

XenC

2.0.0

Generated by Doxygen 1.8.11

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	11
5.1 double_conversion Namespace Reference	11
5.1.1 Enumeration Type Documentation	12
5.1.1.1 BignumDtoaMode	12
5.1.1.2 FastDtoaMode	12
5.1.2 Function Documentation	12
5.1.2.1 BignumDtoa(double v, BignumDtoaMode mode, int requested_digits, Vector< char > buffer, int *length, int *point)	12
5.1.2.2 BitCast(const Source &source)	12
5.1.2.3 BitCast(Source *source)	12
5.1.2.4 double_to_uint64(double d)	12
5.1.2.5 FastDtoa(double d, FastDtoaMode mode, int requested_digits, Vector< char > buffer, int *length, int *decimal_point)	13
5.1.2.6 FastFixedDtoa(double v, int fractional_count, Vector< char > buffer, int *length, int *decimal_point)	13

5.1.2.7	float_to_uint32(float f)	13
5.1.2.8	Max(T a, T b)	13
5.1.2.9	Min(T a, T b)	13
5.1.2.10	StrLength(const char *string)	13
5.1.2.11	Strtod(Vector< const char > buffer, int exponent)	13
5.1.2.12	Strtof(Vector< const char > buffer, int exponent)	13
5.1.2.13	uint32_to_float(uint32_t d32)	13
5.1.2.14	uint64_to_double(uint64_t d64)	14
5.1.3	Variable Documentation	14
5.1.3.1	kCharSize	14
5.1.3.2	kFastDtoaMaximalLength	14
5.1.3.3	kFastDtoaMaximalSingleLength	14
5.2	XenCommon Namespace Reference	14
5.2.1	Detailed Description	15
5.2.2	Function Documentation	15
5.2.2.1	flip_map(const std::map< A, B > &src)	15
5.2.2.2	flip_pair(const std::pair< A, B > &p)	15
5.2.2.3	getStdoutFromCommand(std::string cmd)	16
5.2.2.4	toDouble(const T &Value)	16
5.2.2.5	toInt(const T &Value)	17
5.2.2.6	toString(const T &Value)	17
5.2.2.7	toString0(const T &Value)	18
5.2.2.8	wordCount(const std::string &str)	19

6 Class Documentation	21
6.1 _Options Struct Reference	21
6.1.1 Detailed Description	23
6.1.2 Member Data Documentation	23
6.1.2.1 bp	23
6.1.2.2 dev	23
6.1.2.3 eval	23
6.1.2.4 exclOOVs	24
6.1.2.5 fullVoc	24
6.1.2.6 inSData	24
6.1.2.7 inSLM	24
6.1.2.8 inSStem	24
6.1.2.9 inTData	24
6.1.2.10 inTLM	24
6.1.2.11 inToks	24
6.1.2.12 inTStem	24
6.1.2.13 inv	24
6.1.2.14 iPTable	25
6.1.2.15 local	25
6.1.2.16 log	25
6.1.2.17 maxEvalPC	25
6.1.2.18 mean	25
6.1.2.19 memPC	25
6.1.2.20 minblk	25
6.1.2.21 mode	25
6.1.2.22 mono	25
6.1.2.23 name	25
6.1.2.24 oPTable	26
6.1.2.25 order	26
6.1.2.26 outName	26

6.1.2.27	outSData	26
6.1.2.28	outSLM	26
6.1.2.29	outSStem	26
6.1.2.30	outTData	26
6.1.2.31	outTLM	26
6.1.2.32	outToks	26
6.1.2.33	outTStem	26
6.1.2.34	pc	27
6.1.2.35	rev	27
6.1.2.36	sampleSize	27
6.1.2.37	sim	27
6.1.2.38	simOnly	27
6.1.2.39	sLang	27
6.1.2.40	sortblk	27
6.1.2.41	sortOnly	27
6.1.2.42	stem	27
6.1.2.43	step	27
6.1.2.44	sVocab	28
6.1.2.45	temp	28
6.1.2.46	threads	28
6.1.2.47	tLang	28
6.1.2.48	tVocab	28
6.1.2.49	vecSize	28
6.1.2.50	version	28
6.1.2.51	wFile	28
6.2	double_conversion::Bignum Class Reference	28
6.2.1	Constructor & Destructor Documentation	29
6.2.1.1	Bignum()	29
6.2.2	Member Function Documentation	29
6.2.2.1	AddBignum(const Bignum &other)	29

6.2.2.2	AddUInt16(uint16_t operand)	29
6.2.2.3	AddUInt64(uint64_t operand)	29
6.2.2.4	AssignBignum(const Bignum &other)	30
6.2.2.5	AssignDecimalString(Vector< const char > value)	30
6.2.2.6	AssignHexString(Vector< const char > value)	30
6.2.2.7	AssignPowerUInt16(uint16_t base, int exponent)	30
6.2.2.8	AssignUInt16(uint16_t value)	30
6.2.2.9	AssignUInt64(uint64_t value)	30
6.2.2.10	Compare(const Bignum &a, const Bignum &b)	30
6.2.2.11	DivideModuloIntBignum(const Bignum &other)	30
6.2.2.12	Equal(const Bignum &a, const Bignum &b)	31
6.2.2.13	Less(const Bignum &a, const Bignum &b)	31
6.2.2.14	LessEqual(const Bignum &a, const Bignum &b)	31
6.2.2.15	MultiplyByPowerOfTen(int exponent)	32
6.2.2.16	MultiplyByUInt32(uint32_t factor)	32
6.2.2.17	MultiplyByUInt64(uint64_t factor)	32
6.2.2.18	PlusCompare(const Bignum &a, const Bignum &b, const Bignum &c)	32
6.2.2.19	PlusEqual(const Bignum &a, const Bignum &b, const Bignum &c)	33
6.2.2.20	PlusLess(const Bignum &a, const Bignum &b, const Bignum &c)	33
6.2.2.21	PlusLessEqual(const Bignum &a, const Bignum &b, const Bignum &c)	33
6.2.2.22	ShiftLeft(int shift_amount)	34
6.2.2.23	Square()	34
6.2.2.24	SubtractBignum(const Bignum &other)	34
6.2.2.25	Times10()	34
6.2.2.26	ToHexString(char *buffer, int buffer_size) const	34
6.2.3	Member Data Documentation	34
6.2.3.1	kMaxSignificantBits	34
6.3	BiXEntropy Class Reference	35
6.3.1	Detailed Description	35
6.3.2	Constructor & Destructor Documentation	36

6.3.2.1	BiXEntropy()	36
6.3.2.2	~BiXEntropy()	36
6.3.3	Member Function Documentation	36
6.3.3.1	launch()	36
6.4	Corpus Class Reference	36
6.4.1	Detailed Description	37
6.4.2	Constructor & Destructor Documentation	37
6.4.2.1	Corpus()	37
6.4.2.2	~Corpus()	37
6.4.3	Member Function Documentation	37
6.4.3.1	getLang() const	37
6.4.3.2	getLine(int line)	37
6.4.3.3	getPrint(int line)	38
6.4.3.4	getSize() const	38
6.4.3.5	getWC() const	38
6.4.3.6	getXenFile() const	39
6.4.3.7	initialize(boost::shared_ptr< XenFile > ptrData, std::string lg)	39
6.4.3.8	initialize(std::string filePath, std::string lg)	39
6.4.3.9	removeLine(int line)	39
6.5	CorpusPair Class Reference	40
6.5.1	Detailed Description	40
6.5.2	Constructor & Destructor Documentation	40
6.5.2.1	CorpusPair()	40
6.5.2.2	~CorpusPair()	41
6.5.3	Member Function Documentation	41
6.5.3.1	getPtrInCorp() const	41
6.5.3.2	getPtrOutCorp() const	41
6.6	double_conversion::DiyFp Class Reference	41
6.6.1	Constructor & Destructor Documentation	42
6.6.1.1	DiyFp()	42

6.6.1.2	DiyFp(uint64_t f, int e)	42
6.6.2	Member Function Documentation	42
6.6.2.1	e() const	42
6.6.2.2	f() const	42
6.6.2.3	Minus(const DiyFp &a, const DiyFp &b)	43
6.6.2.4	Multiply(const DiyFp &other)	43
6.6.2.5	Normalize()	43
6.6.2.6	Normalize(const DiyFp &a)	44
6.6.2.7	set_e(int new_value)	44
6.6.2.8	set_f(uint64_t new_value)	45
6.6.2.9	Subtract(const DiyFp &other)	45
6.6.2.10	Times(const DiyFp &a, const DiyFp &b)	45
6.6.3	Member Data Documentation	46
6.6.3.1	kSignificandSize	46
6.7	double_conversion::Double Class Reference	46
6.7.1	Constructor & Destructor Documentation	47
6.7.1.1	Double()	47
6.7.1.2	Double(double d)	47
6.7.1.3	Double(uint64_t d64)	47
6.7.1.4	Double(DiyFp diy_fp)	47
6.7.2	Member Function Documentation	47
6.7.2.1	AsDiyFp() const	47
6.7.2.2	AsNormalizedDiyFp() const	48
6.7.2.3	AsUint64() const	48
6.7.2.4	Exponent() const	49
6.7.2.5	Infinity()	49
6.7.2.6	IsDenormal() const	50
6.7.2.7	IsInfinite() const	51
6.7.2.8	IsNan() const	51
6.7.2.9	IsSpecial() const	51

6.7.2.10	LowerBoundaryIsCloser() const	52
6.7.2.11	NaN()	53
6.7.2.12	NextDouble() const	53
6.7.2.13	NormalizedBoundaries(DiyFp *out_m_minus, DiyFp *out_m_plus) const . . .	54
6.7.2.14	PreviousDouble() const	54
6.7.2.15	Sign() const	55
6.7.2.16	Significand() const	55
6.7.2.17	SignificandSizeForOrderOfMagnitude(int order)	56
6.7.2.18	UpperBoundary() const	56
6.7.2.19	value() const	57
6.7.3	Member Data Documentation	57
6.7.3.1	kExponentMask	57
6.7.3.2	kHiddenBit	57
6.7.3.3	kPhysicalSignificandSize	57
6.7.3.4	kSignificandMask	57
6.7.3.5	kSignificandSize	57
6.7.3.6	kSignMask	57
6.8	double_conversion::DoubleToStringConverter Class Reference	58
6.8.1	Member Enumeration Documentation	58
6.8.1.1	DtoaMode	58
6.8.1.2	Flags	59
6.8.2	Constructor & Destructor Documentation	59
6.8.2.1	DoubleToStringConverter(int flags, const char *infinity_symbol, const char *nan_symbol, char exponent_character, int decimal_in_shortest_low, int decimal_in_shortest_high, int max_leading_padding_zeroes_in_precision_mode, int max_trailing_padding_zeroes_in_precision_mode)	59
6.8.3	Member Function Documentation	59
6.8.3.1	DoubleToAscii(double v, DtoaMode mode, int requested_digits, char *buffer, int buffer_length, bool *sign, int *length, int *point)	59
6.8.3.2	EcmaScriptConverter()	59
6.8.3.3	ToExponential(double value, int requested_digits, StringBuilder *result_builder) const	60
6.8.3.4	ToFixed(double value, int requested_digits, StringBuilder *result_builder) const	60

6.8.3.5	ToPrecision(double value, int precision, StringBuilder *result_builder) const	60
6.8.3.6	ToShortest(double value, StringBuilder *result_builder) const	61
6.8.3.7	ToShortestSingle(float value, StringBuilder *result_builder) const	61
6.8.4	Member Data Documentation	61
6.8.4.1	kBase10MaximalLength	61
6.8.4.2	kMaxExponentialDigits	61
6.8.4.3	kMaxFixedDigitsAfterPoint	61
6.8.4.4	kMaxFixedDigitsBeforePoint	61
6.8.4.5	kMaxPrecisionDigits	61
6.8.4.6	kMinPrecisionDigits	61
6.9	Eval Class Reference	61
6.9.1	Detailed Description	62
6.9.2	Constructor & Destructor Documentation	62
6.9.2.1	Eval()	62
6.9.2.2	Eval(std::string distFile)	62
6.9.2.3	~Eval()	63
6.9.3	Member Function Documentation	63
6.9.3.1	doBP()	63
6.9.3.2	doEval(int high, int low)	64
6.9.3.3	getBP()	65
6.9.3.4	getDist() const	65
6.10	LMPair Class Reference	65
6.10.1	Detailed Description	66
6.10.2	Constructor & Destructor Documentation	66
6.10.2.1	LMPair()	66
6.10.2.2	~LMPair()	66
6.10.3	Member Function Documentation	66
6.10.3.1	getPtrInLM() const	66
6.10.3.2	getPtrOutLM() const	66
6.11	MeanLMPair Class Reference	66

6.11.1	Detailed Description	67
6.11.2	Constructor & Destructor Documentation	67
6.11.2.1	MeanLMPair()	67
6.11.2.2	\sim MeanLMPair()	67
6.11.3	Member Function Documentation	67
6.11.3.1	getPtrOutLM2() const	67
6.11.3.2	getPtrOutLM3() const	67
6.12	MeanPPLPair Class Reference	68
6.12.1	Detailed Description	68
6.12.2	Constructor & Destructor Documentation	68
6.12.2.1	MeanPPLPair()	68
6.12.2.2	\sim MeanPPLPair()	68
6.12.3	Member Function Documentation	68
6.12.3.1	getPtrOutPPL2() const	68
6.12.3.2	getPtrOutPPL3() const	69
6.13	Mode Class Reference	69
6.13.1	Detailed Description	70
6.13.2	Constructor & Destructor Documentation	70
6.13.2.1	\sim Mode()=0	70
6.13.3	Member Function Documentation	70
6.13.3.1	extractSample(boost::shared_ptr< Corpus > ptrCorp, int sSize, bool mean)	70
6.13.3.2	findSampleSize(boost::shared_ptr< Corpus > idCorp, boost::shared_ptr< Corpus > oodCorp)	71
6.13.3.3	launch()=0	72
6.14	MonoXEntropy Class Reference	72
6.14.1	Detailed Description	73
6.14.2	Constructor & Destructor Documentation	73
6.14.2.1	MonoXEntropy()	73
6.14.2.2	\sim MonoXEntropy()	73
6.14.3	Member Function Documentation	74
6.14.3.1	launch()	74

