language models estimation.
- int [getDiscount](#) () const
Accessor to the discounting method for language models estimation.
- bool [getToLower](#) () const
Accessor to the flag for lower case vocabulary mapping in LM estimation.
- bool [noUnkIsWord](#) () const
Accessor to the flag for NOT considering <unk> as a word in LM estimation.
- int [getBinLM](#) () const
Accessor to the estimated LMs output format.
- int [getSampleSize](#) () const
Accessor to the out-of-domain [Corpus](#) current sample size.
- bool [getLog](#) () const
Accessor to the log-domain state of values in the weights file.
- bool [getRev](#) () const
Accessor to the reversed filtered output state.
- bool [getInv](#) () const
Accessor to the inverted filtered output state.
- bool [getStem](#) () const
Accessor to the stem mode execution state.
- bool [getLocal](#) () const
Accessor to the local score for phrase-table scoring execution state.
- bool [getEval](#) () const
Accessor to the evaluation execution state.
- bool [getBp](#) () const
Accessor to the best point execution state.
- int [getStep](#) () const
Accessor to the step size for evaluation/best point.
- std::string [getOutName](#) () const
Accessor to the output name for the filtered files.
- std::string [getName](#) () const

Accessor to the program name.

- int [getThreads](#) () const

Accessor to the requested number of threads.

- bool [getSortOnly](#) () const

Accessor to whether we output the scored file.

- void [setSampleSize](#) (int size)

Mutator to the out-of-domain sample size.

- void [setStep](#) (int step)

Mutator to the evaluation/best point step size.

Static Public Member Functions

- static [XenOption](#) * [getInstance](#) ()

Accessor to the instance of the singleton [XenOption](#) object.

- static [XenOption](#) * [getInstance](#) (LPOptions opt)

Accessor to the instance of the singleton [XenOption](#) object.

- static void [deleteInstance](#) ()

Deletes the unique instance of the [XenOption](#) singleton.

6.29.1 Detailed Description

Singleton class handling XenC options accessors/mutators.

6.29.2 Member Function Documentation

6.29.2.1 void [XenOption::deleteInstance](#) () [static]

Deletes the unique instance of the [XenOption](#) singleton.

Here is the caller graph for this function:



6.29.2.2 int [XenOption::getBinLM](#) () const

Accessor to the estimated LMs output format.

Returns

the estimated LMs output format

Here is the caller graph for this function:

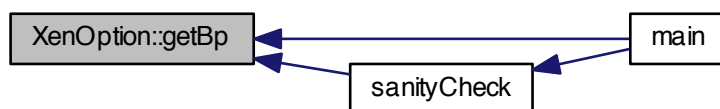
**6.29.2.3 bool XenOption::getBp () const**

Accessor to the best point execution state.

Returns

true if we have to perform a best point search

Here is the caller graph for this function:

**6.29.2.4 boost::shared_ptr< XenFile > XenOption::getDev () const**

Accessor to the development corpus file.

Returns

the development corpus file

Here is the caller graph for this function:



6.29.2.5 `int XenOption::getDiscount () const`

Accessor to the discounting method for language models estimation.

Returns

the discounting method for language models estimation

Here is the caller graph for this function:



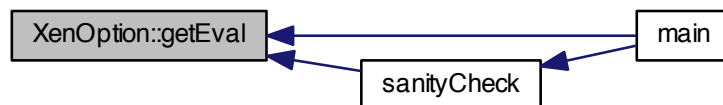
6.29.2.6 `bool XenOption::getEval () const`

Accessor to the evaluation execution state.

Returns

true if we have to perform an evaluation

Here is the caller graph for this function:



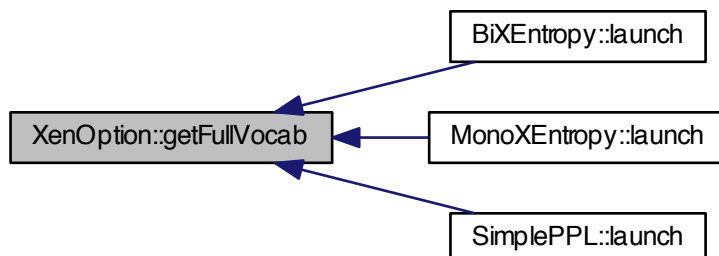
6.29.2.7 `bool XenOption::getFullVocab () const`

Accessor to the global vocabulary execution state.

Returns

true if we use a global vocabulary instead of only in-domain

Here is the caller graph for this function:

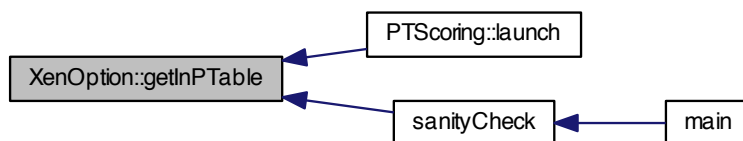
**6.29.2.8 boost::shared_ptr< XenFile > XenOption::getInPTable () const**

Accessor to the in-domain phrase-table file.

Returns

the in-domain phrase-table file

Here is the caller graph for this function:

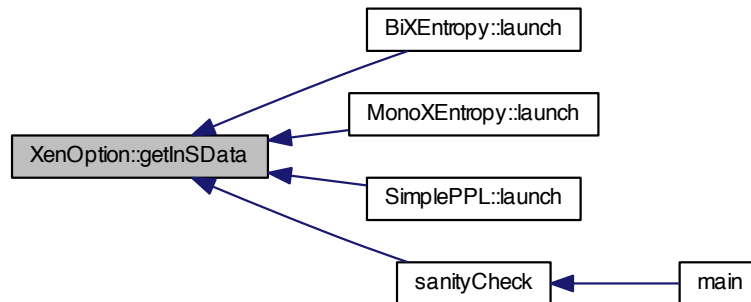
**6.29.2.9 boost::shared_ptr< XenFile > XenOption::getInSData () const**

Accessor to the source language in-domain data file.

Returns

the source language in-domain data file

Here is the caller graph for this function:



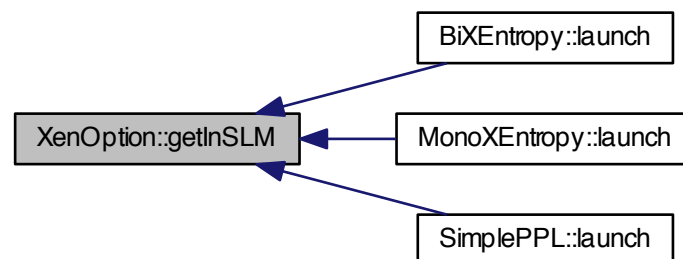
6.29.2.10 `boost::shared_ptr< XenFile > XenOption::getInSLM () const`

Accessor to the source language in-domain language model file.

Returns

the source language in-domain language model file

Here is the caller graph for this function:



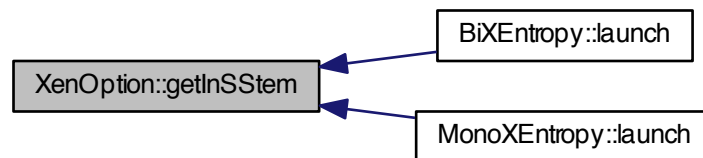
6.29.2.11 `boost::shared_ptr< XenFile > XenOption::getInSStem () const`

Accessor to the source language in-domain stem data file.

Returns

the source language in-domain stem data file

Here is the caller graph for this function:

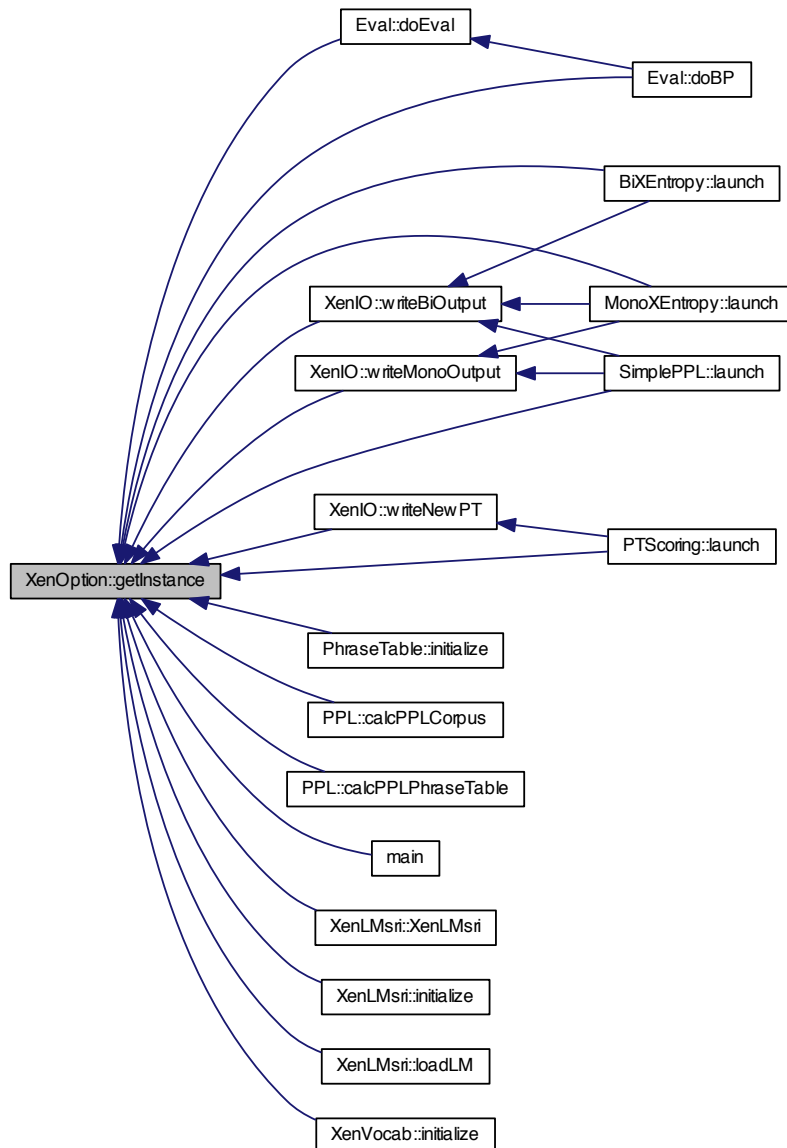
**6.29.2.12 XenOption * XenOption::getInstance () [static]**

Accessor to the instance of the singleton [XenOption](#) object.

Returns

the [XenOption](#) unique instance

Here is the caller graph for this function:



6.29.2.13 `XenOption * XenOption::getInstance (LOptions opt) [static]`

Accessor to the instance of the singleton [XenOption](#) object.

Parameters

<code>opt</code>	: the LOptions struct to build the XenOption object from
------------------	--

Returns

the [XenOption](#) unique instance

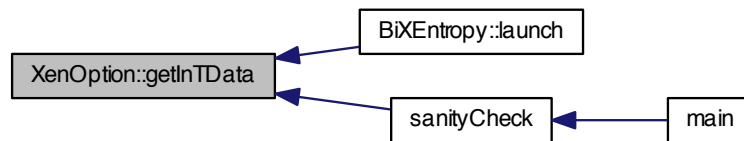
6.29.2.14 `boost::shared_ptr< XenFile > XenOption::getInTData () const`

Accessor to the target language in-domain data file.

Returns

the target language in-domain data file

Here is the caller graph for this function:



6.29.2.15 `boost::shared_ptr< XenFile > XenOption::getInTLM () const`

Accessor to the target language in-domain language model file.

Returns

the target language in-domain language model file

Here is the caller graph for this function:



6.29.2.16 `boost::shared_ptr< XenFile > XenOption::getInTStem () const`

Accessor to the target language in-domain stem data file.

Returns

the target language in-domain stem data file

Here is the caller graph for this function:

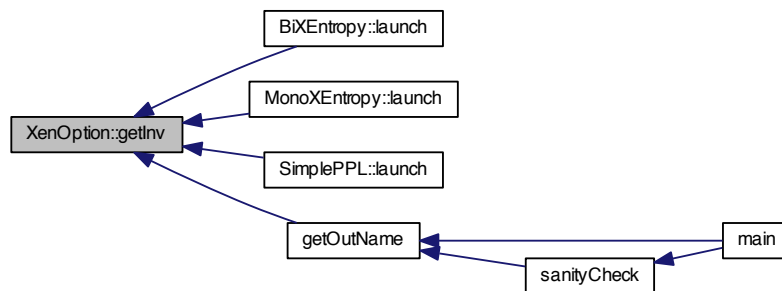
**6.29.2.17 bool XenOption::getInv () const**

Accessor to the inverted filtered output state.

Returns

true if we output inverted scores (1 - score)

Here is the caller graph for this function:

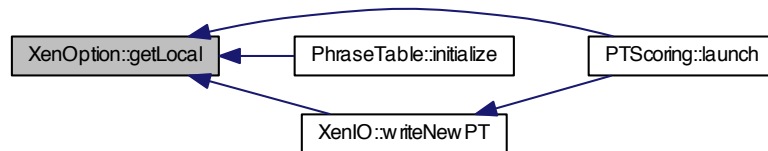
**6.29.2.18 bool XenOption::getLocal () const**

Accessor to the local score for phrase-table scoring execution state.

Returns

true if we also compute local scores in phrase-table scoring mode

Here is the caller graph for this function:

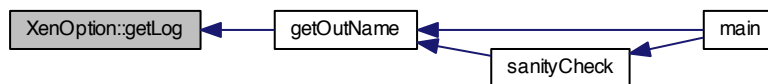
**6.29.2.19 bool XenOption::getLog () const**

Accessor to the log-domain state of values in the weights file.

Returns

true if values in the weights file are in the log-domain

Here is the caller graph for this function:

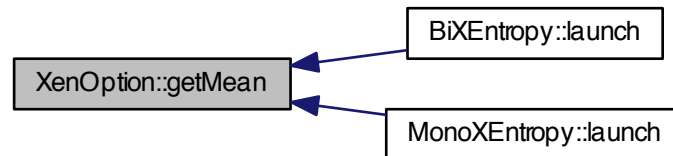
**6.29.2.20 bool XenOption::getMean () const**

Accessor to the mean execution state.

Returns

true if we compute out-of-domain scores with mean of 3 LMs

Here is the caller graph for this function:



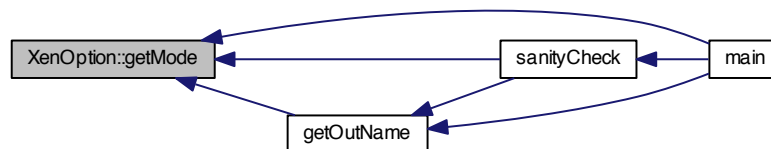
6.29.2.21 int XenOption::getMode () const

Accessor to the filtering mode.

Returns

the filtering mode

Here is the caller graph for this function:



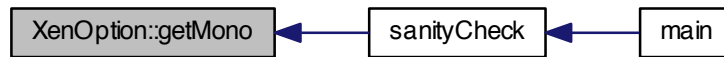
6.29.2.22 bool XenOption::getMono () const

Accessor to the monolingual or bilingual execution state.

Returns

true if we work on monolingual data

Here is the caller graph for this function:

**6.29.2.23** `std::string XenOption::getName () const`

Accessor to the program name.

Returns

the program name

6.29.2.24 `int XenOption::getOrder () const`

Accessor to the order for language models estimation.

Returns

the order for language models estimation

Here is the caller graph for this function:

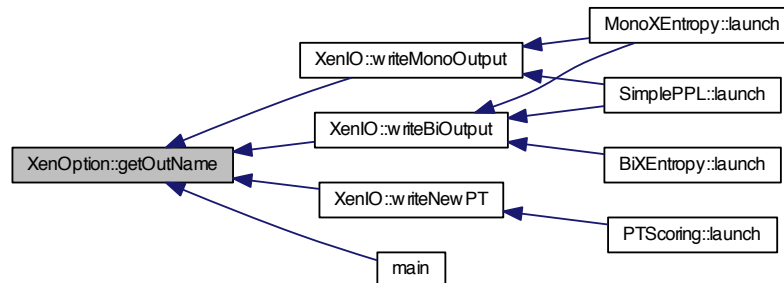
**6.29.2.25** `std::string XenOption::getOutName () const`

Accessor to the output name for the filtered files.

Returns

the output name for the filtered files

Here is the caller graph for this function:



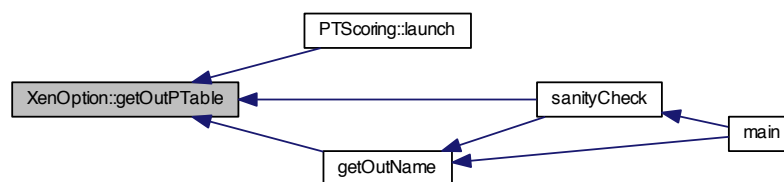
6.29.2.26 `boost::shared_ptr< XenFile > XenOption::getOutPTTable () const`

Accessor to the out-of-domain phrase-table file.

Returns

the out-of-domain phrase-table file

Here is the caller graph for this function:



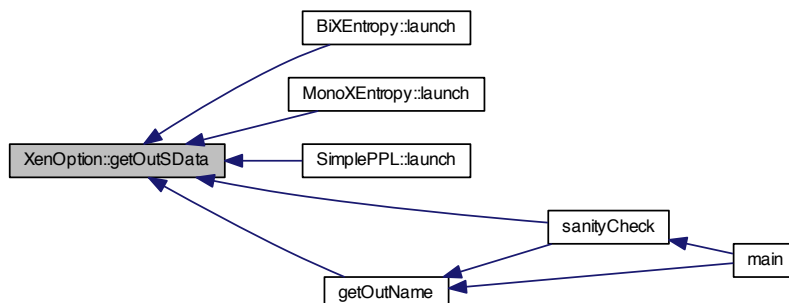
6.29.2.27 `boost::shared_ptr< XenFile > XenOption::getOutSData () const`

Accessor to the source language out-of-domain data file.

Returns

the source language out-of-domain data file

Here is the caller graph for this function:



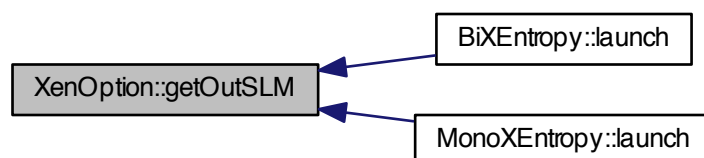
6.29.2.28 `boost::shared_ptr< XenFile > XenOption::getOutSLM () const`

Accessor to the source language out-of-domain language model file.

Returns

the source language out-of-domain language model file

Here is the caller graph for this function:



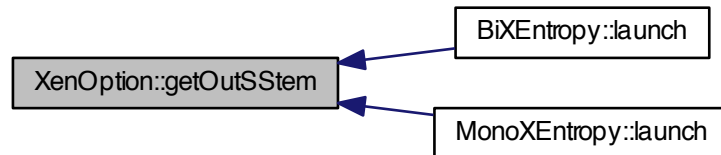
6.29.2.29 `boost::shared_ptr< XenFile > XenOption::getOutSStem () const`

Accessor to the source language out-of-domain stem data file.

Returns

the source language out-of-domain stem data file

Here is the caller graph for this function:



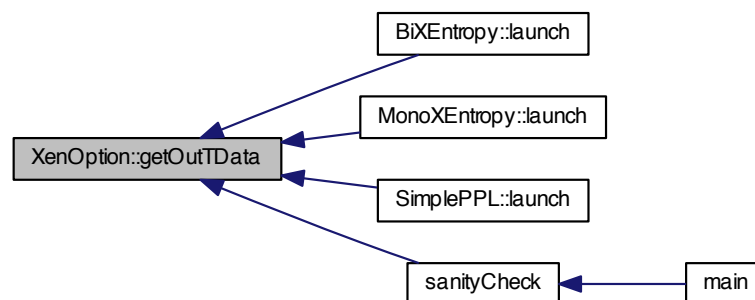
6.29.2.30 `boost::shared_ptr< XenFile > XenOption::getOutTData () const`

Accessor to the target language out-of-domain data file.

Returns

the target language out-of-domain data file

Here is the caller graph for this function:



6.29.2.31 `boost::shared_ptr< XenFile > XenOption::getOutTLM () const`

Accessor to the target language out-of-domain language model file.

Returns

the target language out-of-domain language model file

Here is the caller graph for this function:

**6.29.2.32** `boost::shared_ptr< XenFile > XenOption::getOutTStem () const`

Accessor to the target language out-of-domain stem data file.

Returns

the target language out-of-domain stem data file

Here is the caller graph for this function:

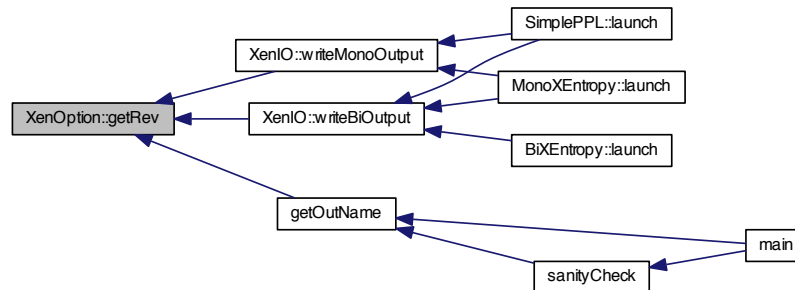
**6.29.2.33** `bool XenOption::getRev () const`

Accessor to the reversed filtered output state.

Returns

true if we output a descending order filtered file

Here is the caller graph for this function:



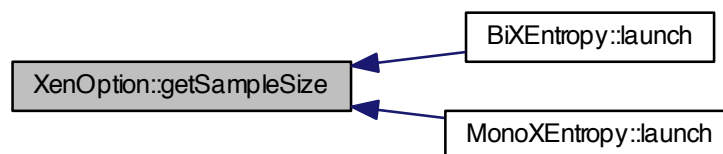
6.29.2.34 int XenOption::getSampleSize () const

Accessor to the out-of-domain [Corpus](#) current sample size.

Returns

the out-of-domain [Corpus](#) current sample size

Here is the caller graph for this function:



6.29.2.35 bool XenOption::getSim () const

Accessor to the similarity measures execution state.

Returns

true if we compute similarity measures

Here is the caller graph for this function:

**6.29.2.36 bool XenOption::getSimOnly () const**

Accessor to the similarity measures ONLY execution state.

Returns

true if we compute similarity measures ONLY

Here is the caller graph for this function:

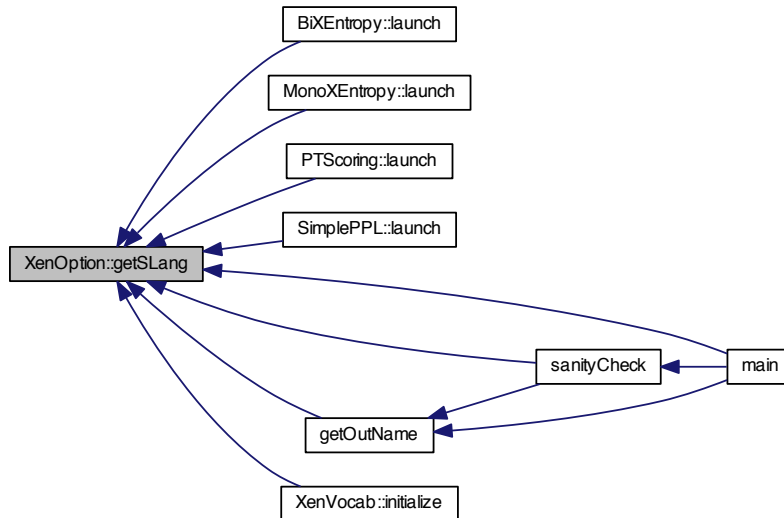
**6.29.2.37 std::string XenOption::getSLang () const**

Accessor to the source language.

Returns

the source language

Here is the caller graph for this function:

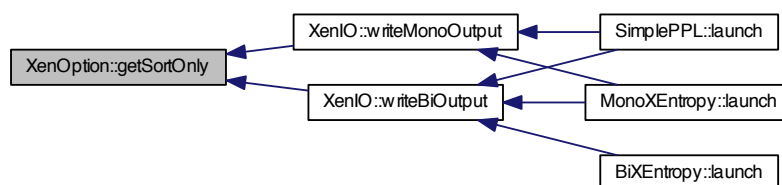
**6.29.2.38 bool XenOption::getSortOnly () const**

Accessor to whether we output the scored file.

Returns

true if we only need to output the sorted file

Here is the caller graph for this function:

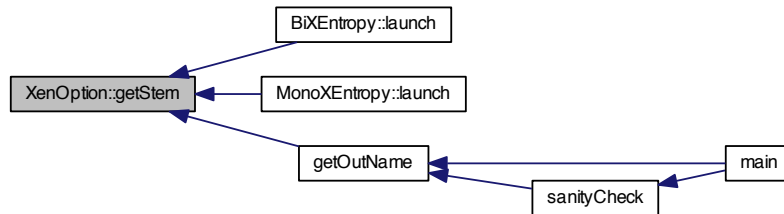
**6.29.2.39 bool XenOption::getStem () const**

Accessor to the stem mode execution state.

Returns

true if we work with stem [Corpus](#) too

Here is the caller graph for this function:



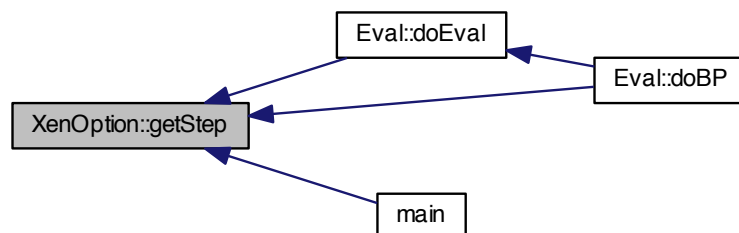
6.29.2.40 int XenOption::getStep () const

Accessor to the step size for evaluation/best point.

Returns

the step size for evaluation/best point

Here is the caller graph for this function:



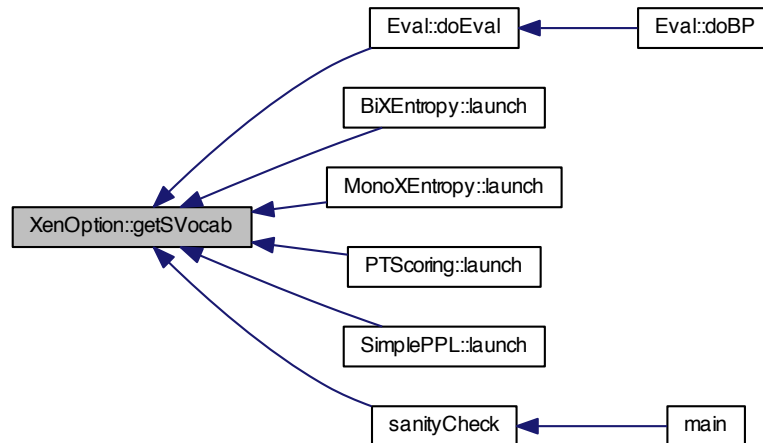
6.29.2.41 boost::shared_ptr< XenFile > XenOption::getSVocab () const

Accessor to the source language vocabulary file.

Returns

the source language vocabulary file

Here is the caller graph for this function:



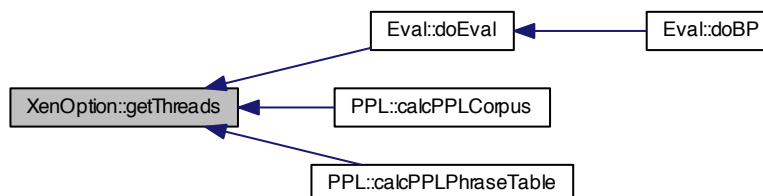
6.29.2.42 `int XenOption::getThreads () const`

Accessor to the requested number of threads.

Returns

the requested number of threads

Here is the caller graph for this function:



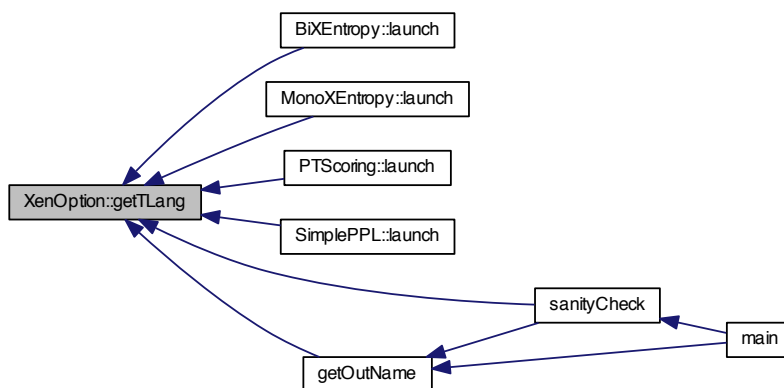
6.29.2.43 `std::string XenOption::getTLang () const`

Accessor to the target language.

Returns

the target language

Here is the caller graph for this function:

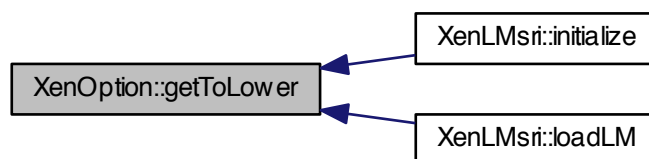
6.29.2.44 `bool XenOption::getToLower () const`

Accessor to the flag for lower case vocabulary mapping in LM estimation.

Returns

true if lower case vocabulary mapping is requested

Here is the caller graph for this function:

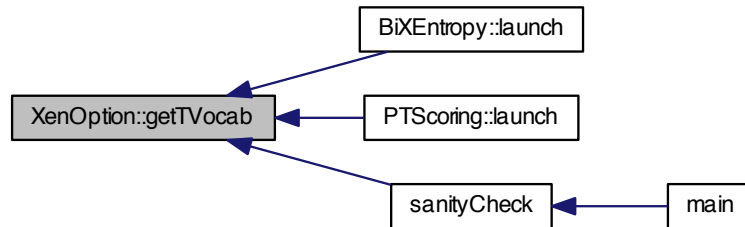
6.29.2.45 `boost::shared_ptr< XenFile > XenOption::getTVocab () const`

Accessor to the target language vocabulary file.