6.15 PhraseTable Class Reference	76
6.15.1 Detailed Description	76
6.15.2 Constructor & Destructor Documentation	76
6.15.2.1 PhraseTable()	76
6.15.2.2 ~PhraseTable()	76
6.15.3 Member Function Documentation	76
6.15.3.1 getAlignment(int n)	76
6.15.3.2 getCounts(int ph)	77
6.15.3.3 getScores(int n)	78
6.15.3.4 getSize() const	79
6.15.3.5 getSource(int n)	79
6.15.3.6 getSrcPhrases()	80
6.15.3.7 getTarget(int n)	80
6.15.3.8 getXenFile() const	81
6.15.3.9 initialize(boost::shared_ptr< XenFile > ptrData)	81
6.15.3.10 setSrcPhrases(std::vector< SourcePhrase > vSP)	82
6.16 PhraseTablePair Class Reference	82
6.16.1 Detailed Description	83
6.16.2 Constructor & Destructor Documentation	83
6.16.2.1 PhraseTablePair()	83
6.16.2.2 ~PhraseTablePair()	83
6.16.3 Member Function Documentation	83
6.16.3.1 getPtrInPT() const	83
6.16.3.2 getPtrOutPT() const	83
6.17 double_conversion::PowersOfTenCache Class Reference	83
6.17.1 Member Function Documentation	84
6.17.1.1 GetCachedPowerForBinaryExponentRange(int min_exponent, int max_exponent, DiyFp *power, int *decimal_exponent)	84
6.17.1.2 GetCachedPowerForDecimalExponent(int requested_exponent, DiyFp *power, int *found_exponent)	84
6.17.2 Member Data Documentation	84

6.17.2.1	kDecimalExponentDistance	84
6.17.2.2	kMaxDecimalExponent	84
6.17.2.3	kMinDecimalExponent	84
6.18	PPL Class Reference	84
6.18.1	Detailed Description	85
6.18.2	Constructor & Destructor Documentation	85
6.18.2.1	PPL()	85
6.18.2.2	~PPL()	85
6.18.3	Member Function Documentation	86
6.18.3.1	calcPPLCorpus()	86
6.18.3.2	calcPPLPhraseTable()	86
6.18.3.3	getCorpPPL()	86
6.18.3.4	getPPL(int n)	87
6.18.3.5	getSize() const	87
6.18.3.6	getXE(int n)	87
6.18.3.7	initialize(boost::shared_ptr< Corpus > ptrCorp, boost::shared_ptr< XenLMken > ptrLM)	87
6.18.3.8	initialize(boost::shared_ptr< PhraseTable > ptrPT, boost::shared_ptr< XenL← Mken > ptrLM, bool source)	87
6.19	PPLPair Class Reference	87
6.19.1	Detailed Description	88
6.19.2	Constructor & Destructor Documentation	88
6.19.2.1	PPLPair()	88
6.19.2.2	~PPLPair()	88
6.19.3	Member Function Documentation	88
6.19.3.1	getPtrInPPL() const	88
6.19.3.2	getPtrOutPPL() const	88
6.20	PTScoring Class Reference	89
6.20.1	Detailed Description	89
6.20.2	Constructor & Destructor Documentation	90
6.20.2.1	PTScoring()	90

6.20.2.2	<code>~PTScoring()</code>	90
6.20.3	<code>Member Function Documentation</code>	90
6.20.3.1	<code>launch()</code>	90
6.21	<code>Score Class Reference</code>	92
6.21.1	<code>Detailed Description</code>	92
6.21.2	<code>Constructor & Destructor Documentation</code>	92
6.21.2.1	<code>Score()</code>	92
6.21.2.2	<code>~Score()</code>	92
6.21.3	<code>Member Function Documentation</code>	92
6.21.3.1	<code>addScore(double sc)</code>	92
6.21.3.2	<code>calibrate()</code>	93
6.21.3.3	<code>getPrint(int n) const</code>	93
6.21.3.4	<code>getScore(int n) const</code>	93
6.21.3.5	<code>getSize() const</code>	94
6.21.3.6	<code>inverse()</code>	94
6.21.3.7	<code>removeScore(int n)</code>	94
6.22	<code>ScoreHolder Class Reference</code>	94
6.22.1	<code>Detailed Description</code>	95
6.22.2	<code>Constructor & Destructor Documentation</code>	95
6.22.2.1	<code>ScoreHolder()</code>	95
6.22.2.2	<code>~ScoreHolder()</code>	95
6.22.3	<code>Member Function Documentation</code>	95
6.22.3.1	<code>getPtrScores() const</code>	95
6.22.3.2	<code>getPtrScSimil() const</code>	95
6.22.3.3	<code>getPtrScXenC() const</code>	96
6.23	<code>Similarity Class Reference</code>	96
6.23.1	<code>Detailed Description</code>	96
6.23.2	<code>Constructor & Destructor Documentation</code>	96
6.23.2.1	<code>Similarity()</code>	96
6.23.2.2	<code>~Similarity()</code>	96

6.23.3 Member Function Documentation	96
6.23.3.1 getSim(int n)	96
6.23.3.2 getSize() const	97
6.23.3.3 initialize(boost::shared_ptr< Corpus > ptrInCorp, boost::shared_ptr< Corpus > ptrOutCorp, boost::shared_ptr< XenVocab > ptrVocab)	97
6.24 SimplePPL Class Reference	98
6.24.1 Detailed Description	98
6.24.2 Constructor & Destructor Documentation	99
6.24.2.1 SimplePPL()	99
6.24.2.2 ~SimplePPL()	99
6.24.3 Member Function Documentation	99
6.24.3.1 launch()	99
6.25 double_conversion::Single Class Reference	101
6.25.1 Constructor & Destructor Documentation	101
6.25.1.1 Single()	101
6.25.1.2 Single(float f)	101
6.25.1.3 Single(uint32_t d32)	101
6.25.2 Member Function Documentation	101
6.25.2.1 AsDiyFp() const	102
6.25.2.2 AsUint32() const	102
6.25.2.3 Exponent() const	102
6.25.2.4 Infinity()	102
6.25.2.5 IsDenormal() const	103
6.25.2.6 IsInfinite() const	103
6.25.2.7 IsNan() const	103
6.25.2.8 IsSpecial() const	103
6.25.2.9 LowerBoundaryIsCloser() const	103
6.25.2.10 NaN()	103
6.25.2.11 NormalizedBoundaries(DiyFp *out_m_minus, DiyFp *out_m_plus) const	103
6.25.2.12 Sign() const	104
6.25.2.13 Significand() const	104

6.25.2.14	UpperBoundary() const	104
6.25.2.15	value() const	104
6.25.3	Member Data Documentation	105
6.25.3.1	kExponentMask	105
6.25.3.2	kHiddenBit	105
6.25.3.3	kPhysicalSignificandSize	105
6.25.3.4	kSignificandMask	105
6.25.3.5	kSignificandSize	105
6.25.3.6	kSignMask	105
6.26	SourcePhrase Class Reference	105
6.26.1	Detailed Description	106
6.26.2	Constructor & Destructor Documentation	106
6.26.2.1	SourcePhrase(std::string src)	106
6.26.2.2	~SourcePhrase()	106
6.26.3	Member Function Documentation	106
6.26.3.1	addAlignments(std::string s)	106
6.26.3.2	addCounts(std::string s)	107
6.26.3.3	addScores(std::string s)	107
6.26.3.4	addTarget(std::string s)	108
6.26.3.5	getScoresXE() const	108
6.26.3.6	getSource() const	109
6.26.3.7	getTargetSize() const	109
6.27	XenCommon::Splitter Class Reference	109
6.27.1	Detailed Description	110
6.27.2	Member Typedef Documentation	110
6.27.2.1	size_type	110
6.27.3	Constructor & Destructor Documentation	110
6.27.3.1	Splitter()	110
6.27.3.2	Splitter(const std::string &src, const std::string &delim)	110
6.27.4	Member Function Documentation	110

6.27.4.1 operator[](size_type i)	110
6.27.4.2 reset(const std::string &src, const std::string &delim)	110
6.27.4.3 size() const	111
6.28 StaticData Class Reference	111
6.28.1 Detailed Description	112
6.28.2 Member Function Documentation	112
6.28.2.1 deleteInstance()	112
6.28.2.2 getDevCorp()	113
6.28.2.3 getInstance()	113
6.28.2.4 getMeanSourceLMs()	114
6.28.2.5 getMeanSourcePPLs()	114
6.28.2.6 getMeanTargetLMs()	115
6.28.2.7 getMeanTargetPPLs()	115
6.28.2.8 getPTPairs()	115
6.28.2.9 getScHold()	116
6.28.2.10 getSim()	116
6.28.2.11 getSourceCorps()	117
6.28.2.12 getSourceLMs()	117
6.28.2.13 getSourcePPLs()	118
6.28.2.14 getStemSourceCorps()	118
6.28.2.15 getStemSourceLMs()	119
6.28.2.16 getStemSourcePPLs()	119
6.28.2.17 getStemTargetCorps()	120
6.28.2.18 getStemTargetLMs()	120
6.28.2.19 getStemTargetPPLs()	120
6.28.2.20 getStemVocabs()	121
6.28.2.21 getTargetCorps()	121
6.28.2.22 getTargetLMs()	122
6.28.2.23 getTargetPPLs()	122
6.28.2.24 getVocabs()	123

6.28.2.25 <code>getWeightsFile()</code>	123
6.28.2.26 <code>getXenResult()</code>	124
6.29 <code>double_conversion::StringBuilder</code> Class Reference	125
6.29.1 Constructor & Destructor Documentation	125
6.29.1.1 <code>StringBuilder(char *buffer, int size)</code>	125
6.29.1.2 <code>~StringBuilder()</code>	125
6.29.2 Member Function Documentation	125
6.29.2.1 <code>AddCharacter(char c)</code>	125
6.29.2.2 <code>AddPadding(char c, int count)</code>	125
6.29.2.3 <code>AddString(const char *s)</code>	125
6.29.2.4 <code>AddSubstring(const char *s, int n)</code>	126
6.29.2.5 <code>Finalize()</code>	126
6.29.2.6 <code>position() const</code>	126
6.29.2.7 <code>Reset()</code>	126
6.29.2.8 <code>size() const</code>	126
6.30 <code>double_conversion::StringToDoubleConverter</code> Class Reference	126
6.30.1 Member Enumeration Documentation	126
6.30.1.1 Flags	126
6.30.2 Constructor & Destructor Documentation	127
6.30.2.1 <code>StringToDoubleConverter(int flags, double empty_string_value, double junk_← string_value, const char *infinity_symbol, const char *nan_symbol)</code>	127
6.30.3 Member Function Documentation	127
6.30.3.1 <code>StringToDouble(const char *buffer, int length, int *processed_characters_count) const</code>	127
6.30.3.2 <code>StringToFloat(const char *buffer, int length, int *processed_characters_count) const</code>	127
6.31 <code>TxtStats</code> Struct Reference	127
6.31.1 Member Data Documentation	127
6.31.1.1 <code>numoov</code>	127
6.31.1.2 <code>numsentences</code>	127
6.31.1.3 <code>numwords</code>	127
6.31.1.4 <code>prob</code>	127

6.31.1.5 zeroprobs	127
6.32 double_conversion::Vector< T > Class Template Reference	128
6.32.1 Constructor & Destructor Documentation	128
6.32.1.1 Vector()	128
6.32.1.2 Vector(T *data, int length)	128
6.32.2 Member Function Documentation	128
6.32.2.1 first()	128
6.32.2.2 is_empty() const	128
6.32.2.3 last()	128
6.32.2.4 length() const	128
6.32.2.5 operator[](int index) const	128
6.32.2.6 start() const	128
6.32.2.7 SubVector(int from, int to)	129
6.33 VocabPair Class Reference	129
6.33.1 Detailed Description	129
6.33.2 Constructor & Destructor Documentation	129
6.33.2.1 VocabPair()	129
6.33.2.2 ~VocabPair()	129
6.33.3 Member Function Documentation	129
6.33.3.1 getPtrSourceVoc() const	129
6.33.3.2 getPtrTargetVoc() const	130
6.34 Wfile Class Reference	130
6.34.1 Detailed Description	130
6.34.2 Constructor & Destructor Documentation	130
6.34.2.1 Wfile()	130
6.34.2.2 ~Wfile()	130
6.34.3 Member Function Documentation	131
6.34.3.1 getSize() const	131
6.34.3.2 getWeight(int n)	131
6.34.3.3 initialize(boost::shared_ptr< XenFile > ptrFile)	131

6.35 XenCommon::XenCEption Struct Reference	132
6.35.1 Detailed Description	133
6.35.2 Constructor & Destructor Documentation	133
6.35.2.1 XenCEption(std::string ss)	133
6.35.2.2 ~XenCEption()	133
6.35.3 Member Function Documentation	133
6.35.3.1 what() const	133
6.35.4 Member Data Documentation	133
6.35.4.1 s	133
6.36 XenFile Class Reference	134
6.36.1 Detailed Description	134
6.36.2 Constructor & Destructor Documentation	134
6.36.2.1 XenFile()	134
6.36.2.2 ~XenFile()	134
6.36.3 Member Function Documentation	135
6.36.3.1 getDirName() const	135
6.36.3.2 getExt()	135
6.36.3.3 getFileName() const	135
6.36.3.4 getFullPath() const	135
6.36.3.5 getPrefix()	135
6.36.3.6 initialize(std::string name)	135
6.36.3.7 isGZ() const	136
6.37 XenIO Class Reference	136
6.37.1 Detailed Description	137
6.37.2 Member Function Documentation	137
6.37.2.1 cleanCorpusBi(boost::shared_ptr< Corpus > ptrCorpSource, boost::shared_ptr< Corpus > ptrCorpTarget, boost::shared_ptr< Score > ptrScore)	137
6.37.2.2 cleanCorpusMono(boost::shared_ptr< Corpus > ptrCorp, boost::shared_ptr< Score > ptrScore)	138
6.37.2.3 delFile(std::string fileName)	138
6.37.2.4 dumpSimilarity(boost::shared_ptr< Corpus > ptrCorp, boost::shared_ptr< Similarity > ptrSim)	139

6.37.2.5	read(boost::shared_ptr< XenFile > ptrFile)	139
6.37.2.6	readDist(std::string distFile)	140
6.37.2.7	writeBiOutput(boost::shared_ptr< Corpus > ptrCorpSource, boost::shared_ptr< Corpus > ptrCorpTarget, boost::shared_ptr< Score > ptrScore)	141
6.37.2.8	writeEval(boost::shared_ptr< EvalMap > ptrEvalMap, std::string distName) . . .	142
6.37.2.9	writeMonoOutput(boost::shared_ptr< Corpus > ptrCorp, boost::shared_ptr< Score > ptrScore)	143
6.37.2.10	writeNewPT(boost::shared_ptr< PhraseTable > ptrPT, boost::shared_ptr< Score > ptrScore)	144
6.37.2.11	writeSourcePhrases(boost::shared_ptr< PhraseTable > ptrPT)	145
6.37.2.12	writeTargetPhrases(boost::shared_ptr< PhraseTable > ptrPT)	146
6.37.2.13	writeVocab(std::map< std::string, int > voc, std::string fileName)	146
6.37.2.14	writeXRpart(boost::shared_ptr< XenResult > ptrXR, int pc, std::string fileName="")	147
6.38	XenLMken Class Reference	148
6.38.1	Detailed Description	148
6.38.2	Constructor & Destructor Documentation	148
6.38.2.1	XenLMken()	148
6.38.2.2	~XenLMken()	149
6.38.3	Member Function Documentation	149
6.38.3.1	createLM()	149
6.38.3.2	getDocumentStats(boost::shared_ptr< Corpus > c)	150
6.38.3.3	getFileName() const	151
6.38.3.4	getSentenceStats(std::string sent)	151
6.38.3.5	initialize(boost::shared_ptr< Corpus > ptrCorp, boost::shared_ptr< XenVocab > ptrVoc)	152
6.38.3.6	initialize(boost::shared_ptr< XenFile > ptrFile, boost::shared_ptr< XenVocab > ptrVoc)	152
6.38.3.7	initialize(boost::shared_ptr< XenResult > ptrXenRes, boost::shared_ptr< XenVocab > ptrVoc, int pc, std::string name="")	153
6.38.3.8	loadLM()	154
6.39	XenOption Class Reference	154
6.39.1	Detailed Description	157

6.39.2 Member Function Documentation	157
6.39.2.1 deleteInstance()	157
6.39.2.2 getBp() const	157
6.39.2.3 getDev() const	158
6.39.2.4 getEval() const	158
6.39.2.5 getExclOOVs() const	158
6.39.2.6 getFullVocab() const	159
6.39.2.7 getInPTable() const	159
6.39.2.8 getInSData() const	160
6.39.2.9 getInSLM() const	160
6.39.2.10 getInSStem() const	161
6.39.2.11 getInstance()	161
6.39.2.12 getInstance(LPOptions opt)	162
6.39.2.13 getInTData() const	163
6.39.2.14 getInTLM() const	163
6.39.2.15 getInTStem() const	164
6.39.2.16 getInv() const	164
6.39.2.17 getLocal() const	165
6.39.2.18 getLog() const	165
6.39.2.19 getMaxEvalPC() const	165
6.39.2.20 getMean() const	166
6.39.2.21 getMemPc() const	166
6.39.2.22 getMinBlk() const	167
6.39.2.23 getMode() const	167
6.39.2.24 getMono() const	167
6.39.2.25 getName() const	168
6.39.2.26 getOrder() const	168
6.39.2.27 getOutName() const	169
6.39.2.28 getOutPTable() const	169
6.39.2.29 getOutSData() const	170

6.39.2.30 getOutSLM() const	170
6.39.2.31 getOutSStem() const	171
6.39.2.32 getOutTData() const	171
6.39.2.33 getOutTLM() const	172
6.39.2.34 getOutTStem() const	172
6.39.2.35 getRev() const	172
6.39.2.36 getSampleSize() const	173
6.39.2.37 getSim() const	173
6.39.2.38 getSimOnly() const	174
6.39.2.39 getSLang() const	174
6.39.2.40 getSortBlk() const	175
6.39.2.41 getSortOnly() const	176
6.39.2.42 getStem() const	176
6.39.2.43 getStep() const	177
6.39.2.44 getSVocab() const	177
6.39.2.45 getTemp() const	178
6.39.2.46 getThreads() const	178
6.39.2.47 getTLang() const	178
6.39.2.48 getTVocab() const	179
6.39.2.49 getVecSize() const	180
6.39.2.50 getWFile() const	180
6.39.2.51 setSampleSize(int size)	180
6.39.2.52 setStep(int step)	181
6.40 XenResult Class Reference	181
6.40.1 Detailed Description	182
6.40.2 Constructor & Destructor Documentation	182
6.40.2.1 XenResult()	182
6.40.2.2 ~XenResult()	182
6.40.3 Member Function Documentation	182
6.40.3.1 getSize() const	182

6.40.3.2	getSortedText() const	182
6.40.3.3	getTextLine(int n)	182
6.40.3.4	getXenFile() const	183
6.40.3.5	initialize(boost::shared_ptr< XenFile > ptrFile)	183
6.41	XenVocab Class Reference	183
6.41.1	Detailed Description	184
6.41.2	Constructor & Destructor Documentation	184
6.41.2.1	XenVocab()	184
6.41.2.2	~XenVocab()	184
6.41.3	Member Function Documentation	184
6.41.3.1	getSize() const	184
6.41.3.2	getXenFile() const	185
6.41.3.3	getXenVocab() const	185
6.41.3.4	initialize(boost::shared_ptr< XenFile > ptrFile)	185
6.41.3.5	initialize(boost::shared_ptr< Corpus > ptrCorp)	185
6.41.3.6	initialize(boost::shared_ptr< Corpus > ptrInCorp, boost::shared_ptr< Corpus > ptrOutCorp)	186
6.41.3.7	initialize(boost::shared_ptr< XenResult > ptrXenRes)	186
7	File Documentation	187
7.1	include/corpus.h File Reference	187
7.1.1	Detailed Description	188
7.2	include/eval.h File Reference	188
7.2.1	Detailed Description	189
7.2.2	Typedef Documentation	190
7.2.2.1	EvalMap	190
7.2.3	Function Documentation	190
7.2.3.1	taskEval(int pc, boost::shared_ptr< Corpus > ptrCorp, boost::shared_ptr< XenVocab > ptrVoc, boost::shared_ptr< Corpus > ptrDevCorp, boost::shared_ptr< EvalMap > ptrDist)	190
7.3	include/kenlm/util/double-conversion/bignum-dtoa.h File Reference	191
7.4	include/kenlm/util/double-conversion/bignum.h File Reference	191

7.5	include/kenlm/util/double-conversion/cached-powers.h File Reference	192
7.6	include/kenlm/util/double-conversion/diy-fp.h File Reference	193
7.7	include/kenlm/util/double-conversion/double-conversion.h File Reference	195
7.8	include/kenlm/util/double-conversion/fast-dtoa.h File Reference	195
7.9	include/kenlm/util/double-conversion/fixed-dtoa.h File Reference	197
7.10	include/kenlm/util/double-conversion/ieee.h File Reference	197
7.11	include/kenlm/util/double-conversion/strtod.h File Reference	198
7.12	include/kenlm/util/double-conversion/utils.h File Reference	199
7.12.1	Macro Definition Documentation	201
7.12.1.1	ARRAY_SIZE	201
7.12.1.2	ASSERT	201
7.12.1.3	DISALLOW_COPY_AND_ASSIGN	201
7.12.1.4	DISALLOW_IMPLICIT_CONSTRUCTORS	201
7.12.1.5	UINT64_2PART_C	201
7.12.1.6	UNIMPLEMENTED	201
7.12.1.7	UNREACHABLE	201
7.13	include/mode.h File Reference	201
7.13.1	Detailed Description	202
7.14	include/modes/biXEntropy.h File Reference	202
7.14.1	Detailed Description	203
7.15	include/modes/monoXEntropy.h File Reference	203
7.15.1	Detailed Description	204
7.16	include/modes/ptScoring.h File Reference	205
7.16.1	Detailed Description	205
7.17	include/modes/simplePPL.h File Reference	206
7.17.1	Detailed Description	206
7.18	include/phrasetable.h File Reference	207
7.18.1	Detailed Description	207
7.19	include/ppl.h File Reference	208
7.19.1	Detailed Description	209

7.19.2 Macro Definition Documentation	209
7.19.2.1 M_LN10	209
7.19.3 Function Documentation	209
7.19.3.1 taskCalcPPL(int numLine, std::string line, boost::shared_ptr< std::vector< double > > ptrPPL, boost::shared_ptr< XenLMken > ptrLM)	209
7.20 include/score.h File Reference	210
7.20.1 Detailed Description	211
7.21 include/similarity.h File Reference	211
7.21.1 Detailed Description	213
7.21.2 Typedef Documentation	213
7.21.2.1 SimMap	213
7.22 include/sourcephrase.h File Reference	213
7.22.1 Detailed Description	214
7.23 include/utils/common.h File Reference	214
7.23.1 Detailed Description	216
7.23.2 Typedef Documentation	217
7.23.2.1 LPOptions	217
7.23.2.2 Options	217
7.24 include/utils/StaticData.h File Reference	217
7.24.1 Detailed Description	218
7.24.2 Macro Definition Documentation	218
7.24.2.1 STATICDATA_H_	218
7.25 include/utils/xenio.h File Reference	218
7.25.1 Detailed Description	219
7.25.2 Macro Definition Documentation	220
7.25.2.1 XENIO_H_	220
7.25.3 Typedef Documentation	220
7.25.3.1 EvalMap	220
7.26 include/wfile.h File Reference	220
7.26.1 Detailed Description	221
7.27 include/Xen.h File Reference	221

7.27.1 Detailed Description	222
7.27.2 Function Documentation	222
7.27.2.1 getOutName(XenOption *opt)	222
7.27.2.2 main(int argc, char *argv[])	224
7.27.2.3 sanityCheck(XenOption *opt)	225
7.28 include/xenfile.h File Reference	227
7.28.1 Detailed Description	228
7.29 include/XenLMken.h File Reference	228
7.29.1 Detailed Description	229
7.29.2 Macro Definition Documentation	230
7.29.2.1 KENLM_MAX_ORDER	230
7.30 include/xenoption.h File Reference	230
7.30.1 Detailed Description	231
7.31 include/xenresult.h File Reference	231
7.31.1 Detailed Description	232
7.32 include/xenvocab.h File Reference	232
7.32.1 Detailed Description	233
7.33 src/corpus.cpp File Reference	233
7.33.1 Detailed Description	234
7.34 src/eval.cpp File Reference	234
7.34.1 Detailed Description	234
7.34.2 Function Documentation	234
7.34.2.1 taskEval(int pc, boost::shared_ptr< Corpus > ptrCorp, boost::shared_ptr< XenVocab > ptrVoc, boost::shared_ptr< Corpus > ptrDevCorp, boost::shared_ptr< EvalMap > ptrDist)	234
7.35 src/mode.cpp File Reference	235
7.35.1 Detailed Description	236
7.36 src/modes/biXEntropy.cpp File Reference	236
7.36.1 Detailed Description	237
7.37 src/modes/monoXEntropy.cpp File Reference	237
7.37.1 Detailed Description	237

7.38 src/modes/ptScoring.cpp File Reference	238
7.38.1 Detailed Description	238
7.39 src/modes/simplePPL.cpp File Reference	238
7.39.1 Detailed Description	239
7.40 src/phrasetable.cpp File Reference	239
7.40.1 Detailed Description	239
7.41 src/ppl.cpp File Reference	240
7.41.1 Detailed Description	240
7.41.2 Function Documentation	240
7.41.2.1 taskCalcPPL(int numLine, std::string line, boost::shared_ptr< std::vector< double > > ptrPPL, boost::shared_ptr< XenLMken > ptrLM)	240
7.41.3 Variable Documentation	241
7.41.3.1 randy	241
7.42 src(score.cpp File Reference	241
7.42.1 Detailed Description	242
7.43 src/similarity.cpp File Reference	242
7.43.1 Detailed Description	243
7.44 src/sourcephrase.cpp File Reference	243
7.44.1 Detailed Description	243
7.45 src/utils/StaticData.cpp File Reference	244
7.45.1 Detailed Description	244
7.46 src/utils/xenio.cpp File Reference	244
7.46.1 Detailed Description	245
7.47 src/wfile.cpp File Reference	245
7.47.1 Detailed Description	245
7.48 src/Xen.cpp File Reference	246
7.48.1 Detailed Description	246
7.48.2 Function Documentation	246
7.48.2.1 getOutName(XenOption *opt)	246
7.48.2.2 main(int argc, char *argv[])	248
7.48.2.3 sanityCheck(XenOption *opt)	249
7.48.3 Variable Documentation	251
7.48.3.1 version	251
7.49 src/xenfile.cpp File Reference	251
7.49.1 Detailed Description	251
7.50 src/XenLMken.cpp File Reference	252
7.50.1 Detailed Description	252
7.51 src/xenoption.cpp File Reference	252
7.51.1 Detailed Description	253
7.52 src/xenresult.cpp File Reference	253
7.52.1 Detailed Description	253
7.53 src/xenvocab.cpp File Reference	254
7.53.1 Detailed Description	254

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

double_conversion	11
XenCommon	
Namespace containing all the common functions of XenC	14