Returns

the target language vocabulary file

Here is the caller graph for this function:



6.29.2.46 `int XenOption::getVecSize () const`

Accessor to the similarity measures vector size.

Returns

the similarity measures vector size

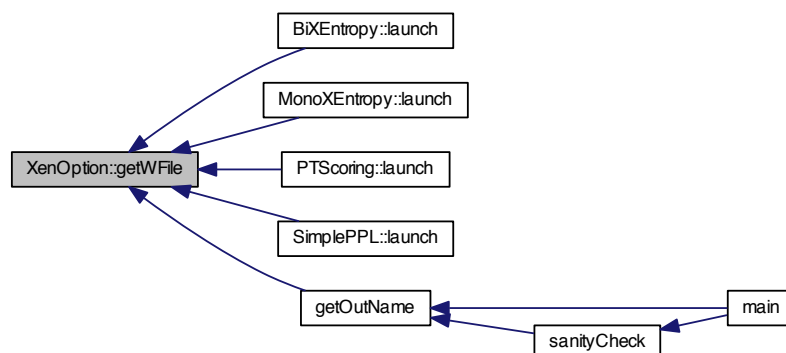
6.29.2.47 `boost::shared_ptr< XenFile > XenOption::getWFile () const`

Accessor to the weights file.

Returns

the weights file

Here is the caller graph for this function:



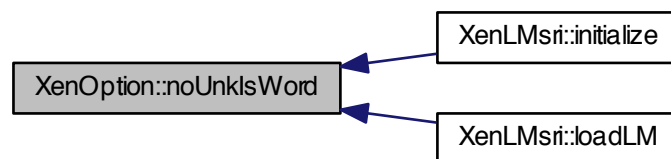
6.29.2.48 bool XenOption::noUnklsWord () const

Accessor to the flag for NOT considering <unk> as a word in LM estimation.

Returns

true if <unk> is NOT considered as a word

Here is the caller graph for this function:



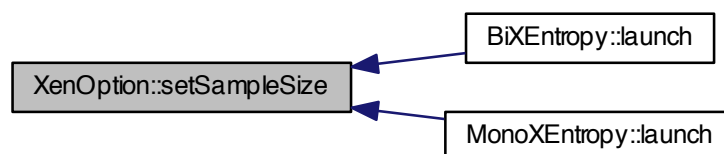
6.29.2.49 void XenOption::setSampleSize (int size)

Mutator to the out-of-domain sample size.

Parameters

<i>size</i>	: the out-of-domain sample size
-------------	---------------------------------

Here is the caller graph for this function:



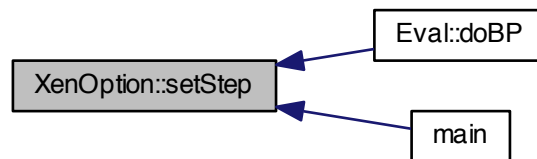
6.29.2.50 void XenOption::setStep (int step)

Mutator to the evaluation/best point step size.

Parameters

<i>step</i>	: the evaluation/best point step size
-------------	---------------------------------------

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- [include/xenoption.h](#)
- [src/xenoption.cpp](#)

6.30 XenResult Class Reference

Class handling a XenC sorted result file for evaluation/best point.

```
#include <xenresult.h>
```

Public Member Functions

- [XenResult](#) ()
Default constructor.
- void [initialize](#) (boost::shared_ptr< [XenFile](#) > ptrFile)
Initialization function from an already instantiated [XenFile](#).
- [~XenResult](#) ()
Default destructor.
- std::vector< std::string > [getSortedText](#) () const
Accessor to the sorted corpus text.
- std::string [getTextLine](#) (int n)
*Accessor to the *n*th line of the sorted corpus text.*
- unsigned int [getSize](#) () const
Accessor to the size of the sorted corpus text.
- boost::shared_ptr< [XenFile](#) > [getXenFile](#) () const
Accessor to the sorted result file.

6.30.1 Detailed Description

Class handling a XenC sorted result file for evaluation/best point.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 [XenResult::XenResult](#) ()

Default constructor.

6.30.2.2 `XenResult::~~XenResult ()`

Default destructor.

6.30.3 Member Function Documentation

6.30.3.1 `unsigned int XenResult::getSize () const`

Accessor to the size of the sorted corpus text.

Returns

the size of the sorted corpus text

6.30.3.2 `std::vector< std::string > XenResult::getSortedText () const`

Accessor to the sorted corpus text.

Returns

the sorted corpus text

6.30.3.3 `std::string XenResult::getTextLine (int n)`

Accessor to the *n*th line of the sorted corpus text.

Parameters

<i>n</i>	: the number of the text line to get
----------	--------------------------------------

Returns

the requested *n*th line of text

6.30.3.4 `boost::shared_ptr< XenFile > XenResult::getXenFile () const`

Accessor to the sorted result file.

Returns

the sorted result file

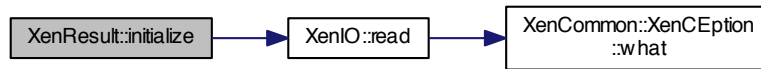
6.30.3.5 `void XenResult::initialize (boost::shared_ptr< XenFile > ptrFile)`

Initialization function from an already instantiated [XenFile](#).

Parameters

<i>ptrFile</i>	: the sorted result file
----------------	--------------------------

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- [include/xenresult.h](#)
- [src/xenresult.cpp](#)

6.31 XenVocab Class Reference

Class handling a XenC vocabulary.

```
#include <xenvocab.h>
```

Public Member Functions

- [XenVocab](#) ()
Default constructor.
- void [initialize](#) (boost::shared_ptr< [XenFile](#) > ptrFile)
Initialization function from an already instantiated [XenFile](#).
- void [initialize](#) (boost::shared_ptr< [Corpus](#) > ptrCorp)
Initialization function from a [Corpus](#).
- void [initialize](#) (boost::shared_ptr< [Corpus](#) > ptrInCorp, boost::shared_ptr< [Corpus](#) > ptrOutCorp)
Initialization function from two [Corpus](#).
- void [initialize](#) (boost::shared_ptr< [XenResult](#) > ptrXenRes)
Initialization function from a sorted result file.
- [~XenVocab](#) ()
Default destructor.
- boost::shared_ptr< Vocab > [getVocab](#) () const
Accessor to the SRILM Vocab object.
- std::map< std::string, int > [getXenVocab](#) () const
Accessor to the XenC vocabulary object.
- boost::shared_ptr< [XenFile](#) > [getXenFile](#) () const
Accessor to the vocabulary file.
- unsigned int [getSize](#) () const
Accessor to the size of the vocabulary text.

6.31.1 Detailed Description

Class handling a XenC vocabulary.

6.31.2 Constructor & Destructor Documentation

6.31.2.1 `XenVocab::XenVocab ()`

Default constructor.

6.31.2.2 `XenVocab::~~XenVocab ()`

Default destructor.

6.31.3 Member Function Documentation

6.31.3.1 `unsigned int XenVocab::getSize () const`

Accessor to the size of the vocabulary text.

Returns

the size of the vocabulary text

6.31.3.2 `boost::shared_ptr< Vocab > XenVocab::getVocab () const`

Accessor to the SRILM Vocab object.

Returns

the SRILM Vocab object

6.31.3.3 `boost::shared_ptr< XenFile > XenVocab::getXenFile () const`

Accessor to the vocabulary file.

Returns

the vocabulary file

6.31.3.4 `std::map< std::string, int > XenVocab::getXenVocab () const`

Accessor to the XenC vocabulary object.

Returns

the XenC vocabulary object

6.31.3.5 `void XenVocab::initialize (boost::shared_ptr< XenFile > ptrFile)`

Initialization function from an already instantiated [XenFile](#).

Parameters

<i>ptrFile</i>	: the vocabulary file
----------------	-----------------------

6.31.3.6 void `XenVocab::initialize` (boost::shared_ptr< `Corpus` > *ptrCorp*)

Initialization function from a [Corpus](#).

Parameters

<i>ptrCorp</i>	: the Corpus to extract the vocabulary from
----------------	---

6.31.3.7 void `XenVocab::initialize` (boost::shared_ptr< `Corpus` > *ptrInCorp*, boost::shared_ptr< `Corpus` > *ptrOutCorp*)

Initialization function from two [Corpus](#).

Parameters

<i>ptrInCorp</i>	: the in-domain Corpus to extract the vocabulary from
<i>ptrOutCorp</i>	: the out-of-domain Corpus to extract the vocabulary from

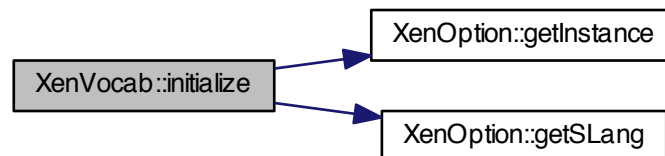
6.31.3.8 void `XenVocab::initialize` (boost::shared_ptr< `XenResult` > *ptrXenRes*)

Initialization function from a sorted result file.

Parameters

<i>ptrXenRes</i>	: the sorted result file to extract the vocabulary from
------------------	---

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- include/[xenvocab.h](#)
- src/[xenvocab.cpp](#)

File Documentation

Class handling corpus-related functionalities.

Include dependency graph for corpus.h:



Classes

- class [Corpus](#)
Corpus-related functionalities.

7.1.1 Detailed Description

Class handling corpus-related functionalities.

Author

Anthony Rousseau

Version

1.2.0

Date

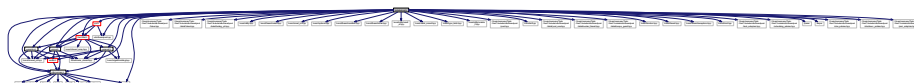
19 August 2013

7.2 include/eval.h File Reference

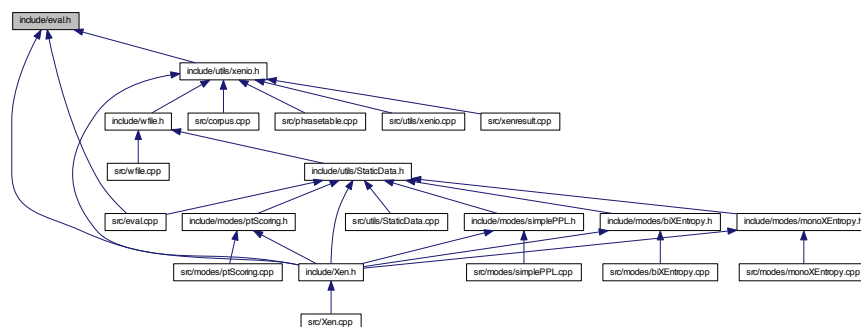
Class handling evaluation system.

```
#include <boost/shared_ptr.hpp>
#include <boost/make_shared.hpp>
#include "utils/common.h"
#include "utils/threadpool.hpp"
#include "corpus.h"
#include "xenoption.h"
#include "XenLMsri.h"
#include "ppl.h"
#include "xenresult.h"
```

Include dependency graph for eval.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Eval](#)
Evaluation system.

Typedefs

- typedef std::map< int, double, std::greater< int > > [EvalMap](#)
descending ordered map on integers as keys and doubles as values

Functions

- void [taskEval](#) (int pc, boost::shared_ptr< [XenResult](#) > ptrXR, boost::shared_ptr< [XenVocab](#) > ptrVoc, boost::shared_ptr< [Corpus](#) > ptrDevCorp, boost::shared_ptr< [EvalMap](#) > ptrDist)
Thread-safe evaluation function.

7.2.1 Detailed Description

Class handling evaluation system.

Author

Anthony Rousseau

Version

1.2.0

Date

19 August 2013

7.2.2 Typedef Documentation

7.2.2.1 typedef std::map<int, double, std::greater<int> > [EvalMap](#)

descending ordered map on integers as keys and doubles as values

7.2.3 Function Documentation

7.2.3.1 void [taskEval](#) (int pc, boost::shared_ptr< [XenResult](#) > ptrXR, boost::shared_ptr< [XenVocab](#) > ptrVoc, boost::shared_ptr< [Corpus](#) > ptrDevCorp, boost::shared_ptr< [EvalMap](#) > ptrDist)

Thread-safe evaluation function.

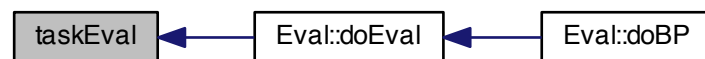
Parameters

<i>pc</i>	: integer representing the percentage of the scored out-of-domain corpus to take
<i>ptrXR</i>	: shared pointer on the XenResult object representing the selection result file
<i>ptrVoc</i>	: shared pointer on the XenVocab object representing the vocabulary to use for eval
<i>ptrDevCorp</i>	: shared pointer on the Corpus object representing the development set
<i>ptrDist</i>	: shared pointer on the EvalMap type containing the evaluation scores

Here is the call graph for this function:



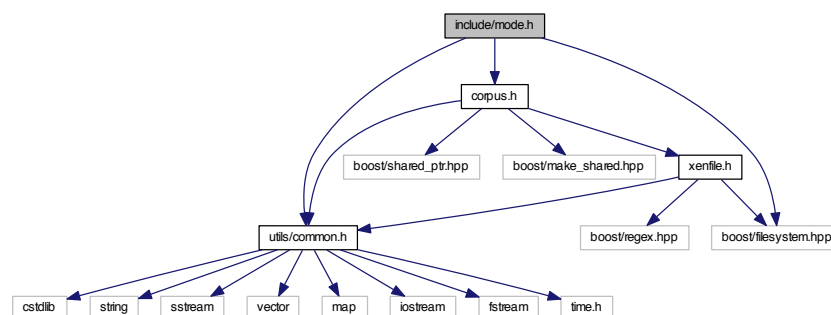
Here is the caller graph for this function:



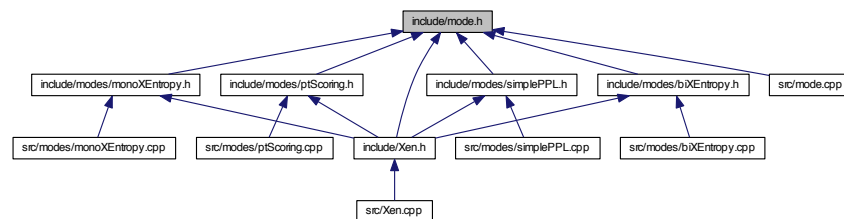
7.3 include/mode.h File Reference

Abstract class defining the filtering modes architecture.

```
#include "corpus.h"
#include "utils/common.h"
#include <boost/filesystem.hpp>
Include dependency graph for mode.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Mode](#)

Filtering modes interface.

7.3.1 Detailed Description

Abstract class defining the filtering modes architecture.

Author

Anthony Rousseau

Version

1.2.0

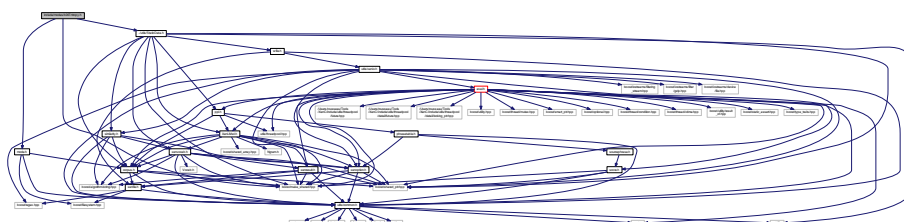
Date

19 August 2013

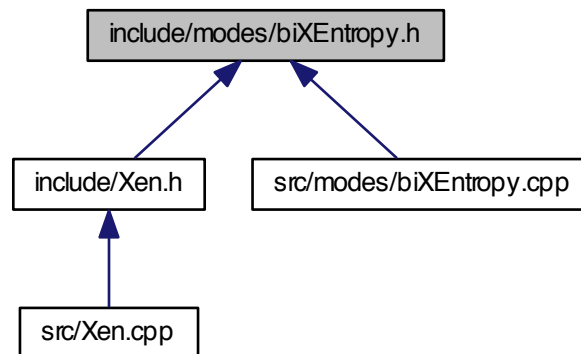
7.4 include/modes/biXEntropy.h File Reference

Derived class to handle filtering mode 3: bilingual cross-entropy.