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

_Options	21
double_conversion::Bignum	28
Corpus	36
CorpusPair	40
double_conversion::DiyFp	41
double_conversion::Double	46
double_conversion::DoubleToStringConverter	58
Eval	61
exception	
XenCommon::XenCEption	132
LMPair	65
MeanLMPair	66
MeanPPLPair	68
Mode	69
BiXEntropy	35
MonoXEntropy	72
PTScoring	89
SimplePPL	98
PhraseTable	76
PhraseTablePair	82
double_conversion::PowersOfTenCache	83
PPL	84
PPLPair	87
Score	92
ScoreHolder	94
Similarity	96
double_conversion::Single	101
SourcePhrase	105
XenCommon::Splitter	109
StaticData	111
double_conversion::StringBuilder	125
double_conversion::StringtoDoubleConverter	126
TxtStats	127
double_conversion::Vector< T >	128
double_conversion::Vector< char >	128

double_conversion::Vector< Chunk >	128
VocabPair	129
Wfile	130
XenFile	134
XenIO	136
XenLMken	148
XenOption	154
XenResult	181
XenVocab	183

Chapter 3

Class Index

3.1 Class List

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

_Options	
XenC options structure	21
double_conversion::Bignum	28
BiXEntropy	
Filtering mode 3: bilingual cross-entropy	35
Corpus	
Corpus-related functionalities	36
CorpusPair	
Tiny class holding two related Corpus	40
double_conversion::DiyFp	41
double_conversion::Double	46
double_conversion::DoubleToStringConverter	58
Eval	
Evaluation system	61
LMPair	
Tiny class holding two related language models	65
MeanLMPair	
Tiny class holding two additional LMs for mean scoring feature	66
MeanPPLPair	
Tiny class holding two additional PPL objects for mean scoring feature	68
Mode	
Filtering modes interface	69
MonoXEntropy	
Filtering mode 2: monolingual cross-entropy	72
PhraseTable	
Class handling phrase-table related functionalities	76
PhraseTablePair	
Tiny class holding the two phrase-tables	82
double_conversion::PowersOfTenCache	83
PPL	
Perplexity/Cross-entropy computations	84
PPLPair	
Tiny class holding two related PPL objects	87
PTScoring	
Filtering mode 4: phrase-table cross-entropy	89

Score	
Class holding the XenC scores representation	92
ScoreHolder	
Tiny class holding three Score objects (global scores, similarity, cross-entropy)	94
Similarity	
Class taking care of all the similarity measure computations	96
SimplePPL	
Filtering mode 1: simple perplexity	98
double_conversion::Single	
.	101
SourcePhrase	
Class holding a merged source phrase and all associated data	105
XenCommon::Splitter	
Class defining a splitter	109
StaticData	
Class gathering all data used and generated by XenC	111
double_conversion::StringBuilder	
.	125
double_conversion::StringToDoubleConverter	
.	126
TxtStats	
.	127
double_conversion::Vector< T >	
.	128
VocabPair	
Tiny class holding the two vocabularies	129
Wfile	
Class handling a file with values intended at weighting XenC scores	130
XenCommon::XenCEption	
XenC exception structure	132
XenFile	
Class providing some basic functions around files	134
XenIO	
Class handling all input/output operations of XenC	136
XenLMken	
Class handling KenLM estimation, loading, querying..	148
XenOption	
Singleton class handling XenC options accessors/mutators	154
XenResult	
Class handling a XenC sorted result file for evaluation/best point	181
XenVocab	
Class handling a XenC vocabulary	183

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	187
include/eval.h	Class handling evaluation system	188
include/mode.h	Abstract class defining the filtering modes architecture	201
include/phrasetable.h	Class handling phrase-table related functionalities	207
include/ppl.h	Class handling the perplexity/cross-entropy computations	208
include/score.h	Class holding the XenC scores representation	210
include/similarity.h	Class taking care of all the similarity measure computations	211
include/sourcephrase.h	Class holding a merged source phrase and all associated data	213
include/wfile.h	Class handling a file with values intended at weighting XenC scores	220
include/Xen.h	Main file of XenC, controls execution	221
include/xenfile.h	Class providing some basic functions around files	227
include/XenLMken.h	Class handling KenLM estimation, loading, querying..	228
include/xenoption.h	Singleton class handling XenC options accessors/mutators	230
include/xenresult.h	Class handling a XenC sorted result file for evaluation/best point	231
include/xenvocab.h	Class handling a XenC vocabulary	232
include/kenlm/util/double-conversion/bignum-dtoa.h	191
include/kenlm/util/double-conversion/bignum.h	191
include/kenlm/util/double-conversion/cached-powers.h	192
include/kenlm/util/double-conversion/diy-fp.h	193
include/kenlm/util/double-conversion/double-conversion.h	195

include/kenlm/util/double-conversion/fast-dtoa.h	195
include/kenlm/util/double-conversion/fixed-dtoa.h	197
include/kenlm/util/double-conversion/ieee.h	197
include/kenlm/util/double-conversion/strtod.h	198
include/kenlm/util/double-conversion/utils.h	199
include/modes/biXEntropy.h	
Derived class to handle filtering mode 3: bilingual cross-entropy	202
include/modes/monoXEntropy.h	
Derived class to handle filtering mode 2: monolingual cross-entropy	203
include/modes/ptScoring.h	
Derived class to handle filtering mode 4: phrase-table cross-entropy	205
include/modes/simplePPL.h	
Derived class to handle filtering mode 1: simple perplexity	206
include/utils/common.h	
File containing all common classes/structures/functions of many classes of XenC	214
include/utils/StaticData.h	
File handling all data objects used by XenC in a static way	217
include/utils/xenio.h	
Class handling all input/output operations of XenC	218
src/corpus.cpp	
Class handling corpus-related functionalities	233
src/eval.cpp	
Class handling evaluation system	234
src/mode.cpp	
Abstract class defining the filtering modes architecture	235
src/phrasetable.cpp	
Class handling phrase-table related functionalities	239
src/ppl.cpp	
Class handling the perplexity/cross-entropy computations	240
src(score.cpp	
Class holding the XenC scores representation	241
src/similarity.cpp	
Class taking care of all the similarity measure computations	242
src/sourcephrase.cpp	
Class holding a merged source phrase and all associated data	243
src/wfile.cpp	
Class handling a file with values intended at weighting XenC scores	245
src/Xen.cpp	
Main file of XenC, controls execution	246
src/xenfile.cpp	
Class providing some basic functions around files	251
src/XenLMken.cpp	
Class handling KenLM estimation, loading, querying..	252
src/xenoption.cpp	
Singleton class handling XenC options accessors/mutators	252
src/xenresult.cpp	
Class handling a XenC sorted result file for evaluation/best point	253
src/xenvocab.cpp	
Class handling a XenC vocabulary	254
src/modes/biXEntropy.cpp	
Derived class to handle filtering mode 3: bilingual cross-entropy	236
src/modes/monoXEntropy.cpp	
Derived class to handle filtering mode 2: monolingual cross-entropy	237
src/modes/ptScoring.cpp	
Derived class to handle filtering mode 4: phrase-table cross-entropy	238
src/modes/simplePPL.cpp	
Derived class to handle filtering mode 1: simple perplexity	238

src/utils/ StaticData.cpp	File handling all data objects used by XenC in a static way	244
src/utils/ xenio.cpp	Class handling all input/output operations of XenC	244

Chapter 5

Namespace Documentation

5.1 double_conversion Namespace Reference

Classes

- class [Bignum](#)
- class [DiyFp](#)
- class [Double](#)
- class [DoubleToStringConverter](#)
- class [PowersOfTenCache](#)
- class [Single](#)
- class [StringBuilder](#)
- class [StringtoDoubleConverter](#)
- class [Vector](#)

Enumerations

- enum [BignumDtoaMode](#) { [BIGNUM_DTOA_SHORTEST](#), [BIGNUM_DTOA_SHORTEST_SINGLE](#), [BIGNUM_DTOA_FIXED](#), [BIGNUM_DTOA_PRECISION](#) }
- enum [FastDtoaMode](#) { [FAST_DTOA_SHORTEST](#), [FAST_DTOA_SHORTEST_SINGLE](#), [FAST_DTOA_PRECISION](#) }

Functions

- void [BignumDtoa](#) (double v, [BignumDtoaMode](#) mode, int requested_digits, [Vector<char>](#) buffer, int *length, int *point)
- bool [FastDtoa](#) (double d, [FastDtoaMode](#) mode, int requested_digits, [Vector<char>](#) buffer, int *length, int *decimal_point)
- bool [FastFixedDtoa](#) (double v, int fractional_count, [Vector<char>](#) buffer, int *length, int *decimal_point)
- static uint64_t [double_to_uint64](#) (double d)
- static double [uint64_to_double](#) (uint64_t d64)
- static uint32_t [float_to_uint32](#) (float f)
- static float [uint32_to_float](#) (uint32_t d32)
- double [Strtod](#) ([Vector<const char>](#) buffer, int exponent)
- float [Strtob](#) ([Vector<const char>](#) buffer, int exponent)

- template<typename T >
static T **Max** (T a, T b)
- template<typename T >
static T **Min** (T a, T b)
- int **StrLength** (const char *string)
- template<class Dest , class Source >
Dest **BitCast** (const Source &source)
- template<class Dest , class Source >
Dest **BitCast** (Source *source)

Variables

- static const int **kFastDtoaMaximalLength** = 17
- static const int **kFastDtoaMaximalSingleLength** = 9
- static const int **kCharSize** = sizeof(char)

5.1.1 Enumeration Type Documentation

5.1.1.1 enum double_conversion::BignumDtoaMode

Enumerator

BIGNUM_DTOA_SHORTEST
BIGNUM_DTOA_SHORTEST_SINGLE
BIGNUM_DTOA_FIXED
BIGNUM_DTOA_PRECISION

5.1.1.2 enum double_conversion::FastDtoaMode

Enumerator

FAST_DTOA_SHORTEST
FAST_DTOA_SHORTEST_SINGLE
FAST_DTOA_PRECISION

5.1.2 Function Documentation

5.1.2.1 void double_conversion::BignumDtoa (double v, BignumDtoaMode mode, int requested_digits, Vector< char > buffer, int * length, int * point)

5.1.2.2 template<class Dest , class Source > Dest double_conversion::BitCast (const Source & source) [inline]

5.1.2.3 template<class Dest , class Source > Dest double_conversion::BitCast (Source * source) [inline]

5.1.2.4 static uint64_t double_conversion::double_to_uint64 (double d) [static]

5.1.2.5 `bool double_conversion::FastDtoa (double d, FastDtoaMode mode, int requested_digits, Vector< char > buffer, int * length, int * decimal_point)`

5.1.2.6 `bool double_conversion::FastFixedDtoa (double v, int fractional_count, Vector< char > buffer, int * length, int * decimal_point)`

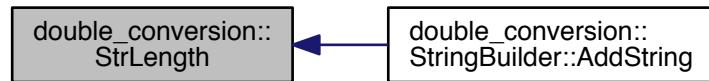
5.1.2.7 `static uint32_t double_conversion::float_to_uint32 (float f) [static]`

5.1.2.8 `template<typename T> static T double_conversion::Max (T a, T b) [static]`

5.1.2.9 `template<typename T> static T double_conversion::Min (T a, T b) [static]`

5.1.2.10 `int double_conversion::StrLength (const char * string) [inline]`

Here is the caller graph for this function:

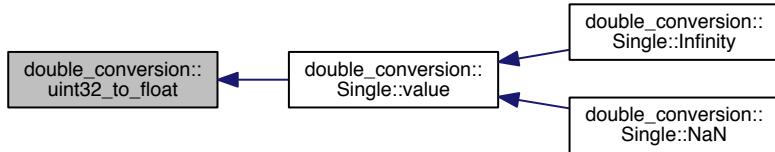


5.1.2.11 `double double_conversion::Strtod (Vector< const char > buffer, int exponent)`

5.1.2.12 `float double_conversion::Strtof (Vector< const char > buffer, int exponent)`

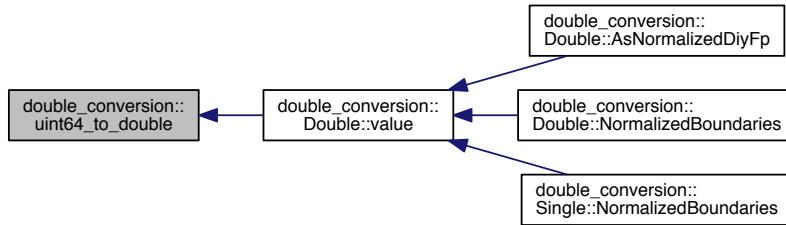
5.1.2.13 `static float double_conversion::uint32_to_float (uint32_t d32) [static]`

Here is the caller graph for this function:



5.1.2.14 static double double_conversion::uint64_to_double (uint64_t d64) [static]

Here is the caller graph for this function:



5.1.3 Variable Documentation

5.1.3.1 const int double_conversion::kCharSize = sizeof(char) [static]

5.1.3.2 const int double_conversion::kFastDtoaMaximalLength = 17 [static]

5.1.3.3 const int double_conversion::kFastDtoaMaximalSingleLength = 9 [static]

5.2 XenCommon Namespace Reference

Namespace containing all the common functions of XenC.

Classes

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

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 [tolnt](#) (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 an 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.

5.2.1 Detailed Description

Namespace containing all the common functions of XenC.

5.2.2 Function Documentation

5.2.2.1 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.

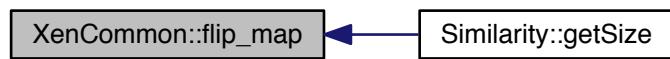
Template Parameters

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

Returns

flipped multimap with descending order

Here is the caller graph for this function:



5.2.2.2 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>

5.2.2.3 std::string XenCommon::getStdoutFromCommand(std::string cmd) [inline]

Executes a system command and returns the output.