```
#include <boost/make_shared.hpp>
#include "mode.h"
#include "../utils/StaticData.h"
Include dependency graph for biXEntropy.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [BiXEntropy](#)

Filtering mode 3: bilingual cross-entropy.

7.4.1 Detailed Description

Derived class to handle filtering mode 3: bilingual cross-entropy.

Author

Anthony Rousseau

Version

1.2.0

Date

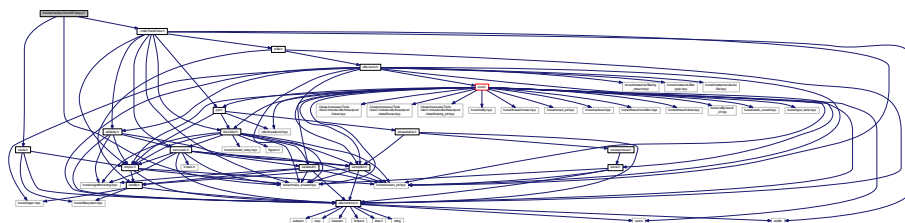
19 August 2013

7.5 include/modes/monoXEntropy.h File Reference

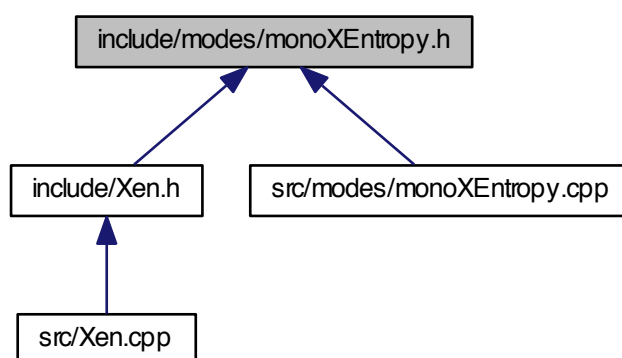
Derived class to handle filtering mode 2: monolingual cross-entropy.

```
#include <boost/make_shared.hpp>
#include "mode.h"
#include "../utils/StaticData.h"
```


Include dependency graph for monoXEntropy.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [MonoXEntropy](#)
Filtering mode 2: monolingual cross-entropy.

7.5.1 Detailed Description

Derived class to handle filtering mode 2: monolingual cross-entropy.

Author

Anthony Rousseau

Version

1.2.0

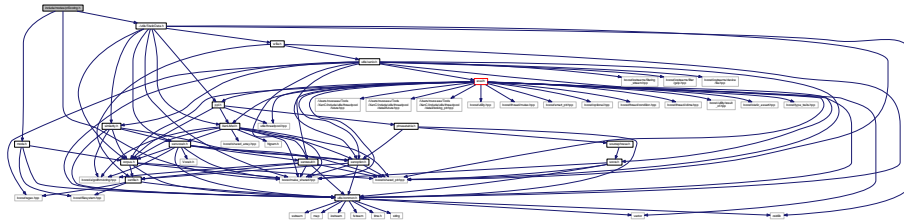
Date

19 August 2013

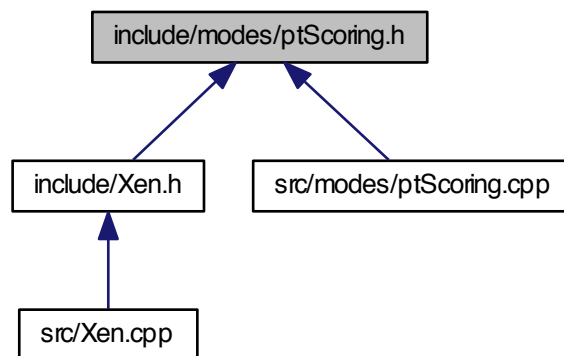
7.6 include/modes/ptScoring.h File Reference

Derived class to handle filtering mode 4: phrase-table cross-entropy.

```
#include <boost/make_shared.hpp>
#include "mode.h"
#include "../utils/StaticData.h"
Include dependency graph for ptScoring.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [PTScoring](#)
Filtering mode 4: phrase-table cross-entropy.

7.6.1 Detailed Description

Derived class to handle filtering mode 4: phrase-table cross-entropy.

Author

Anthony Rousseau

Version

1.2.0

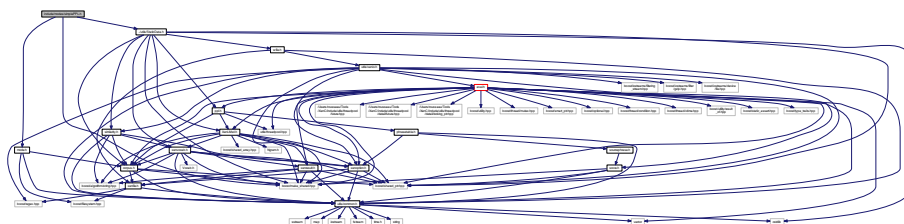
Date

19 August 2013

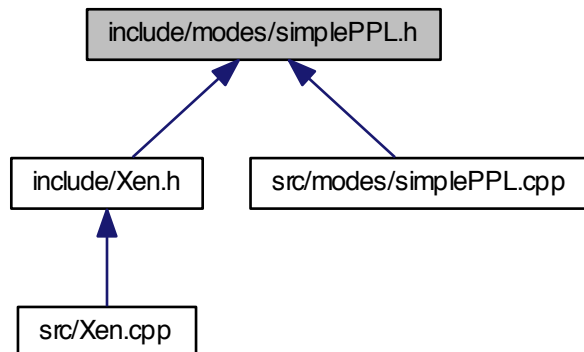
7.7 include/modes/simplePPL.h File Reference

Derived class to handle filtering mode 1: simple perplexity.

```
#include <boost/make_shared.hpp>
#include "mode.h"
#include "../utils/StaticData.h"
Include dependency graph for simplePPL.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [SimplePPL](#)
Filtering mode 1: simple perplexity.

7.7.1 Detailed Description

Derived class to handle filtering mode 1: simple perplexity.

Author

Anthony Rousseau

Version

1.2.0

Date

19 August 2013

7.8 include/phrasetable.h File Reference

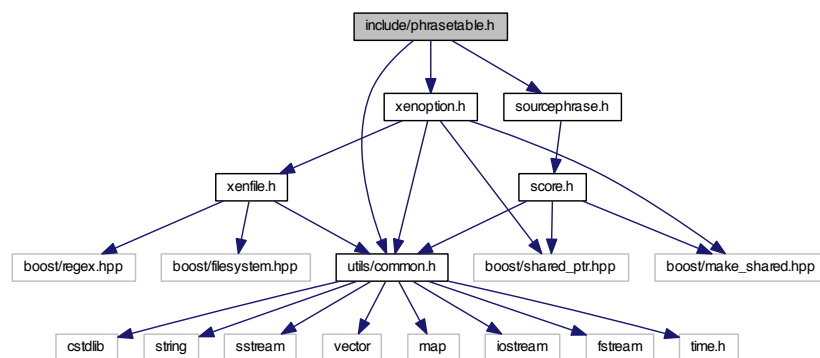
Class handling phrase-table related functionalities.

```
#include "utils/common.h"
```

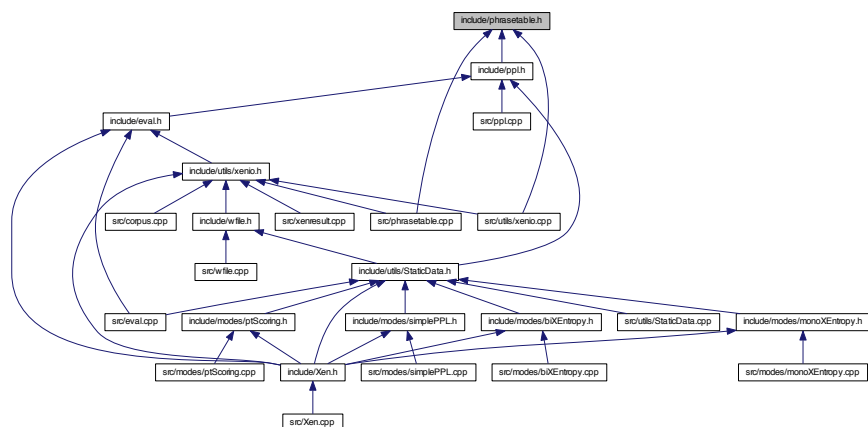
```
#include "xenoption.h"
```

```
#include "sourcephrase.h"
```

Include dependency graph for phrasetable.h:



This graph shows which files directly or indirectly include this file:



- class **PhraseTable**

7.8.1 Detailed Description

Author

Anthony Rousseau

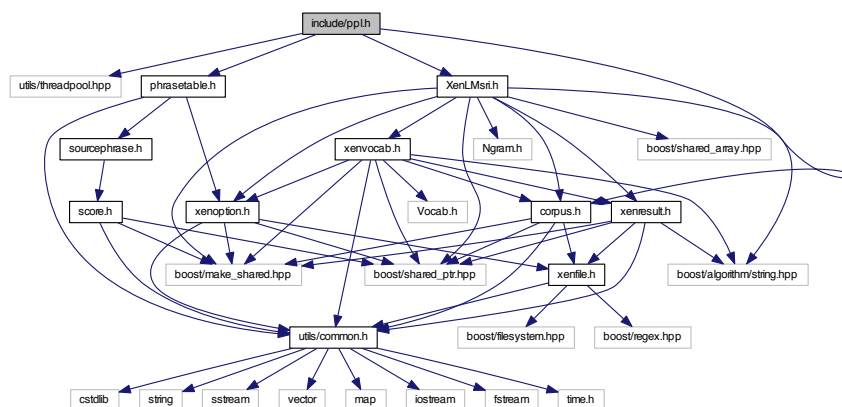
1.2.0

Date _____

19 August 2013

7.9 include/ppl.h File Reference

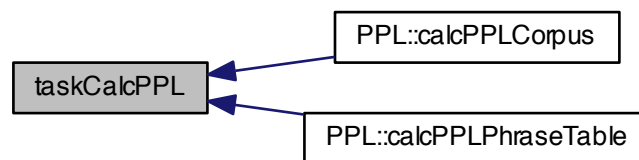
```
#include "utils/threadpool.hpp"
#include "corpus.h"
#include "phrasetable.h"
#include "XenLMsri.h"
Include dependency graph for ppl.h:
```



Parameters

<i>numLine</i>	: integer to the line number to compute perplexity for
<i>line</i>	: string to the text line to compute perplexity for
<i>ptrPPL</i>	: shared pointer on the vector of doubles containing the perplexity scores
<i>ptrLM</i>	: shared pointer on the language model to compute perplexity and cross-entropy from

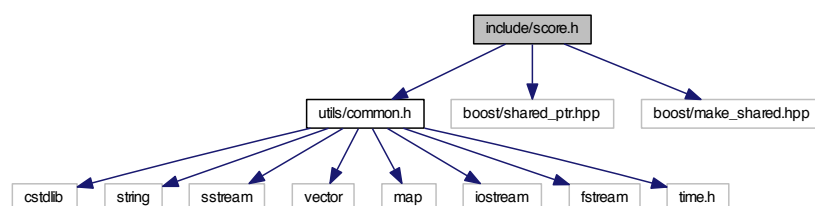
Here is the caller graph for this function:



7.10 include/score.h File Reference

Class holding the XenC scores representation.

```
#include "utils/common.h"
#include <boost/shared_ptr.hpp>
#include <boost/make_shared.hpp>
Include dependency graph for score.h:
```




```

graph TD
    include_similarity_h[include/similarity.h] --> xenvocab_h[xenvocab.h]
    include_similarity_h --> boost_algorithm_string_hpp[boost/algorithm/string.hpp]
    include_similarity_h --> boost_shared_ptr_hpp[boost/shared_ptr.hpp]
    include_similarity_h --> boost_make_shared_hpp[boost/make_shared.hpp]
    include_similarity_h --> utils_common_h[utils/common.h]
    xenvocab_h --> Vocab_h[Vocab.h]
    xenvocab_h --> xerresult_h[xerresult.h]
    xenvocab_h --> xeroption_h[xeroption.h]
    xenvocab_h --> corpus_h[corpus.h]
    xenvocab_h --> boost_algorithm_string_hpp
    xenvocab_h --> boost_shared_ptr_hpp
    xenvocab_h --> boost_make_shared_hpp
    xenvocab_h --> xenfile_h[xenfile.h]
    xenvocab_h --> utils_common_h
    xenfile_h --> boost_filesys_hpp[boost/filesystem.hpp]
    xenfile_h --> boost_regex_hpp[boost/regex.hpp]
    utils_common_h --> cstdlib[cstdlib]
    utils_common_h --> string[string]
    utils_common_h --> sstream[sstream]
    utils_common_h --> vector[vector]
    utils_common_h --> map[map]
    utils_common_h --> iostream[iostream]
    utils_common_h --> fstream[fstream]
    utils_common_h --> time_h[time.h]
  
```

- class **Similarity**
Class taking care of all the similarity measure computations.

- `typedef std::map< int, float > SimMap`
Map of integers as keys and floats as values to represent the similarity measures by sentence number.

Class taking care of all the similarity measure computations.

Anthony Rousseau

1.2.0

19 August 2013

Classes

- class [SourcePhrase](#)

Class holding a merged source phrase and all associated data.

7.12.1 Detailed Description

Class holding a merged source phrase and all associated data.