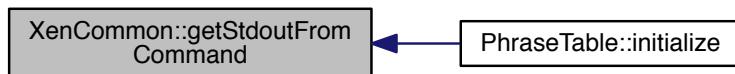
Parameters

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

Returns

the output of the executed command

Here is the caller graph for this function:

**5.2.2.4 template<typename T > double XenCommon::toDouble(const T & Value)**

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

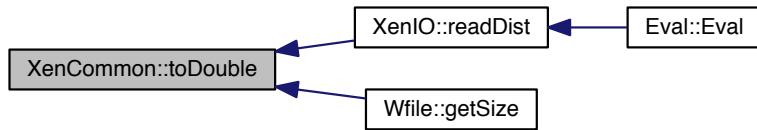
Template Parameters

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

Returns

string containing the converted value

Here is the caller graph for this function:



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

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

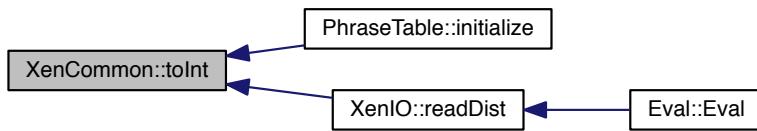
Template Parameters

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

Returns

string containing the converted value

Here is the caller graph for this function:



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

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

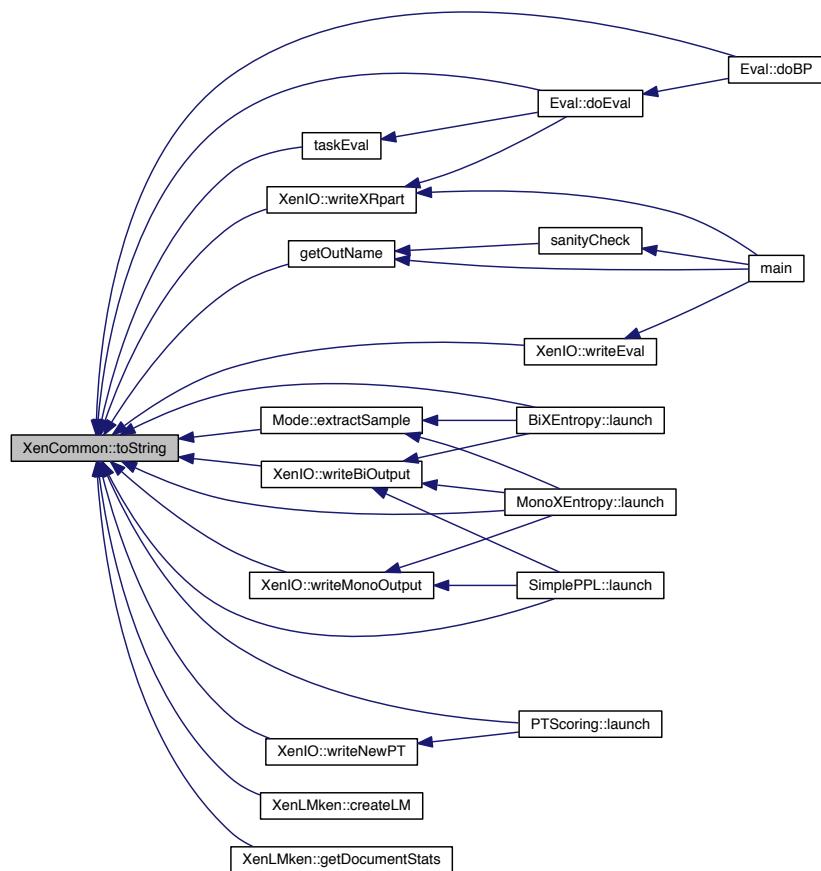
Template Parameters

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

Returns

string containing the converted value

Here is the caller graph for this function:



5.2.2.7 template<typename T > std::string XenCommon::toString0 (const T & Value)

Template converting a value into a string with no precision.

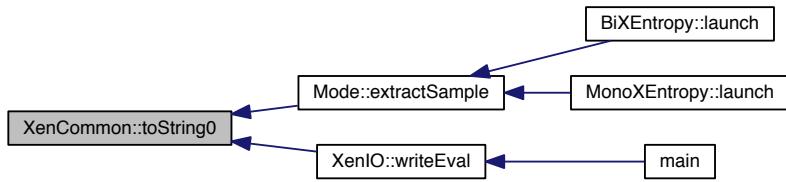
Template Parameters

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

Returns

string containing the converted value

Here is the caller graph for this function:



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

Computes the word count of a string.

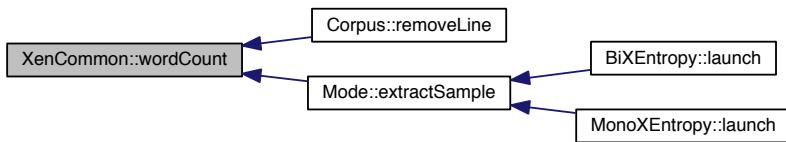
Parameters

<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.
- std::size_t **memPC**
The percentage of system memory to use.
- std::string **temp**
The temporary files location.
- std::size_t **minblk**
The minimum block size.
- std::size_t **sortblk**
The minimum block size for sort.
- bool **exclOOVs**
Indicates if the OOVs must be excluded from PPL computation.
- 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**

- bool `eval`
Indicates local scores computation for phrase table filtering.
- bool `bp`
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)
- int `maxEvalPC`
The maximum eval percentage.

6.1.1 Detailed Description

XenC options structure.

6.1.2 Member Data Documentation

6.1.2.1 bool _Options::bp

Indicates best-point evaluation mode.

6.1.2.2 std::string _Options::dev

The development corpus (for evaluation)

6.1.2.3 bool _Options::eval

Indicates evaluation mode.

6.1.2.4 bool _Options::exclOOVs

Indicates if the OOVs must be excluded from [PPL](#) computation.

6.1.2.5 bool _Options::fullVoc

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

6.1.2.6 std::string _Options::inSData

The in-domain source corpus.

6.1.2.7 std::string _Options::inSLM

The in-domain source language model.

6.1.2.8 std::string _Options::inSStem

The in-domain source stem corpus.

6.1.2.9 std::string _Options::inTData

The in-domain target corpus.

6.1.2.10 std::string _Options::inTLM

The in-domain target language model.

6.1.2.11 int _Options::inToks

The number of in-domain tokens.

6.1.2.12 std::string _Options::inTStem

The in-domain target stem corpus.

6.1.2.13 bool _Options::inv

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

6.1.2.14 std::string _Options::iPTable

The in-domain phrase-table.

6.1.2.15 bool _Options::local

Indicates local scores computation for phrase table filtering.

6.1.2.16 bool _Options::log

Indicates if the weights are given in the log domain.

6.1.2.17 int _Options::maxEvalPC

The maximum eval percentage.

6.1.2.18 bool _Options::mean

Indicates mean computation.

6.1.2.19 std::size_t _Options::memPC

The percentage of system memory to use.

6.1.2.20 std::size_t _Options::minblk

The minimum block size.

6.1.2.21 int _Options::mode

The filtering mode.

6.1.2.22 bool _Options::mono

Indicates if monolingual data is being filtered or not.

6.1.2.23 std::string _Options::name

The program name.

6.1.2.24 std::string _Options::oPTable

The out-of-domain phrase-table.

6.1.2.25 int _Options::order

The order for language models estimation.

6.1.2.26 std::string _Options::outName

The output file name.

6.1.2.27 std::string _Options::outSData

The out-of-domain source corpus.

6.1.2.28 std::string _Options::outSLM

The out-of-domain source language model.

6.1.2.29 std::string _Options::outSStem

The out-of-domain source stem corpus.

6.1.2.30 std::string _Options::outTData

The out-of-domain target corpus.

6.1.2.31 std::string _Options::outTLM

The out-of-domain target language model.

6.1.2.32 int _Options::outToks

The number of out-of-domain tokens.

6.1.2.33 std::string _Options::outTStem

The out-of-domain source stem corpus.

6.1.2.34 int _Options::pc

The current percentage being evaluated.

6.1.2.35 bool _Options::rev

Indicates if a reversed output is requested (descending order)

6.1.2.36 int _Options::sampleSize

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

6.1.2.37 bool _Options::sim

Indicates similarity computation.

6.1.2.38 bool _Options::simOnly

Indicates similarity computation only.

6.1.2.39 std::string _Options::sLang

The source language.

6.1.2.40 std::size_t _Options::sortblk

The minimum block size for sort.

6.1.2.41 bool _Options::sortOnly

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

6.1.2.42 bool _Options::stem

Indicates stem computation.

6.1.2.43 int _Options::step

The step size for evaluation and best-point.

6.1.2.44 std::string _Options::sVocab

The source language vocabulary.

6.1.2.45 std::string _Options::temp

The temporary files location.

6.1.2.46 int _Options::threads

The number of threads.

6.1.2.47 std::string _Options::tLang

The target language.

6.1.2.48 std::string _Options::tVocab

The target language vocabulary.

6.1.2.49 int _Options::vecSize

The vector size for similarity.

6.1.2.50 bool _Options::version

The program version.

6.1.2.51 std::string _Options::wFile

The weight file.

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

- include/utils/common.h

6.2 double_conversion::Bignum Class Reference

```
#include <bignum.h>
```

Public Member Functions

- `Bignum ()`
- `void AssignUInt16 (uint16_t value)`
- `void AssignUInt64 (uint64_t value)`
- `void AssignBignum (const Bignum &other)`
- `void AssignDecimalString (Vector< const char > value)`
- `void AssignHexString (Vector< const char > value)`
- `void AssignPowerUInt16 (uint16_t base, int exponent)`
- `void AddUInt16 (uint16_t operand)`
- `void AddUInt64 (uint64_t operand)`
- `void AddBignum (const Bignum &other)`
- `void SubtractBignum (const Bignum &other)`
- `void Square ()`
- `void ShiftLeft (int shift_amount)`
- `void MultiplyByUInt32 (uint32_t factor)`
- `void MultiplyByUInt64 (uint64_t factor)`
- `void MultiplyByPowerOfTen (int exponent)`
- `void Times10 ()`
- `uint16_t DivideModuloIntBignum (const Bignum &other)`
- `bool ToHexString (char *buffer, int buffer_size) const`

Static Public Member Functions

- `static int Compare (const Bignum &a, const Bignum &b)`
- `static bool Equal (const Bignum &a, const Bignum &b)`
- `static bool LessEqual (const Bignum &a, const Bignum &b)`
- `static bool Less (const Bignum &a, const Bignum &b)`
- `static int PlusCompare (const Bignum &a, const Bignum &b, const Bignum &c)`
- `static bool PlusEqual (const Bignum &a, const Bignum &b, const Bignum &c)`
- `static bool PlusLessEqual (const Bignum &a, const Bignum &b, const Bignum &c)`
- `static bool PlusLess (const Bignum &a, const Bignum &b, const Bignum &c)`

Static Public Attributes

- `static const int kMaxSignificantBits = 3584`

6.2.1 Constructor & Destructor Documentation

6.2.1.1 double_conversion::Bignum::Bignum ()

6.2.2 Member Function Documentation

6.2.2.1 void double_conversion::Bignum::AddBignum (const Bignum & other)

6.2.2.2 void double_conversion::Bignum::AddUInt16 (uint16_t operand)

6.2.2.3 void double_conversion::Bignum::AddUInt64 (uint64_t operand)

6.2.2.4 void double_conversion::Bignum::AssignBignum (const Bignum & other)

6.2.2.5 void double_conversion::Bignum::AssignDecimalString (Vector< const char > value)

6.2.2.6 void double_conversion::Bignum::AssignHexString (Vector< const char > value)

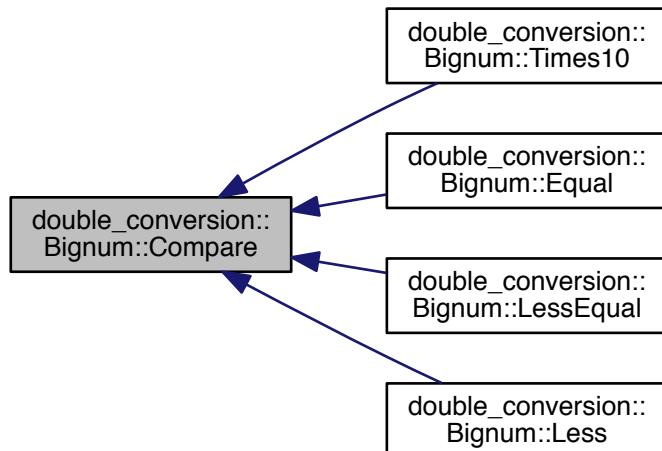
6.2.2.7 void double_conversion::Bignum::AssignPowerUInt16 (uint16_t base, int exponent)

6.2.2.8 void double_conversion::Bignum::AssignUInt16 (uint16_t value)

6.2.2.9 void double_conversion::Bignum::AssignUInt64 (uint64_t value)

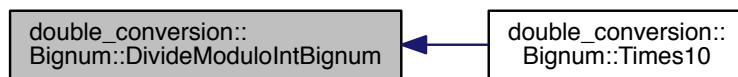
6.2.2.10 static int double_conversion::Bignum::Compare (const Bignum & a, const Bignum & b) [static]

Here is the caller graph for this function:



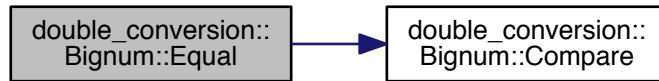
6.2.2.11 uint16_t double_conversion::Bignum::DivideModuloIntBignum (const Bignum & other)

Here is the caller graph for this function:



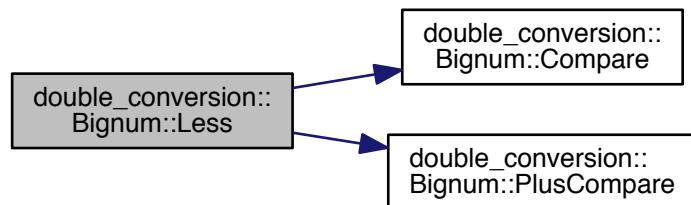
6.2.2.12 static bool double_conversion::Bignum::Equal (const Bignum & a, const Bignum & b) [inline],
[static]

Here is the call graph for this function:



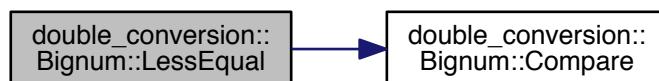
6.2.2.13 static bool double_conversion::Bignum::Less (const Bignum & a, const Bignum & b) [inline],
[static]