Author

Anthony Rousseau

Version

1.2.0

Date

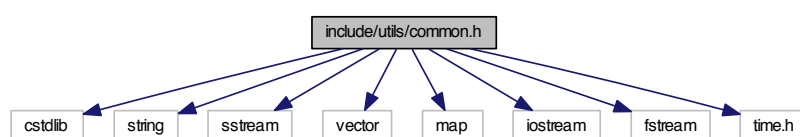
19 August 2013

7.13 include/utls/common.h File Reference

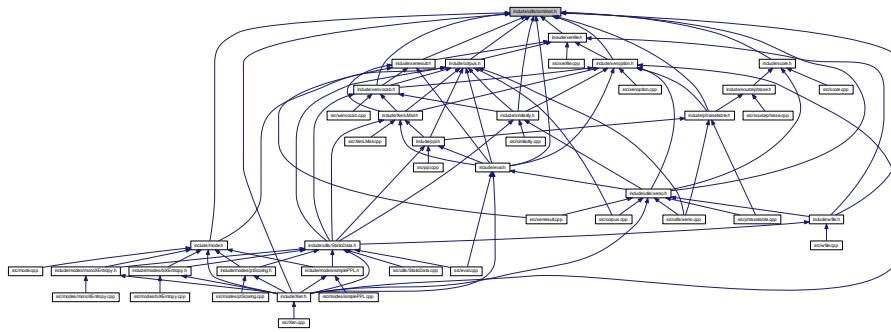
File containing all common classes/structures/functions of many classes of XenC.

```
#include <cstdlib>
#include <string>
#include <sstream>
#include <vector>
#include <map>
#include <iostream>
#include <fstream>
#include <time.h>
```

Include dependency graph for common.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct [_Options](#)
XenC options structure.
- struct [XenCommon::XenCEption](#)
XenC exception structure.
- class [XenCommon::Splitter](#)
Class defining a splitter.

Namespaces

- namespace [XenCommon](#)
Namespace containing all the common functions of XenC.

Typedefs

- typedef struct [_Options](#) [Options](#)
- typedef struct [_Options](#) * [LPOptions](#)

Functions

- template<typename T >
std::string [XenCommon::toString](#) (const T &Value)
Template converting a value into a string with a precision of 20.
- template<typename T >
std::string [XenCommon::toString0](#) (const T &Value)
Template converting a value into a string with no precision.
- template<typename T >
int [XenCommon::toInt](#) (const T &Value)
Template converting a value (generally a string) into an integer.
- template<typename T >
double [XenCommon::toDouble](#) (const T &Value)
Template converting a value (generally a string) into a double.
- template<typename A , typename B >
std::pair< B, A > [XenCommon::flip_pair](#) (const std::pair< A, B > &p)
Template flipping a pair key type with value type.

- `template<typename A , typename B >`
`std::multimap< B, A,`
`std::greater< B > > XenCommon::flip_map (const std::map< A, B > &src)`
Template flipping a multimap with descending order keys with values.
- `int XenCommon::wordCount (const std::string &str)`
Computes the word count of a string.
- `std::string XenCommon::getStdoutFromCommand (std::string cmd)`
Executes a system command and returns the output.

7.13.1 Detailed Description

File containing all common classes/structures/functions of many classes of XenC.

Author

Anthony Rousseau

Version

1.2.0

Date

19 August 2013

7.13.2 Typedef Documentation

7.13.2.1 `typedef struct _Options * LPOptions`

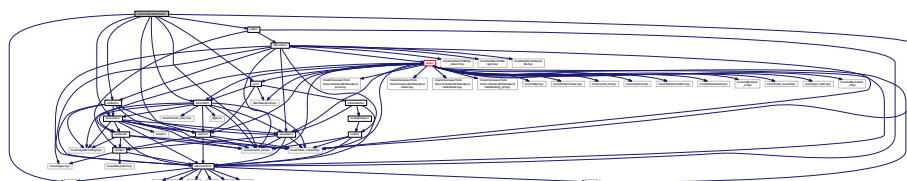
7.13.2.2 `typedef struct _Options Options`

7.14 include/utils/StaticData.h File Reference

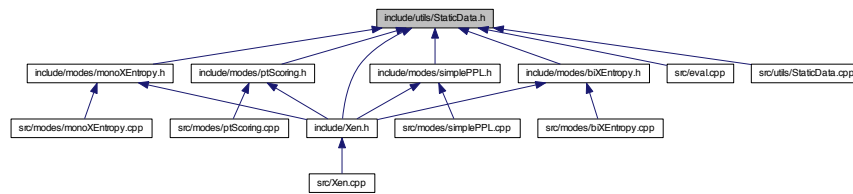
File handling all data objects used by XenC in a static way.

```
#include <cstdlib>
#include <boost/shared_ptr.hpp>
#include <boost/make_shared.hpp>
#include "corpus.h"
#include "XenLMsri.h"
#include "xenvocab.h"
#include "ppl.h"
#include "similarity.h"
#include "wfile.h"
```

Include dependency graph for StaticData.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [CorpusPair](#)
Tiny class holding two related [Corpus](#).
- class [LMPair](#)
Tiny class holding two related language models.
- class [VocabPair](#)
Tiny class holding the two vocabularies.
- class [PPLPair](#)
Tiny class holding two related [PPL](#) objects.
- class [PhraseTablePair](#)
Tiny class holding the two phrase-tables.
- class [MeanLMPair](#)
Tiny class holding two additional LMs for mean scoring feature.
- class [MeanPPLPair](#)
Tiny class holding two additional [PPL](#) objects for mean scoring feature.
- class [ScoreHolder](#)
Tiny class holding three [Score](#) objects (global scores, similarity, cross-entropy)
- class [StaticData](#)
Class gathering all data used and generated by [XenC](#).

7.14.1 Detailed Description

File handling all data objects used by [XenC](#) in a static way.

Author

Anthony Rousseau

Version

1.2.0

Date

19 August 2013

7.15 include/utls/xenio.h File Reference

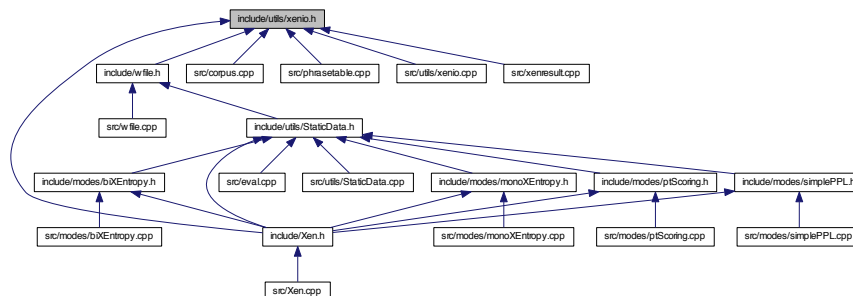
Class handling all input/output operations of XenC.

```
#include "common.h"
#include "eval.h"
#include "score.h"
#include "xenoption.h"
#include "similarity.h"
#include <boost/iostreams/filtering_stream.hpp>
#include <boost/iostreams/filter/gzip.hpp>
#include <boost/iostreams/device/file.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/make_shared.hpp>
#include <boost/regex.hpp>
```

Include dependency graph for xenio.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [XenIO](#)

Class handling all input/output operations of XenC.

7.15.1 Detailed Description

Class handling all input/output operations of XenC.

Author

Anthony Rousseau

Version

1.2.0

Date

19 August 2013

7.16 include/wfile.h File Reference

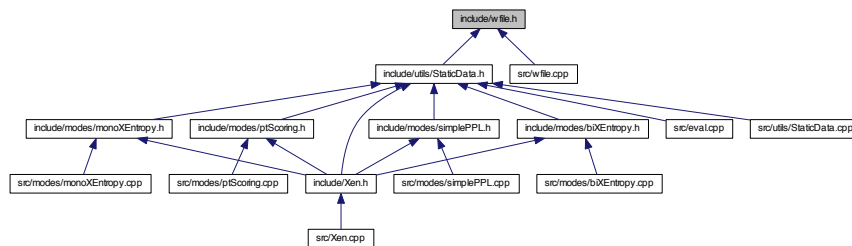
Class handling a file with values intended at weighting XenC scores.

```
#include "utils/common.h"
#include "utils/xenio.h"
#include "xenfile.h"
```

Include dependency graph for wfile.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Wfile](#)

Class handling a file with values intended at weighting XenC scores.

7.16.1 Detailed Description

Class handling a file with values intended at weighting XenC scores.

Author

Anthony Rousseau

Version

1.2.0

Date

19 August 2013

7.17 include/Xen.h File Reference

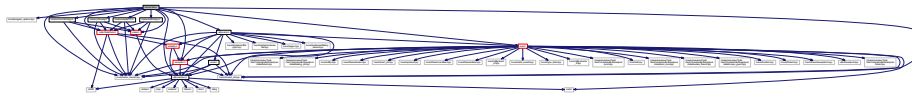
Main file of XenC, controls execution.


```

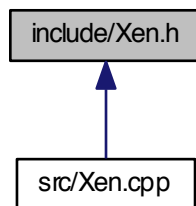
#include <boost/program_options.hpp>
#include <boost/make_shared.hpp>
#include <boost/shared_ptr.hpp>
#include "utils/common.h"
#include "utils/xenio.h"
#include "modes/simplePPL.h"
#include "modes/monoXEntropy.h"
#include "modes/biXEntropy.h"
#include "modes/ptScoring.h"
#include "eval.h"
#include "mode.h"
#include "xenoption.h"
#include "utils/StaticData.h"

```

Include dependency graph for Xen.h:



This graph shows which files directly or indirectly include this file:



Functions

- int [main](#) (int argc, char *argv[])
Main function of XenC.
- std::string [sanityCheck](#) ([XenOption](#) *opt)
Controls the mandatory options.
- std::string [getOutName](#) ([XenOption](#) *opt)
Computes the output file name.

7.17.1 Detailed Description

Main file of XenC, controls execution.

Author

Anthony Rousseau

Version

1.2.0

Date

19 August 2013

7.17.2 Function Documentation**7.17.2.1** `std::string getOutName (XenOption * opt)`

Computes the output file name.

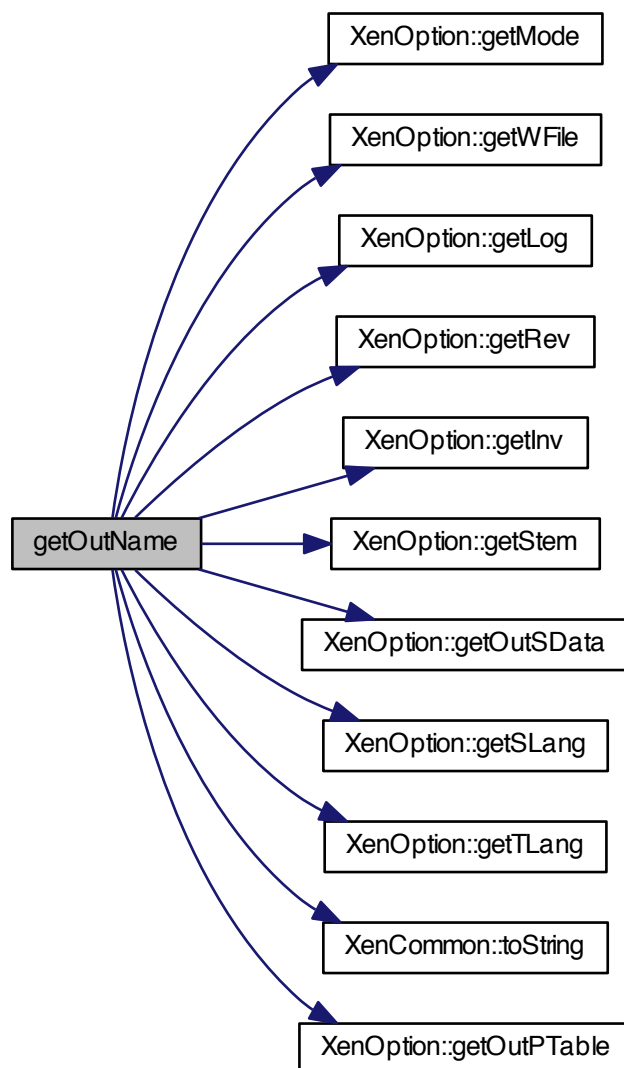
Parameters

<i>opt</i>	: XenOption object containing all the passed options
------------	--

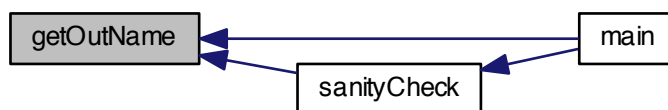
Returns

the output file name

Here is the call graph for this function:



Here is the caller graph for this function:



7.17.2.2 `int main (int argc, char * argv[])`

Main function of XenC.

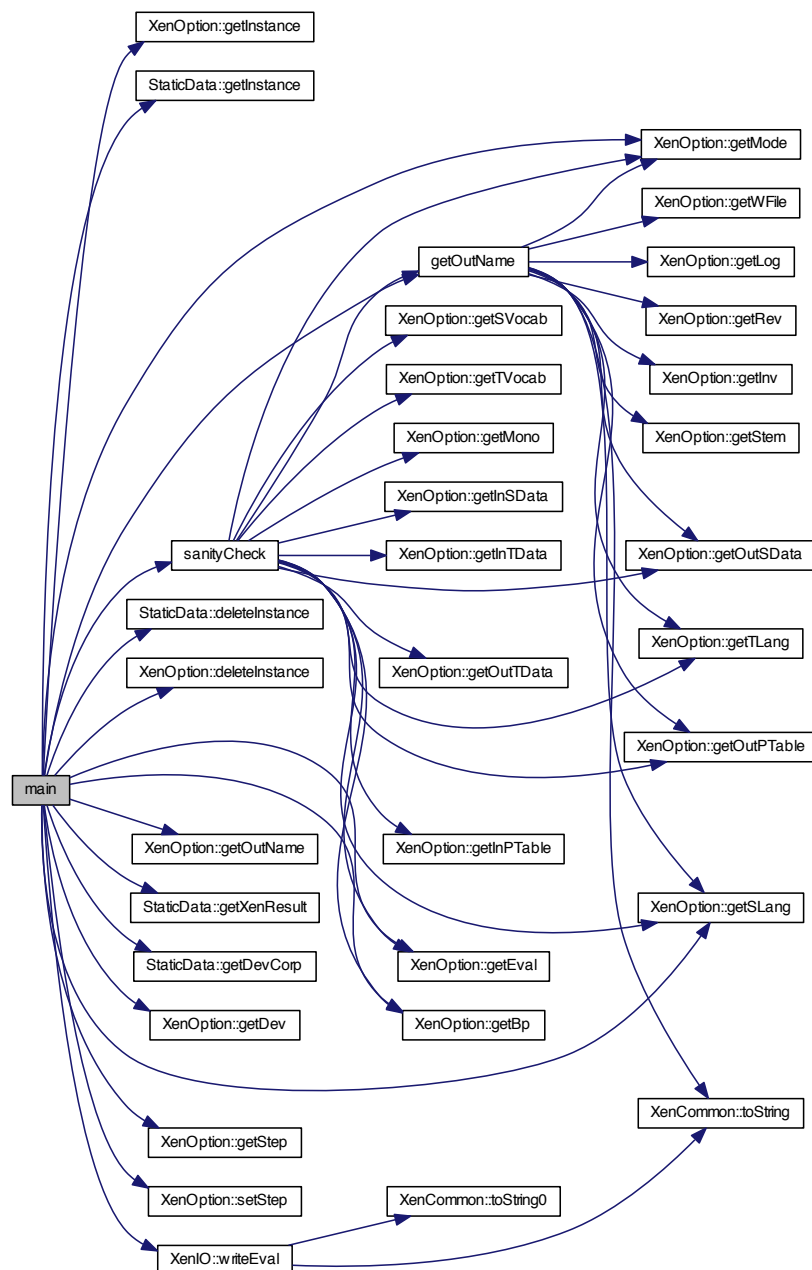
Parameters

<code>argc</code>	: number of arguments
<code>argv</code>	: passed arguments to the program

Returns

0 if execution ended well

Here is the call graph for this function:



7.17.2.3 `std::string sanityCheck (XenOption * opt)`

Controls the mandatory options.