Here is the call graph for this function:



6.2.2.14 static bool double_conversion::Bignum::LessEqual (const Bignum & a, const Bignum & b) [inline],
[static]

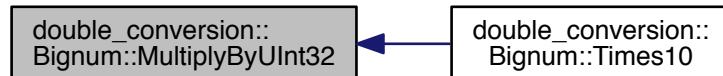
Here is the call graph for this function:



6.2.2.15 void double_conversion::Bignum::MultiplyByPowerOfTen (int exponent)

6.2.2.16 void double_conversion::Bignum::MultiplyByUInt32 (uint32_t factor)

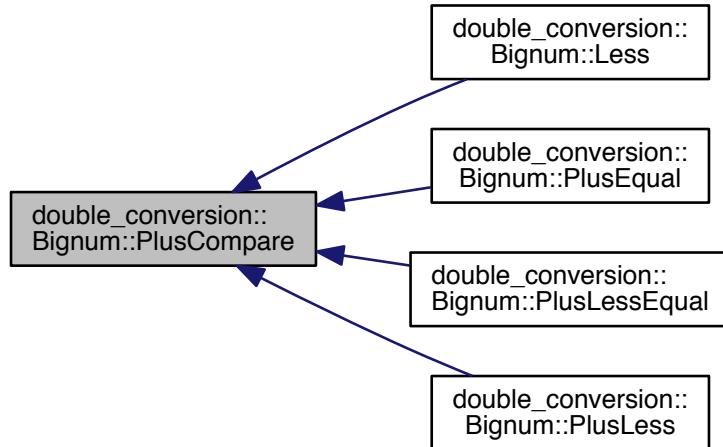
Here is the caller graph for this function:



6.2.2.17 void double_conversion::Bignum::MultiplyByUInt64 (uint64_t factor)

6.2.2.18 static int double_conversion::Bignum::PlusCompare (const Bignum & a, const Bignum & b, const Bignum & c)
[static]

Here is the caller graph for this function:



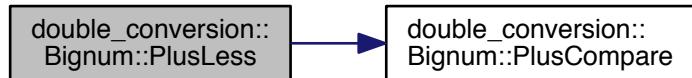
6.2.2.19 static bool double_conversion::Bignum::PlusEqual (const Bignum & a, const Bignum & b, const Bignum & c)
[inline], [static]

Here is the call graph for this function:



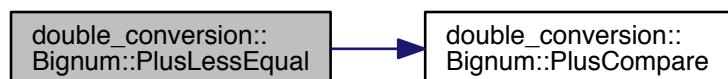
6.2.2.20 static bool double_conversion::Bignum::PlusLess (const Bignum & a, const Bignum & b, const Bignum & c)
[inline], [static]

Here is the call graph for this function:



6.2.2.21 static bool double_conversion::Bignum::PlusLessEqual (const Bignum & a, const Bignum & b, const Bignum & c) [inline], [static]

Here is the call graph for this function:



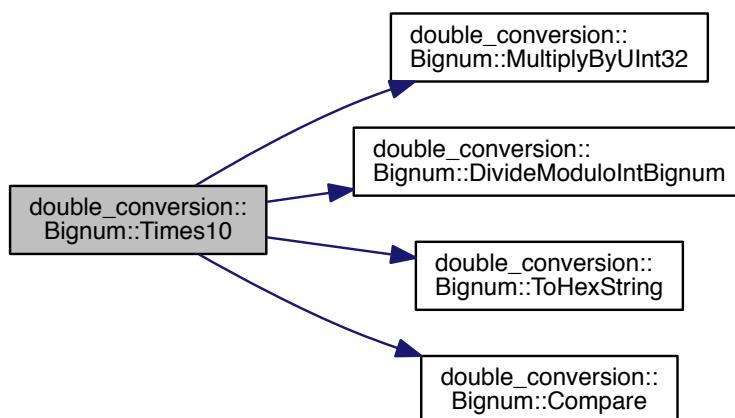
6.2.2.22 void double_conversion::Bignum::ShiftLeft (int *shift_amount*)

6.2.2.23 void double_conversion::Bignum::Square ()

6.2.2.24 void double_conversion::Bignum::SubtractBignum (const Bignum & *other*)

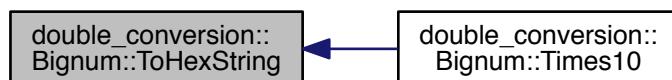
6.2.2.25 void double_conversion::Bignum::Times10 () [inline]

Here is the call graph for this function:



6.2.2.26 bool double_conversion::Bignum::ToHexString (char * *buffer*, int *buffer_size*) const

Here is the caller graph for this function:



6.2.3 Member Data Documentation

6.2.3.1 const int double_conversion::Bignum::kMaxSignificantBits = 3584 [static]

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

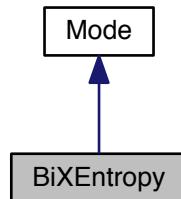
- include/kenlm/util/double-conversion/bignum.h

6.3 BiXEntropy Class Reference

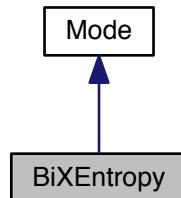
Filtering mode 3: bilingual cross-entropy.

```
#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

6.3.1 Detailed Description

Filtering mode 3: bilingual cross-entropy.

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

6.3.2 Constructor & Destructor Documentation

6.3.2.1 BiXEntropy::BiXEntropy()

Default constructor.

6.3.2.2 BiXEntropy::~BiXEntropy()

Default destructor.

6.3.3 Member Function Documentation

6.3.3.1 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](#)

6.4 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.

6.4.1 Detailed Description

Corpus-related functionalities.

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

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Corpus::Corpus ()

Default constructor.

6.4.2.2 Corpus::~Corpus ()

Default destructor.

6.4.3 Member Function Documentation

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

Accessor to the language of the Corpus.

Returns

string containing the language

6.4.3.2 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

6.4.3.3 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

6.4.3.4 unsigned int Corpus::getSize () const

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

Returns

unsigned int representing the size

6.4.3.5 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:



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

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

Returns

shared pointer to the [XenFile](#)

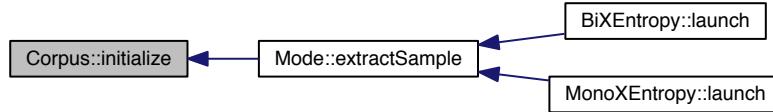
6.4.3.7 `void Corpus::initialize(boost::shared_ptr< XenFile > ptrData, std::string lg)`

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

Parameters

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

Here is the caller graph for this function:

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

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

Parameters

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

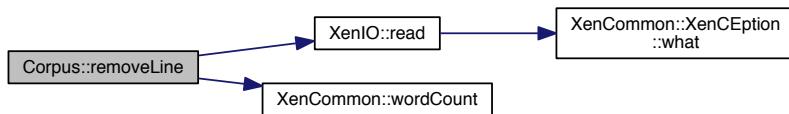
6.4.3.9 `void Corpus::removeLine(int line)`

Put the printing status of a line to false.

Parameters

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

Here is the call graph for this function:



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

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

6.5 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.

6.5.1 Detailed Description

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

6.5.2 Constructor & Destructor Documentation

6.5.2.1 CorpusPair::CorpusPair() [inline]

Default constructor.

6.5.2.2 CorpusPair::~CorpusPair() [inline]

Default destructor.

6.5.3 Member Function Documentation

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

Accessor to the in-domain corpus.

Returns

the in-domain corpus

6.5.3.2 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/utils/StaticData.h

6.6 double_conversion::DiyFp Class Reference

```
#include <diy-fp.h>
```

Public Member Functions

- [DiyFp\(\)](#)
- [DiyFp\(uint64_t f, int e\)](#)
- [void Subtract\(const DiyFp &other\)](#)
- [void Multiply\(const DiyFp &other\)](#)
- [void Normalize\(\)](#)
- [uint64_t f\(\) const](#)
- [int e\(\) const](#)
- [void set_f\(uint64_t new_value\)](#)
- [void set_e\(int new_value\)](#)

Static Public Member Functions

- [static DiyFp Minus\(const DiyFp &a, const DiyFp &b\)](#)
- [static DiyFp Times\(const DiyFp &a, const DiyFp &b\)](#)
- [static DiyFp Normalize\(const DiyFp &a\)](#)

Static Public Attributes

- static const int `kSignificandSize` = 64

6.6.1 Constructor & Destructor Documentation

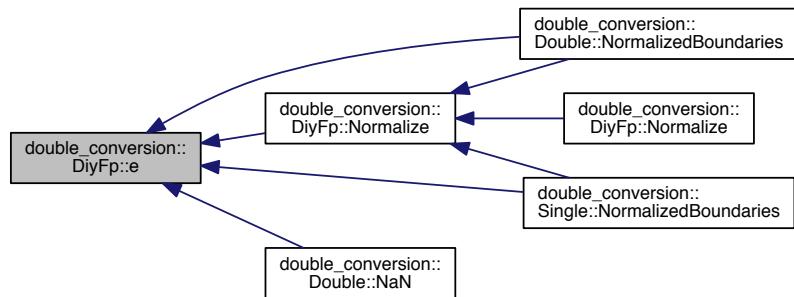
6.6.1.1 `double_conversion::DiyFp::DiyFp() [inline]`

6.6.1.2 `double_conversion::DiyFp::DiyFp(uint64_t f, int e) [inline]`

6.6.2 Member Function Documentation

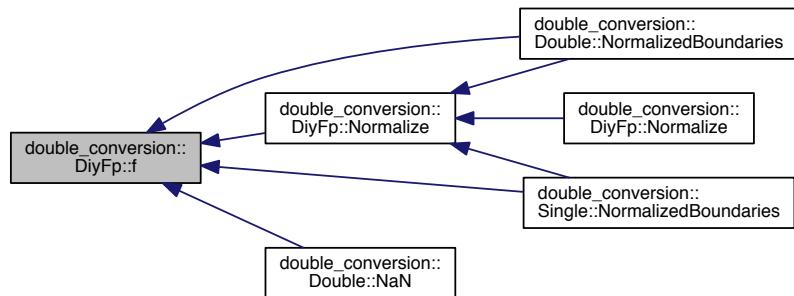
6.6.2.1 `int double_conversion::DiyFp::e() const [inline]`

Here is the caller graph for this function:



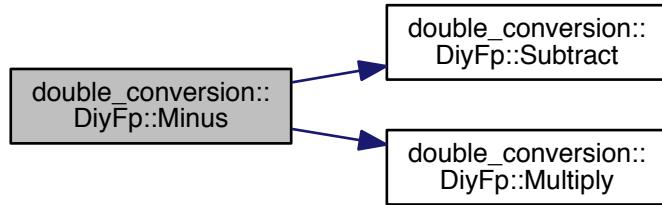
6.6.2.2 `uint64_t double_conversion::DiyFp::f() const [inline]`

Here is the caller graph for this function:



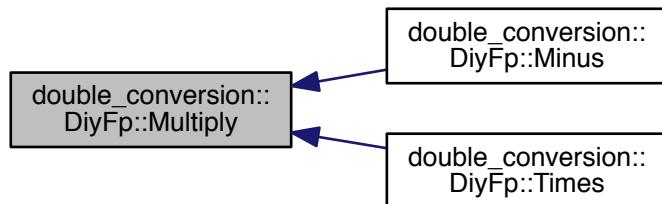
6.6.2.3 static DiyFp double_conversion::DiyFp::Minus (const DiyFp & a, const DiyFp & b) [inline], [static]

Here is the call graph for this function:



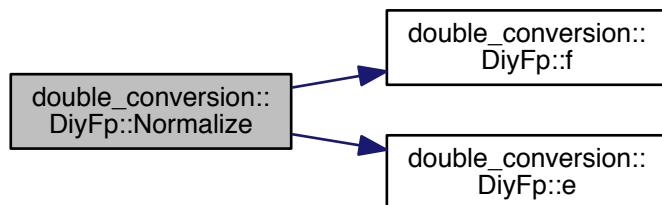
6.6.2.4 void double_conversion::DiyFp::Multiply (const DiyFp & other)

Here is the caller graph for this function:

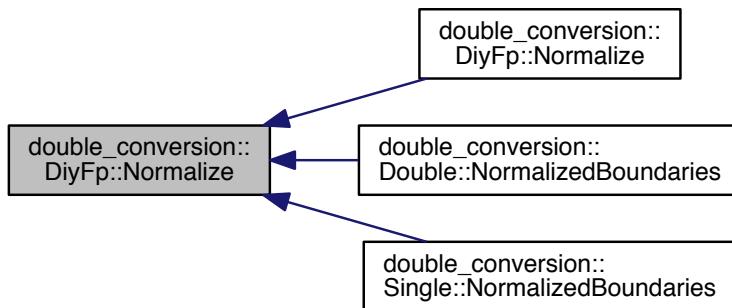


6.6.2.5 void double_conversion::DiyFp::Normalize () [inline]

Here is the call graph for this function:

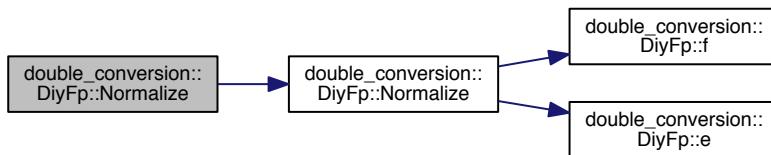


Here is the caller graph for this function:



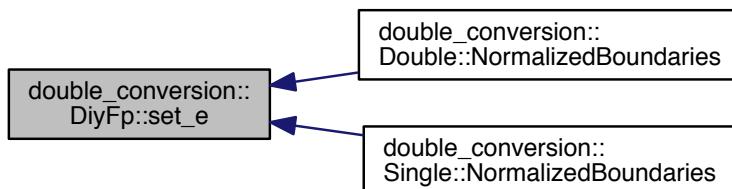
6.6.2.6 static DiyFp double_conversion::DiyFp::Normalize (const DiyFp & a) [inline], [static]

Here is the call graph for this function:



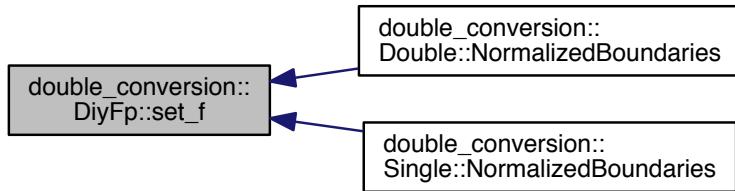
6.6.2.7 void double_conversion::DiyFp::set_e (int new_value) [inline]

Here is the caller graph for this function:



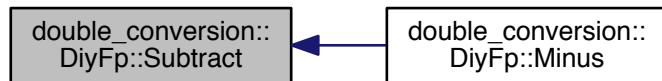
6.6.2.8 void double_conversion::DiyFp::set_f(uint64_t new_value) [inline]

Here is the caller graph for this function:



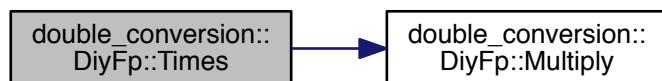
6.6.2.9 void double_conversion::DiyFp::Subtract (const DiyFp & other) [inline]

Here is the caller graph for this function:



6.6.2.10 static DiyFp double_conversion::DiyFp::Times (const DiyFp & a, const DiyFp & b) [inline], [static]

Here is the call graph for this function:



6.6.3 Member Data Documentation

6.6.3.1 const int double_conversion::DiyFp::kSignificandSize = 64 [static]

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

- include/kenlm/util/double-conversion/diy-fp.h

6.7 double_conversion::Double Class Reference

```
#include <ieee.h>
```

Public Member Functions

- `Double ()`
- `Double (double d)`
- `Double (uint64_t d64)`
- `Double (DiyFp diy_fp)`
- `DiyFp AsDiyFp () const`
- `DiyFp AsNormalizedDiyFp () const`
- `uint64_t AsUint64 () const`
- `double NextDouble () const`
- `double PreviousDouble () const`
- `int Exponent () const`
- `uint64_t Significand () const`
- `bool IsDenormal () const`
- `bool IsSpecial () const`
- `bool IsNan () const`
- `bool IsInfinite () const`
- `int Sign () const`
- `DiyFp UpperBoundary () const`
- `void NormalizedBoundaries (DiyFp *out_m_minus, DiyFp *out_m_plus) const`
- `bool LowerBoundaryIsCloser () const`
- `double value () const`

Static Public Member Functions

- `static int SignificandSizeForOrderOfMagnitude (int order)`
- `static double Infinity ()`
- `static double NaN ()`

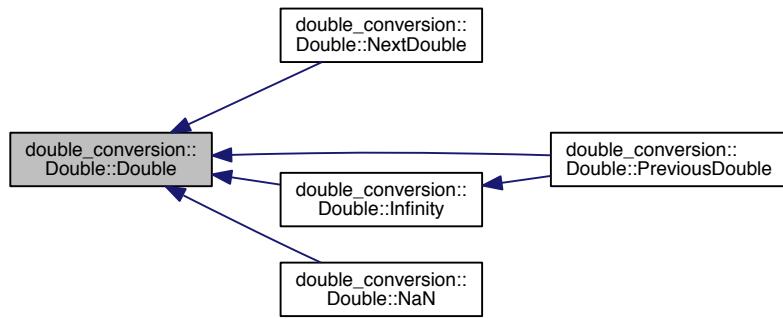
Static Public Attributes

- `static const uint64_t kSignMask = UINT64_2PART_C(0x80000000, 00000000)`
- `static const uint64_t kExponentMask = UINT64_2PART_C(0x7FF00000, 00000000)`
- `static const uint64_t kSignificandMask = UINT64_2PART_C(0x000FFFFF, FFFFFFFF)`
- `static const uint64_t kHiddenBit = UINT64_2PART_C(0x00100000, 00000000)`
- `static const int kPhysicalSignificandSize = 52`
- `static const int kSignificandSize = 53`

6.7.1 Constructor & Destructor Documentation

6.7.1.1 double_conversion::Double::Double() [inline]

Here is the caller graph for this function:



6.7.1.2 double_conversion::Double::Double(double d) [inline], [explicit]

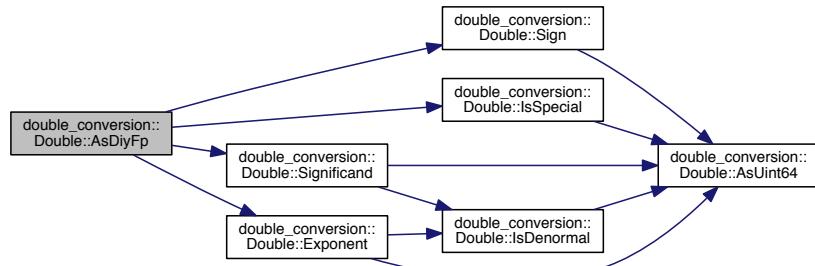
6.7.1.3 double_conversion::Double::Double(uint64_t d64) [inline], [explicit]

6.7.1.4 double_conversion::Double::Double(DiyFp diy_fp) [inline], [explicit]

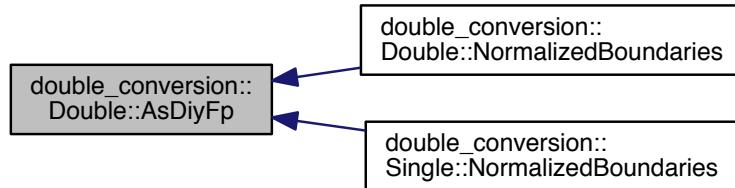
6.7.2 Member Function Documentation

6.7.2.1 DiyFp double_conversion::Double::AsDiyFp() const [inline]

Here is the call graph for this function:

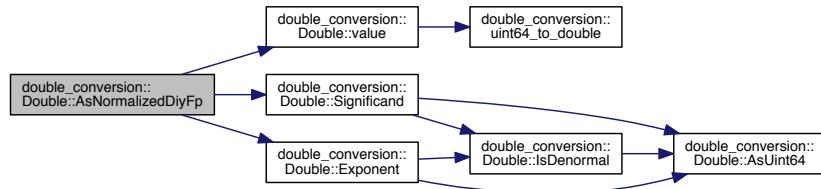


Here is the caller graph for this function:



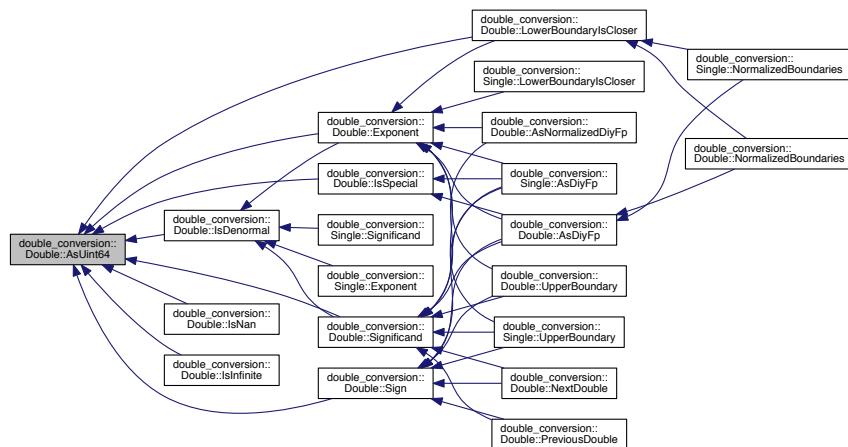
6.7.2.2 DiyFp double_conversion::Double::AsNormalizedDiyFp () const [inline]

Here is the call graph for this function:



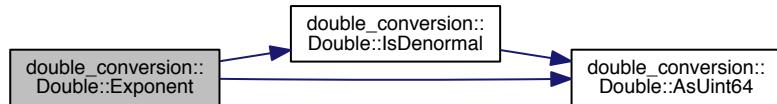
6.7.2.3 uint64_t double_conversion::Double::AsUint64 () const [inline]

Here is the caller graph for this function:

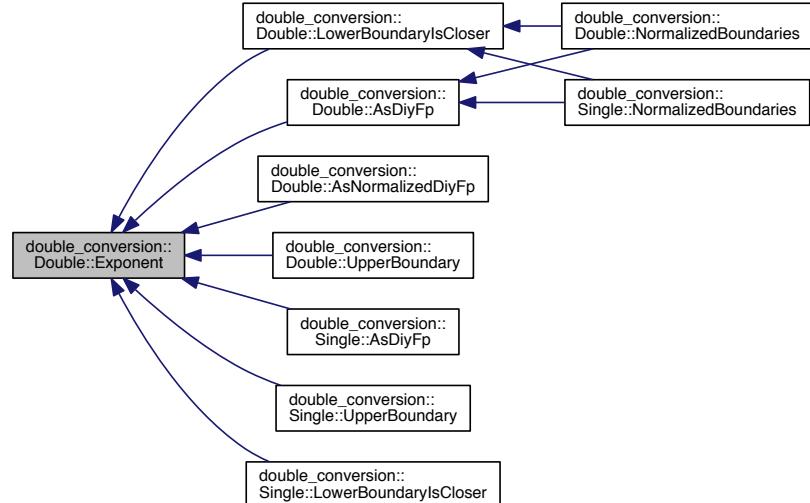


6.7.2.4 int double_conversion::Double::Exponent() const [inline]

Here is the call graph for this function:



Here is the caller graph for this function:

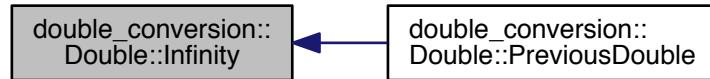


6.7.2.5 static double double_conversion::Double::Infinity() [inline], [static]

Here is the call graph for this function:



Here is the caller graph for this function:

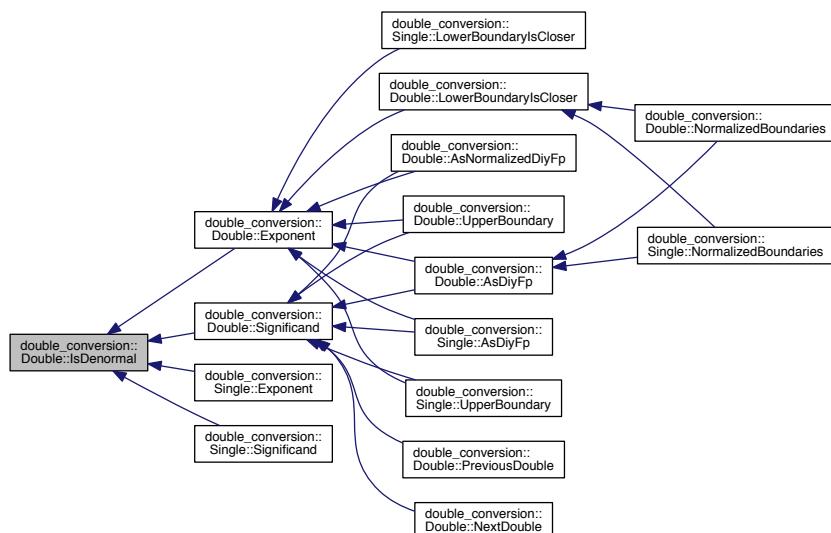


6.7.2.6 bool double_conversion::Double::IsDenormal() const [inline]

Here is the call graph for this function:

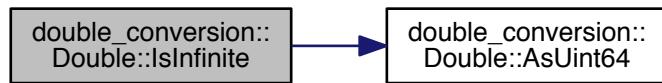


Here is the caller graph for this function:



6.7.2.7 bool double_conversion::Double::IsInfinite() const [inline]

Here is the call graph for this function:



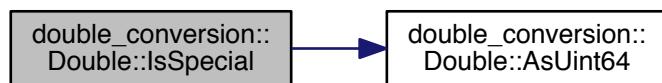
6.7.2.8 bool double_conversion::Double::IsNaN() const [inline]

Here is the call graph for this function:

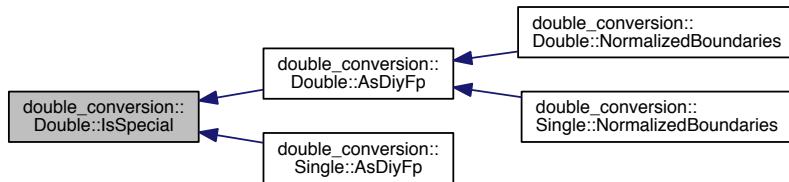


6.7.2.9 bool double_conversion::Double::IsSpecial() const [inline]

Here is the call graph for this function:

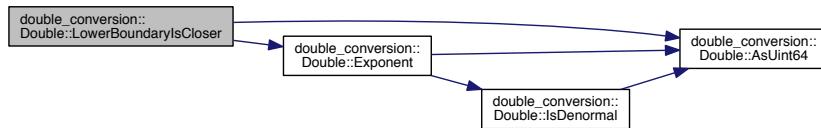


Here is the caller graph for this function:

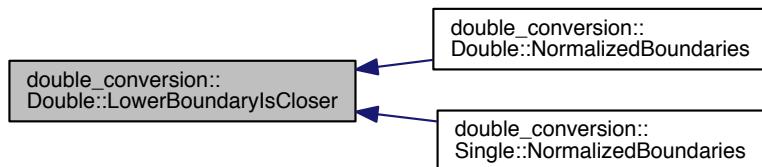


6.7.2.10 bool double_conversion::Double::LowerBoundaryIsCloser () const [inline]

Here is the call graph for this function:

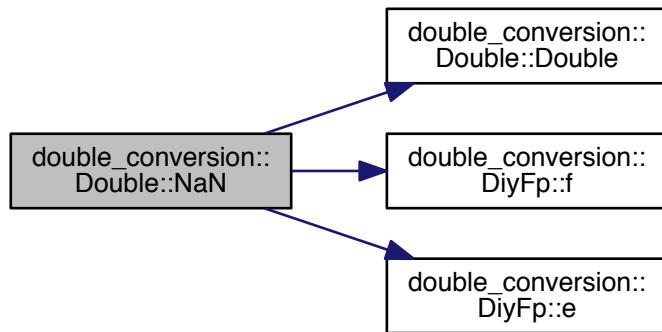


Here is the caller graph for this function:



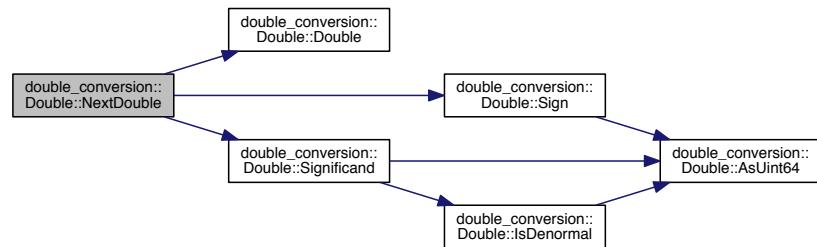
6.7.2.11 static double double_conversion::Double::NaN() [inline], [static]

Here is the call graph for this function:



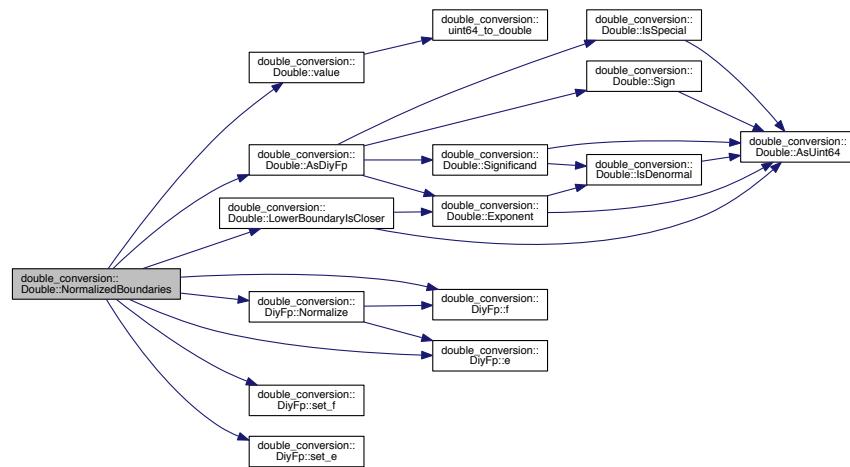
6.7.2.12 double double_conversion::Double::NextDouble() const [inline]

Here is the call graph for this function:



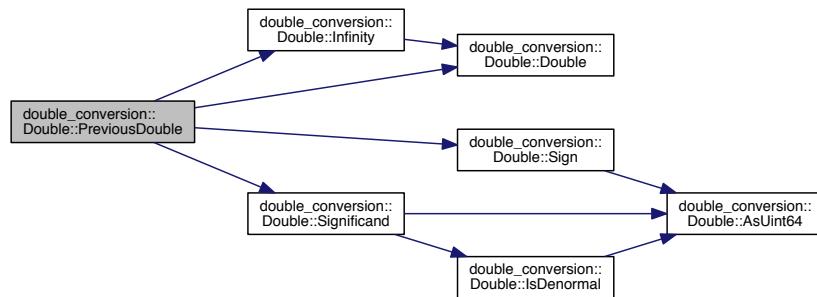
6.7.2.13 void double_conversion::Double::NormalizedBoundaries (**DiyFp** * *out_m_minus*, **DiyFp** * *out_m_plus*) const [inline]

Here is the call graph for this function:



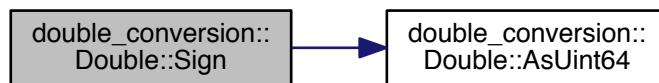
6.7.2.14 double double_conversion::Double::PreviousDouble () const [inline]

Here is the call graph for this function:

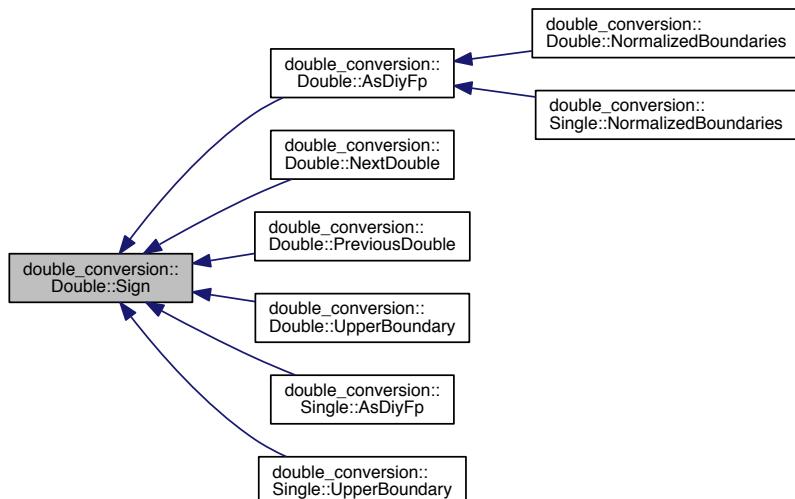


6.7.2.15 int double_conversion::Double::Sign() const [inline]

Here is the call graph for this function:

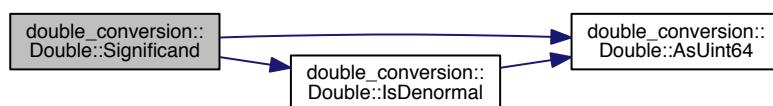


Here is the caller graph for this function:

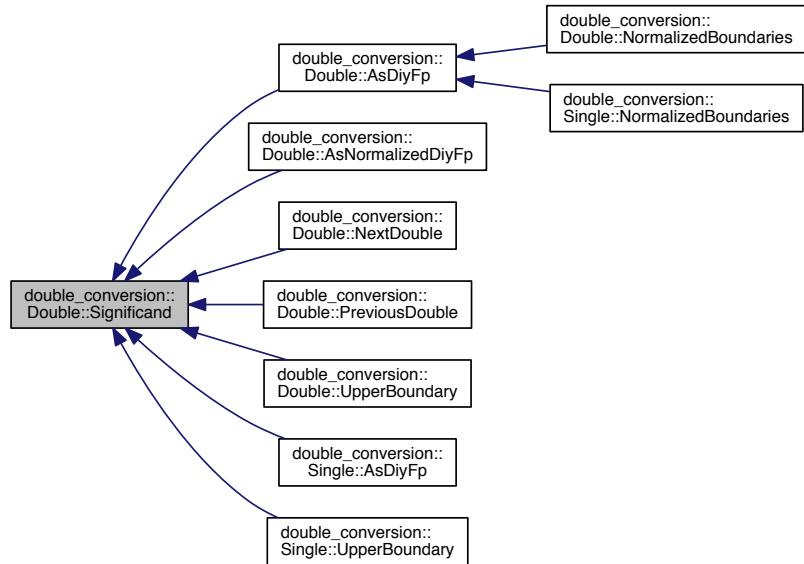


6.7.2.16 uint64_t double_conversion::Double::Significand() const [inline]

Here is the call graph for this function:



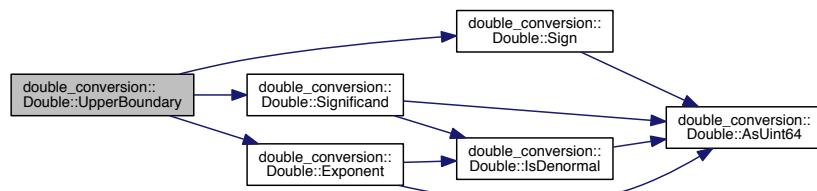
Here is the caller graph for this function:



6.7.2.17 static int double_conversion::Double::SignificandSizeForOrderOfMagnitude (int *order*) [inline], [static]

6.7.2.18 DiyFp double_conversion::Double::UpperBoundary () const [inline]

Here is the call graph for this function:

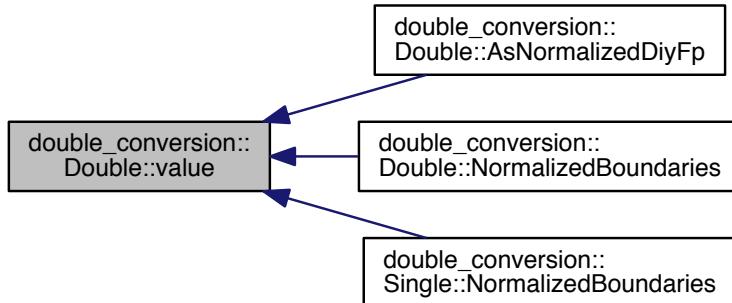


6.7.2.19 double double_conversion::Double::value () const [inline]

Here is the call graph for this function:



Here is the caller graph for this function:



6.7.3 Member Data Documentation

6.7.3.1 `const uint64_t double_conversion::Double::kExponentMask = UINT64_2PART_C(0x7FF00000, 00000000)` [static]

6.7.3.2 `const uint64_t double_conversion::Double::kHiddenBit = UINT64_2PART_C(0x00100000, 00000000)` [static]

6.7.3.3 `const int double_conversion::Double::kPhysicalSignificandSize = 52` [static]

6.7.3.4 `const uint64_t double_conversion::Double::kSignificandMask = UINT64_2PART_C(0x000FFFFF, FFFFFFFF)` [static]

6.7.3.5 `const int double_conversion::Double::kSignificandSize = 53` [static]

6.7.3.6 `const uint64_t double_conversion::Double::kSignMask = UINT64_2PART_C(0x80000000, 00000000)` [static]

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

- include/kenlm/util/double-conversion/ieee.h

6.8 double_conversion::DoubleToStringConverter Class Reference