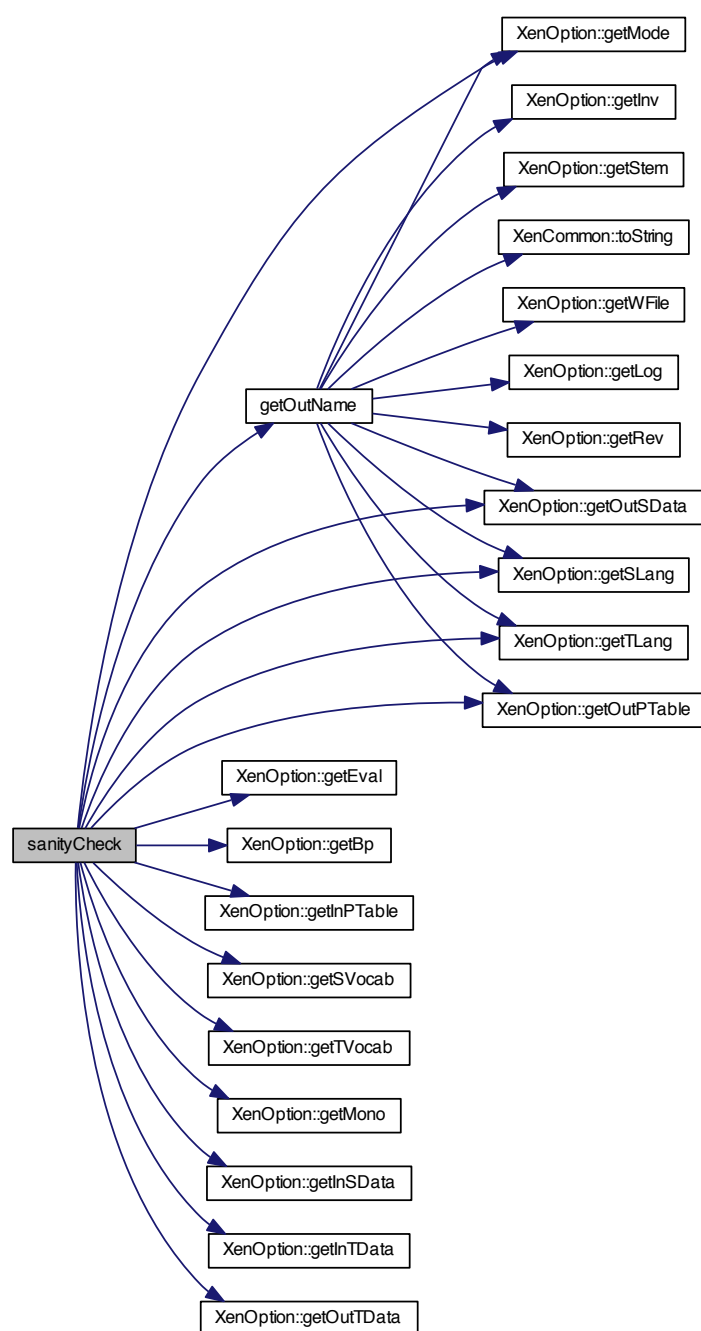
Parameters

<i>opt</i>	: XenOption object containing all the passed options
------------	--

Returns

0 if all is good, an error message otherwise

Here is the call graph for this function:



Here is the caller graph for this function:

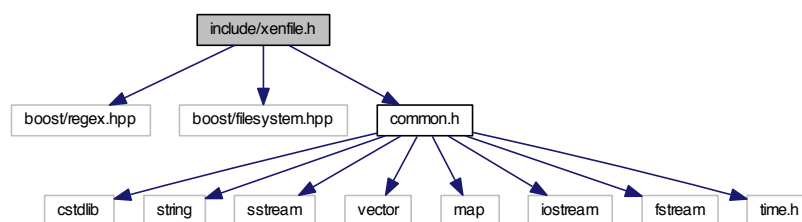


7.18 include/xenfile.h File Reference

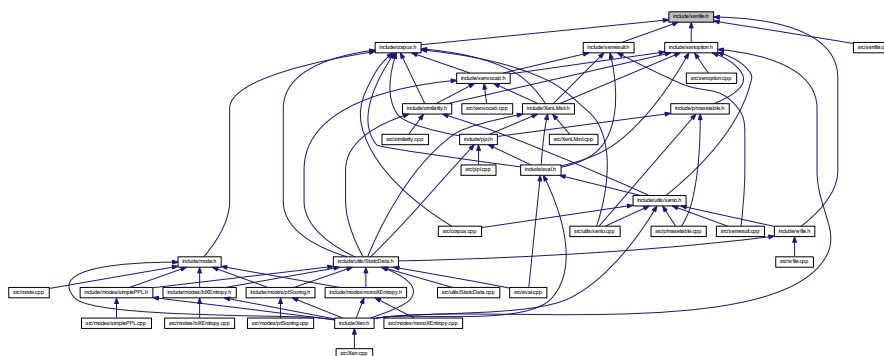
Class providing some basic functions around files.

```
#include <boost/regex.hpp>
#include <boost/filesystem.hpp>
#include "common.h"
```

Include dependency graph for xenfile.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [XenFile](#)

Class providing some basic functions around files.

7.18.1 Detailed Description

Class providing some basic functions around files.

Author

Anthony Rousseau

Version

1.2.0

Date

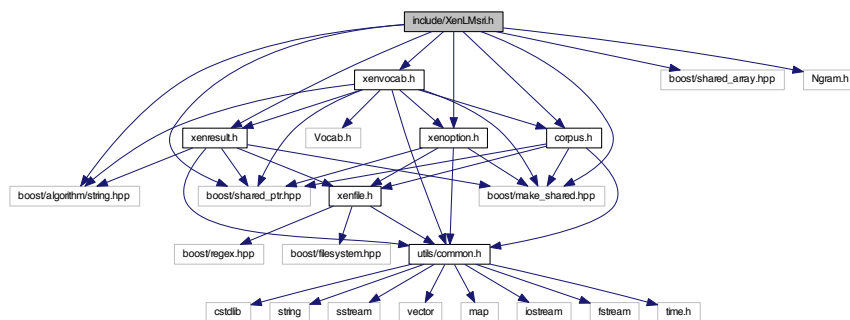
19 August 2013

7.19 include/XenLMsri.h File Reference

Class handling SRI LM estimation, loading, querying...

```
#include <boost/algorithm/string.hpp>
#include <boost/shared_array.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/make_shared.hpp>
#include "Ngram.h"
#include "corpus.h"
#include "xenvocab.h"
#include "xenoption.h"
#include "xenresult.h"
```

Include dependency graph for XenLMsri.h:



[illegible]

- class **XenLMsri**
Class handling SRI LM estimation, loading, querying...

- #define **MAX_ORDER** 9
Maximum LM order.
- #define **MAX_WORDS** 16384
Maximum tokens per line of text.
- #define **MAX_CHARS** **MAX_WORDS** * 16
Maximum characters per line of text.

Class handling SRI LM estimation, loading, querying...

Anthony Rousseau

1.2.0

19 August 2013

7.19.2.1 #define MAX_CHARS MAX_WORDS * 16

Maximum characters per line of text.

7.20.1 Detailed Description

Singleton class handling XenC options accessors/mutators.

Author

Anthony Rousseau

Version

1.2.0

Date

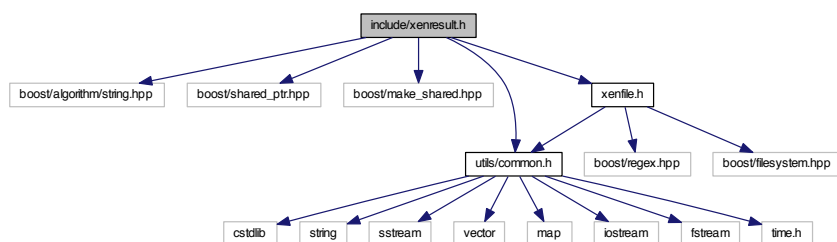
19 August 2013

7.21 include/xenresult.h File Reference

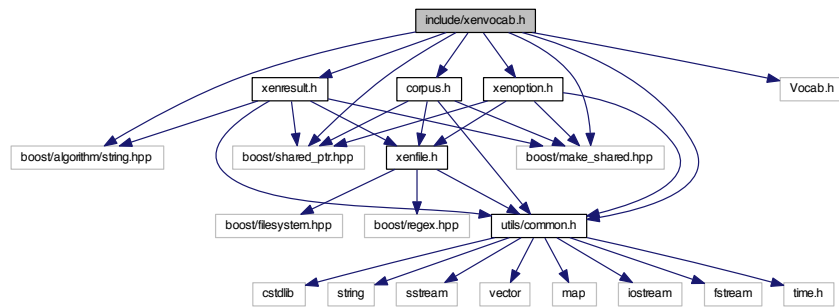
Class handling a XenC sorted result file for evaluation/best point.

```
#include <boost/algorithm/string.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/make_shared.hpp>
#include "utils/common.h"
#include "xenfile.h"
```

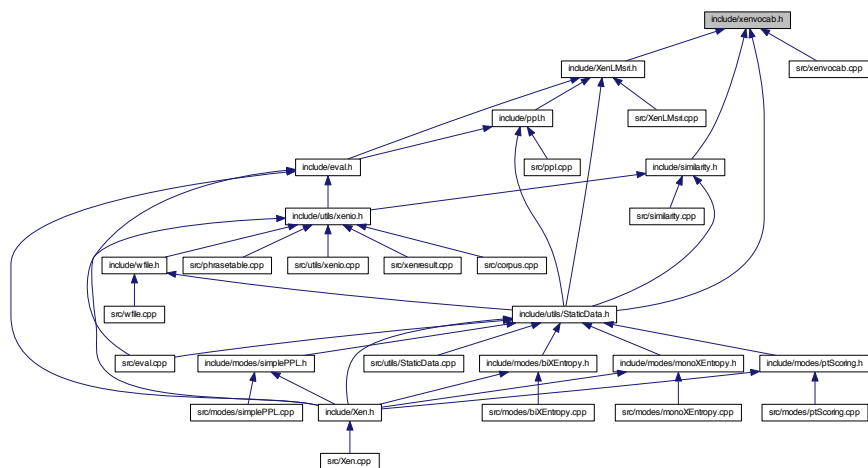
Include dependency graph for xenresult.h:



Include dependency graph for xenvocab.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [XenVocab](#)
Class handling a XenC vocabulary.

7.22.1 Detailed Description

Class handling a XenC vocabulary.

Author

Anthony Rousseau

Version

1.2.0

Date

19 August 2013

7.23 src/corpus.cpp File Reference

Class handling corpus-related functionalities.

```
#include "corpus.h"
```

```
#include "utils/xenio.h"
```

Include dependency graph for corpus.cpp:



7.23.1 Detailed Description

Class handling corpus-related functionalities.

Author

Anthony Rousseau

Version

1.2.0

Date

19 August 2013

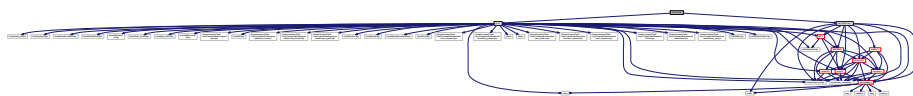
7.24 src/eval.cpp File Reference

Class handling evaluation system.

```
#include "eval.h"
```

```
#include "utils/StaticData.h"
```

Include dependency graph for eval.cpp:



Functions

- void [taskEval](#) (int pc, boost::shared_ptr< [XenResult](#) > ptrXR, boost::shared_ptr< [XenVocab](#) > ptrVoc, boost::shared_ptr< [Corpus](#) > ptrDevCorp, boost::shared_ptr< [EvalMap](#) > ptrDist)

Thread-safe evaluation function.

7.24.1 Detailed Description

Class handling evaluation system.

Author

Anthony Rousseau

Version

1.2.0

Date

19 August 2013

7.24.2 Function Documentation

7.24.2.1 void taskEval (int *pc*, boost::shared_ptr< XenResult > *ptrXR*, boost::shared_ptr< XenVocab > *ptrVoc*, boost::shared_ptr< Corpus > *ptrDevCorp*, boost::shared_ptr< EvalMap > *ptrDist*)

Thread-safe evaluation function.

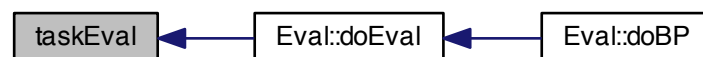
Parameters

<i>pc</i>	: integer representing the percentage of the scored out-of-domain corpus to take
<i>ptrXR</i>	: shared pointer on the XenResult object representing the selection result file
<i>ptrVoc</i>	: shared pointer on the XenVocab object representing the vocabulary to use for eval
<i>ptrDevCorp</i>	: shared pointer on the Corpus object representing the development set
<i>ptrDist</i>	: shared pointer on the EvalMap type containing the evaluation scores

Here is the call graph for this function:



Here is the caller graph for this function:

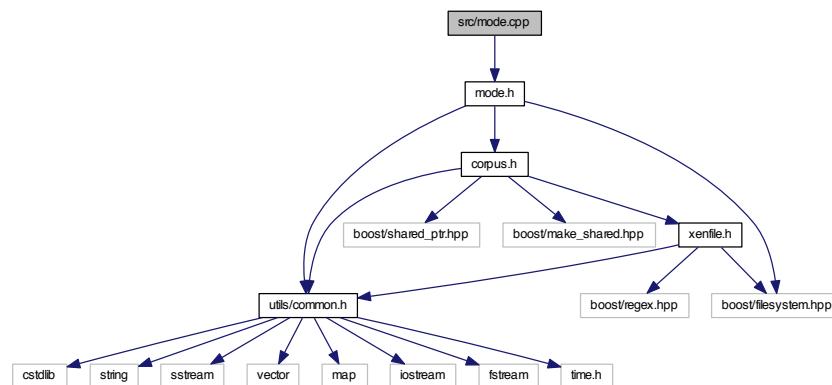


7.25 src/mode.cpp File Reference

Abstract class defining the filtering modes architecture.

```
#include "mode.h"
```

Include dependency graph for mode.cpp:



7.25.1 Detailed Description

Abstract class defining the filtering modes architecture.

Author

Anthony Rousseau

Version

1.2.0

Date

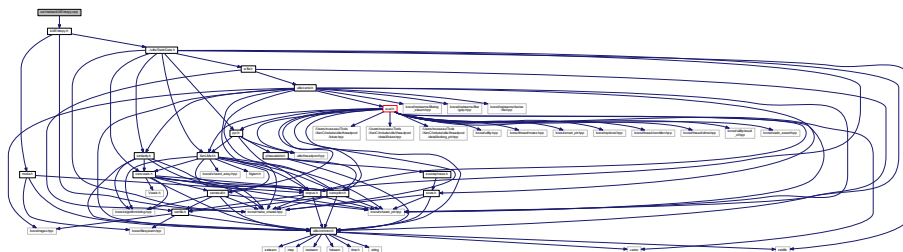
19 August 2013

7.26 src/modes/biXEntropy.cpp File Reference

Derived class to handle filtering mode 3: bilingual cross-entropy.

```
#include "biXEntropy.h"
```

Include dependency graph for biXEntropy.cpp:



7.26.1 Detailed Description

Derived class to handle filtering mode 3: bilingual cross-entropy.

Author

Anthony Rousseau

Version

1.2.0

Date

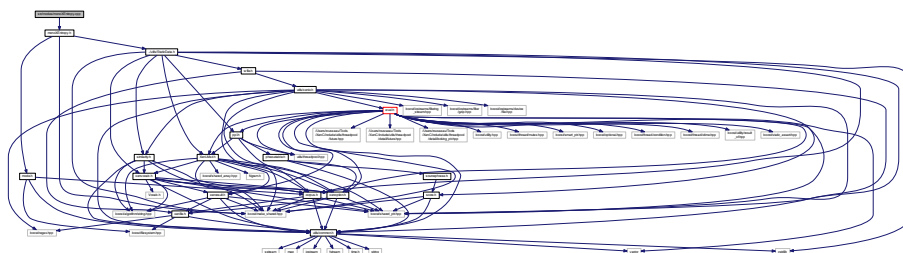
19 August 2013

7.27 src/modes/monoXEntropy.cpp File Reference

Derived class to handle filtering mode 2: monolingual cross-entropy.

```
#include "monoXEntropy.h"
```

Include dependency graph for monoXEntropy.cpp:



7.27.1 Detailed Description

Derived class to handle filtering mode 2: monolingual cross-entropy.

Author

Anthony Rousseau

Version

1.2.0

Date

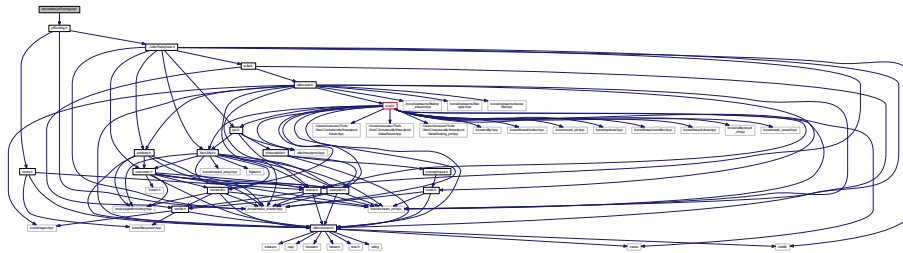
19 August 2013

7.28 src/modes/ptScoring.cpp File Reference

Derived class to handle filtering mode 4: phrase-table cross-entropy.

```
#include "ptScoring.h"
```

Include dependency graph for ptScoring.cpp:



7.28.1 Detailed Description

Derived class to handle filtering mode 4: phrase-table cross-entropy.

Author

Anthony Rousseau

Version

1.2.0

Date

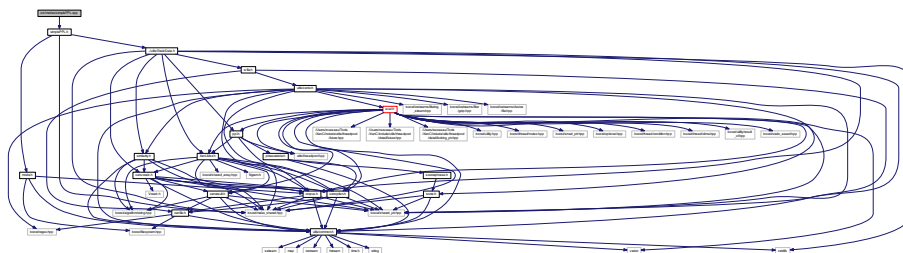
19 August 2013

7.29 src/modes/simplePPL.cpp File Reference

Derived class to handle filtering mode 1: simple perplexity.

```
#include "simplePPL.h"
```

Include dependency graph for simplePPL.cpp:



7.29.1 Detailed Description

Derived class to handle filtering mode 1: simple perplexity.

Author

Anthony Rousseau

Version

1.2.0

Date

19 August 2013

7.30 src/phrasetable.cpp File Reference

Class handling phrase-table related functionalities.

```
#include "phrasetable.h"
```

```
#include "utils/xenio.h"
```

Include dependency graph for phrasetable.cpp:



7.30.1 Detailed Description

Class handling phrase-table related functionalities.

Author

Anthony Rousseau

Version

1.2.0

Date

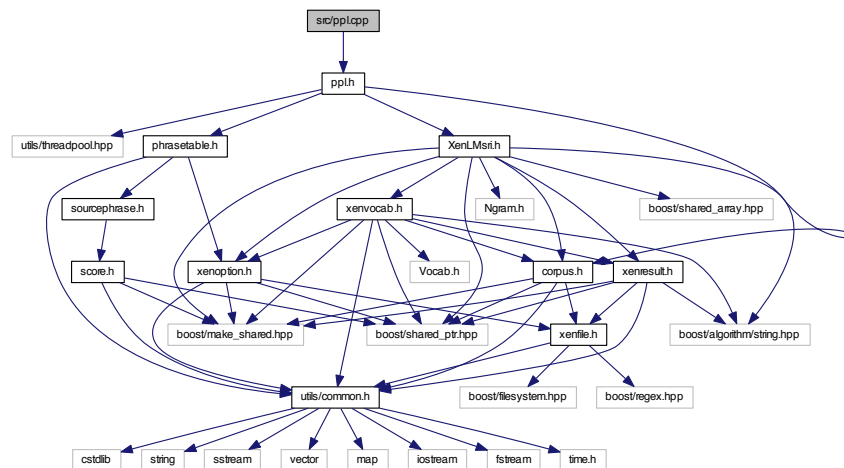
19 August 2013

7.31 src/ppl.cpp File Reference

Class handling the perplexity/cross-entropy computations.

```
#include "ppl.h"
```

Include dependency graph for ppl.cpp:



Functions

- void [taskCalcPPL](#) (int numLine, std::string line, boost::shared_ptr< std::vector< double > > ptrPPL, boost::shared_ptr< [XenLMsri](#) > ptrLM)

Thread-safe perplexity computation function.

7.31.1 Detailed Description

Class handling the perplexity/cross-entropy computations.

Author

Anthony Rousseau

Version

1.2.0

Date

19 August 2013

7.31.2 Function Documentation

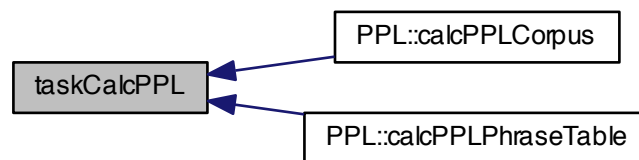
7.31.2.1 `void taskCalcPPL (int numLine, std::string line, boost::shared_ptr< std::vector< double > > ptrPPL, boost::shared_ptr< XenLMsri > ptrLM)`

Thread-safe perplexity computation function.

Parameters

<i>numLine</i>	: integer to the line number to compute perplexity for
<i>line</i>	: string to the text line to compute perplexity for
<i>ptrPPL</i>	: shared pointer on the vector of doubles containing the perplexity scores
<i>ptrLM</i>	: shared pointer on the language model to compute perplexity and cross-entropy from

Here is the caller graph for this function:

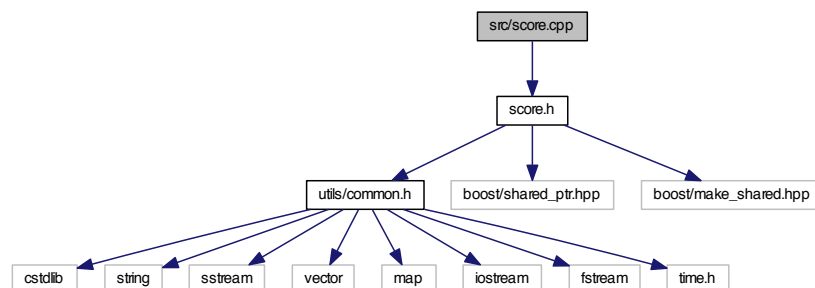


7.32 src/score.cpp File Reference

Class holding the XenC scores representation.

```
#include "score.h"
```

Include dependency graph for score.cpp:



7.32.1 Detailed Description

Class holding the XenC scores representation.

Author

Anthony Rousseau

Version

1.2.0

Date

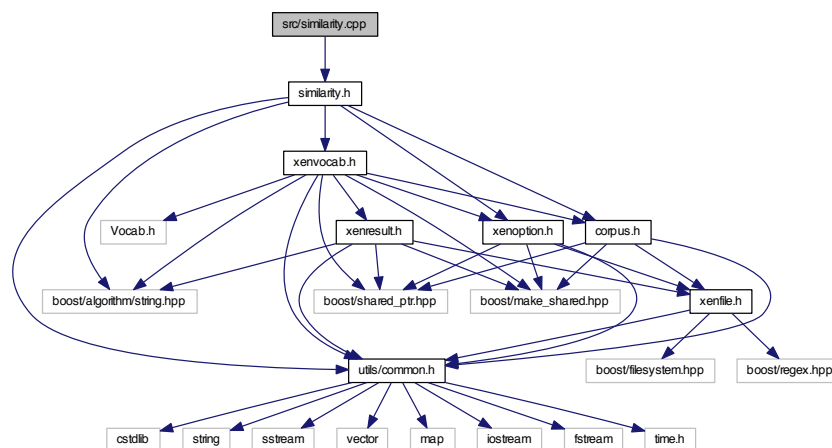
19 August 2013

7.33 src/similarity.cpp File Reference

Class taking care of all the similarity measure computations.

```
#include "similarity.h"
```

Include dependency graph for similarity.cpp:



7.33.1 Detailed Description

Class taking care of all the similarity measure computations.

Author

Anthony Rousseau

Version

1.2.0

Date

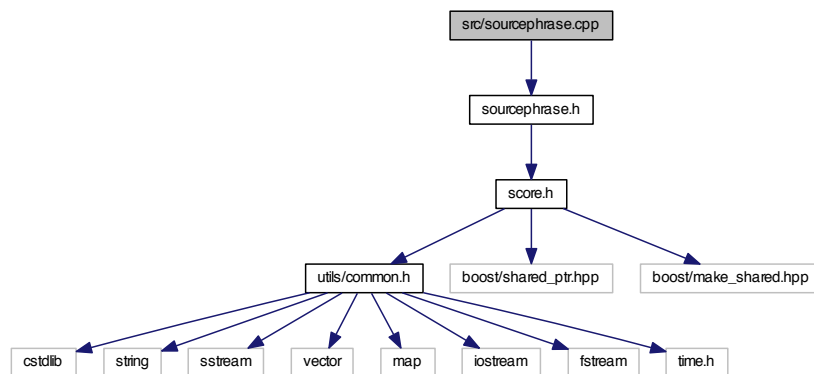
19 August 2013

7.34 src/sourcephrase.cpp File Reference

Class holding a merged source phrase and all associated data.

```
#include "sourcephrase.h"
```

Include dependency graph for sourcephrase.cpp:



7.34.1 Detailed Description

Class holding a merged source phrase and all associated data.

Author

Anthony Rousseau

Version

1.2.0

Date

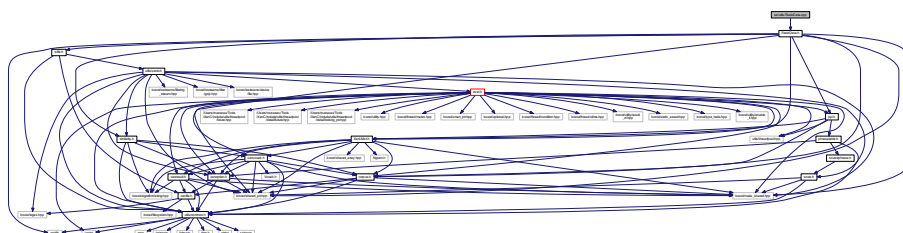
19 August 2013

7.35 src/utils/StaticData.cpp File Reference

File handling all data objects used by XenC in a static way.

```
#include "StaticData.h"
```

Include dependency graph for StaticData.cpp:



7.35.1 Detailed Description

File handling all data objects used by XenC in a static way.

Author

Anthony Rousseau

Version

1.2.0

Date

19 August 2013

7.36 src/utils/xenio.cpp File Reference

Class handling all input/output operations of XenC.

```
#include "xenio.h"
#include "corpus.h"
#include "phrasetable.h"
Include dependency graph for xenio.cpp:
```



7.36.1 Detailed Description

Class handling all input/output operations of XenC.

Author

Anthony Rousseau

Version

1.2.0

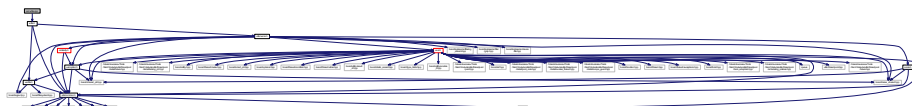
Date

19 August 2013

7.37 src/wfile.cpp File Reference

Class handling a file with values intended at weighting XenC scores.

```
#include "wfile.h"
Include dependency graph for wfile.cpp:
```



7.37.1 Detailed Description

Class handling a file with values intended at weighting XenC scores.

Author

Anthony Rousseau

Version

1.2.0

Date

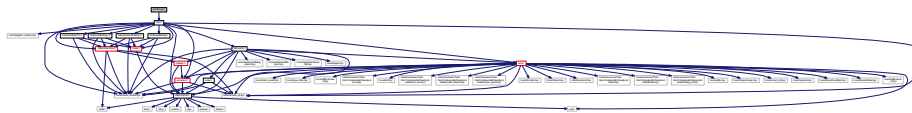
19 August 2013

7.38 src/Xen.cpp File Reference

Main file of XenC, controls execution.

```
#include "Xen.h"
```

Include dependency graph for Xen.cpp:



Functions

- int [main](#) (int argc, char *argv[])
Main function of XenC.
- std::string [sanityCheck](#) ([XenOption](#) *opt)
Controls the mandatory options.
- std::string [getOutName](#) ([XenOption](#) *opt)
Computes the output file name.

Variables

- const string [version](#) = "1.2.0"

7.38.1 Detailed Description

Main file of XenC, controls execution.

Author

Anthony Rousseau

Version

1.2.0

Date

19 August 2013

7.38.2 Function Documentation

7.38.2.1 `std::string getOutName (XenOption * opt)`

Computes the output file name.

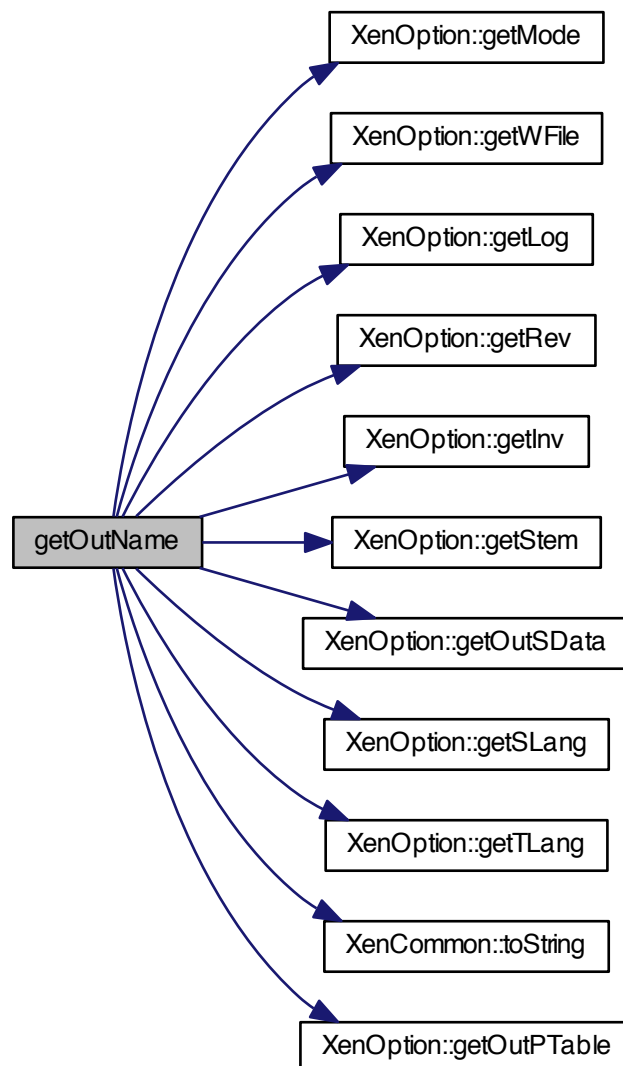
Parameters

<code>opt</code>	: XenOption object containing all the passed options
------------------	--

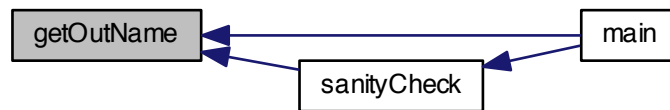
Returns

the output file name

Here is the call graph for this function:



Here is the caller graph for this function:



7.38.2.2 `int main (int argc, char * argv[])`

Main function of XenC.

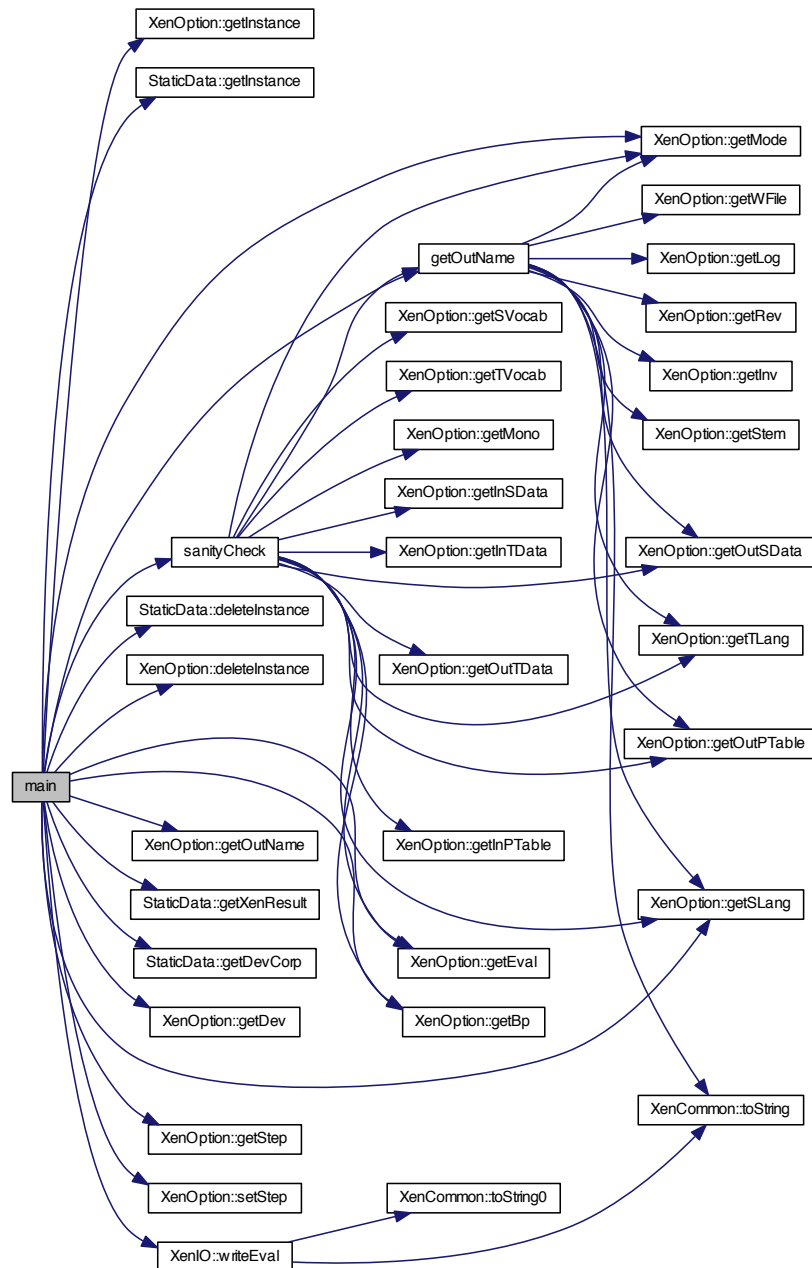
Parameters

<i>argc</i>	: number of arguments
<i>argv</i>	: passed arguments to the program

Returns

0 if execution ended well

Here is the call graph for this function:



7.38.2.3 std::string sanityCheck (XenOption * opt)

Controls the mandatory options.

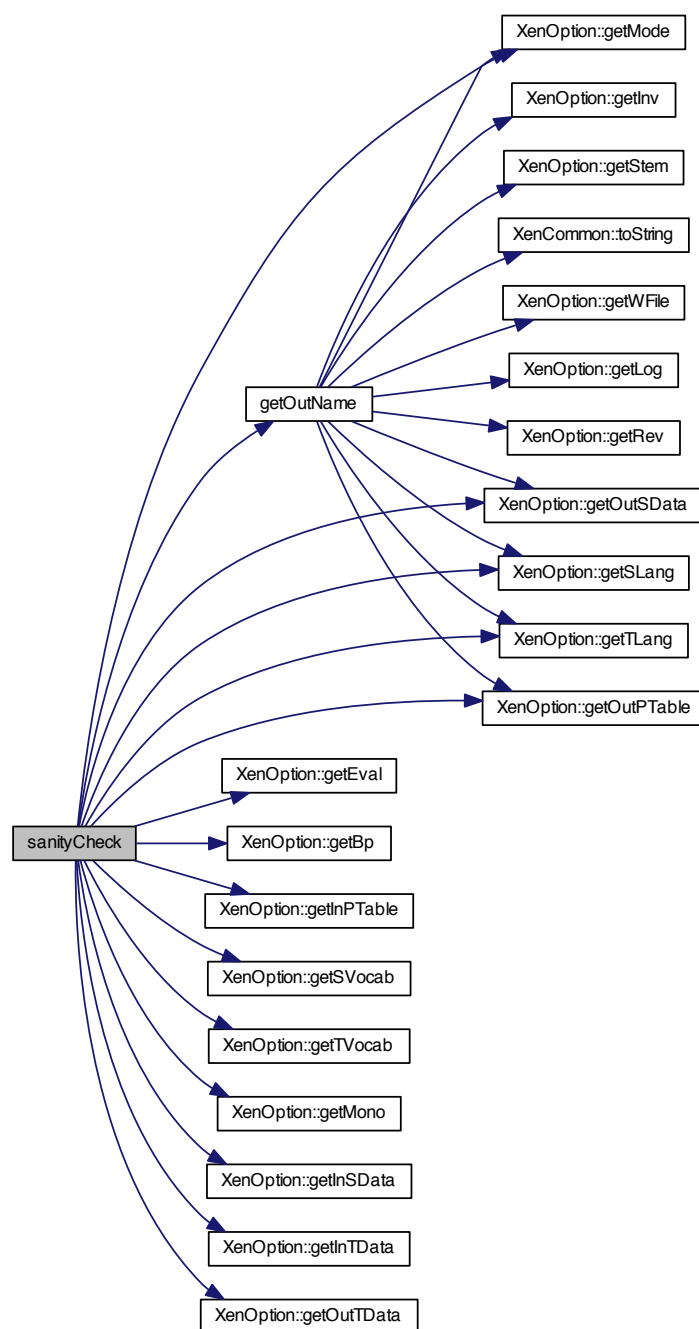
Parameters

<i>opt</i>	: XenOption object containing all the passed options
------------	--

Returns

0 if all is good, an error message otherwise

Here is the call graph for this function:



Here is the caller graph for this function:



7.38.3 Variable Documentation

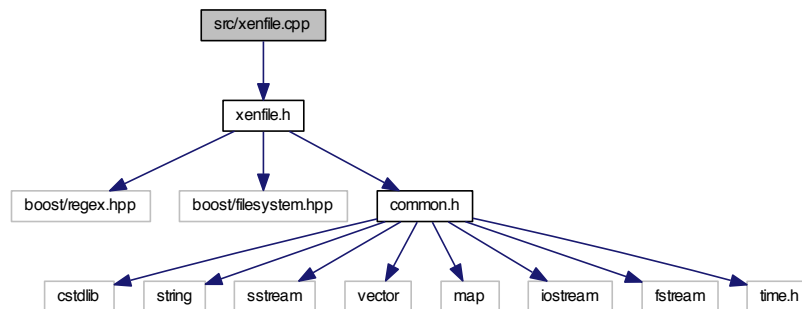
7.38.3.1 `const string version = "1.2.0"`

7.39 `src/xenfile.cpp` File Reference

Class providing some basic functions around files.

```
#include "xenfile.h"
```

Include dependency graph for `xenfile.cpp`:



7.39.1 Detailed Description

Class providing some basic functions around files.

Author

Anthony Rousseau

Version

1.2.0

Date

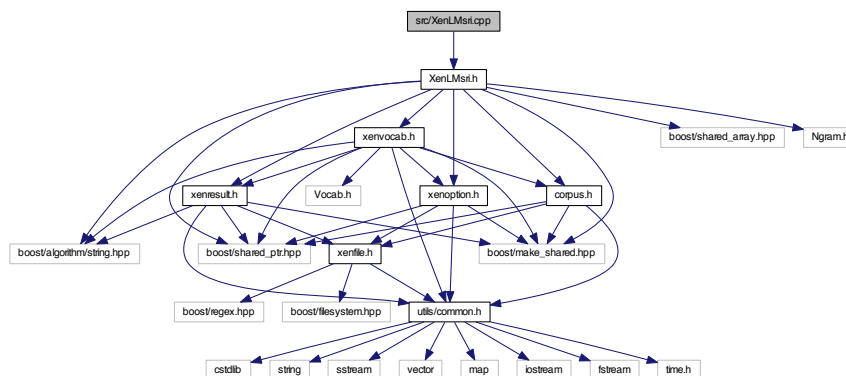
19 August 2013

7.40 src/XenLMsri.cpp File Reference

Class handling SRI LM estimation, loading, querying...

```
#include "XenLMsri.h"
```

Include dependency graph for XenLMsri.cpp:



Macros

- #define USE_STATS(what) (ptrNStats->what)
- #define USE_STATS(what) (ptrNStats->what)

7.40.1 Detailed Description

Class handling SRI LM estimation, loading, querying...

Author

Anthony Rousseau

Version

1.2.0

Date _____

19 August 2013

7.40.2 Macro Definition Documentation

```
7.40.2.1 #define USE_STATS( what ) (ptrNStats->what)
```

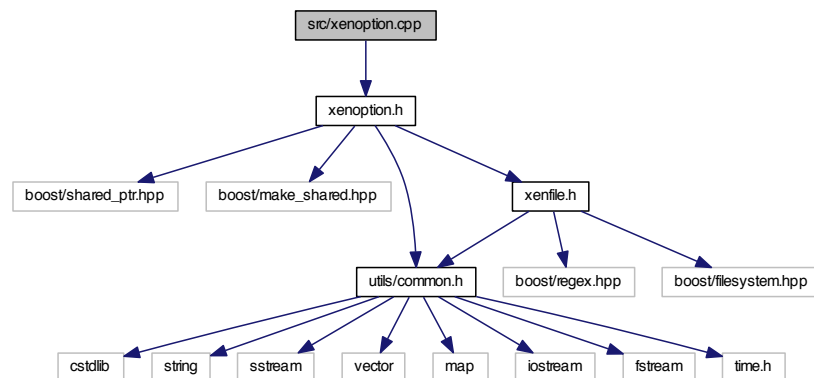
```
7.40.2.2 #define USE_STATS( what ) (ptrNStats->what)
```

7.41 src/xenoption.cpp File Reference

Singleton class handling XenC options accessors/mutators.

```
#include "xenoption.h"
```

Include dependency graph for xenoption.cpp:



7.41.1 Detailed Description

Singleton class handling XenC options accessors/mutators.

Author

Anthony Rousseau

Version

1.2.0

Date

19 August 2013

7.42 src/xenresult.cpp File Reference

Class handling a XenC sorted result file for evaluation/best point.

```
#include "xenresult.h"
```

```
#include "utils/xenio.h"
```

Include dependency graph for xenresult.cpp:



7.42.1 Detailed Description

Class handling a XenC sorted result file for evaluation/best point.

Author

Anthony Rousseau

Version

1.2.0

Date

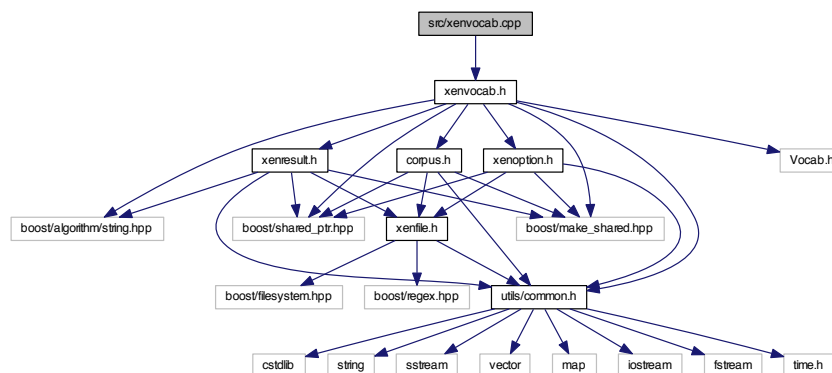
19 August 2013

7.43 src/xenvocab.cpp File Reference

Class handling a XenC vocabulary.

```
#include "xenvocab.h"
```

Include dependency graph for xenvocab.cpp:



7.43.1 Detailed Description

Class handling a XenC vocabulary.

Author

Anthony Rousseau

Version

1.2.0

Date

19 August 2013