```
#include <double-conversion.h>
```

Public Types

- enum `Flags` {
 `NO_FLAGS` = 0, `EMIT_POSITIVE_EXPONENT_SIGN` = 1, `EMIT_TRAILING_DECIMAL_POINT` = 2, `EMIT_TRAILING_ZERO_AFTER_POINT` = 4,
`UNIQUE_ZERO` = 8 }
- enum `DtoaMode` { `SHORTEST`, `SHORTEST_SINGLE`, `FIXED`, `PRECISION` }

Public Member Functions

- `DoubleToStringConverter` (int flags, const char *infinity_symbol, const char *nan_symbol, char exponent_character, int decimal_in_shortest_low, int decimal_in_shortest_high, int max_leading_padding_zeroes_in_precision_mode, int max_trailing_padding_zeroes_in_precision_mode)
- bool `ToShortest` (double value, `StringBuilder` *result_builder) const
- bool `ToShortestSingle` (float value, `StringBuilder` *result_builder) const
- bool `ToFixed` (double value, int requested_digits, `StringBuilder` *result_builder) const
- bool `ToExponential` (double value, int requested_digits, `StringBuilder` *result_builder) const
- bool `ToPrecision` (double value, int precision, `StringBuilder` *result_builder) const

Static Public Member Functions

- static const `DoubleToStringConverter` & `EcmaScriptConverter` ()
- static void `DoubleToAscii` (double v, `DtoaMode` mode, int requested_digits, char *buffer, int buffer_length, bool *sign, int *length, int *point)

Static Public Attributes

- static const int `kMaxFixedDigitsBeforePoint` = 60
- static const int `kMaxFixedDigitsAfterPoint` = 60
- static const int `kMaxExponentialDigits` = 120
- static const int `kMinPrecisionDigits` = 1
- static const int `kMaxPrecisionDigits` = 120
- static const int `kBase10MaximalLength` = 17

6.8.1 Member Enumeration Documentation

6.8.1.1 enum double_conversion::DoubleToStringConverter::DtoaMode

Enumerator

`SHORTEST`

`SHORTEST_SINGLE`

`FIXED`

`PRECISION`

6.8.1.2 enum double_conversion::DoubleToStringConverter::Flags

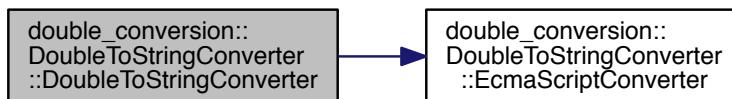
Enumerator

```
NO_FLAGS
EMIT_POSITIVE_EXPONENT_SIGN
EMIT_TRAILING_DECIMAL_POINT
EMIT_TRAILING_ZERO_AFTER_POINT
UNIQUE_ZERO
```

6.8.2 Constructor & Destructor Documentation

- 6.8.2.1 `double_conversion::DoubleToStringConverter::DoubleToStringConverter (int flags, const char *infinity_symbol, const char *nan_symbol, char exponent_character, int decimal_in_shortest_low, int decimal_in_shortest_high, int max_leading_padding_zeroes_in_precision_mode, int max_trailing_padding_zeroes_in_precision_mode)` [inline]

Here is the call graph for this function:

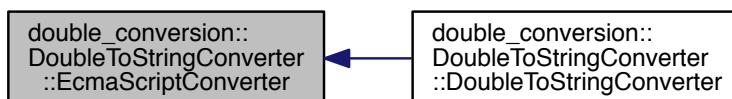


6.8.3 Member Function Documentation

- 6.8.3.1 `static void double_conversion::DoubleToStringConverter::DoubleToAscii (double v, DtoaMode mode, int requested_digits, char * buffer, int buffer_length, bool * sign, int * length, int * point)` [static]

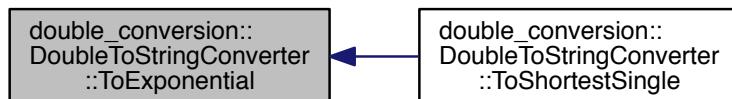
- 6.8.3.2 `static const DoubleToStringConverter& double_conversion::DoubleToStringConverter::EcmaScriptConverter ()` [static]

Here is the caller graph for this function:



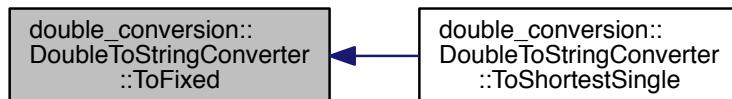
```
6.8.3.3 bool double_conversion::DoubleToStringConverter::ToExponential ( double value, int requested_digits,  
StringBuilder * result_builder ) const
```

Here is the caller graph for this function:



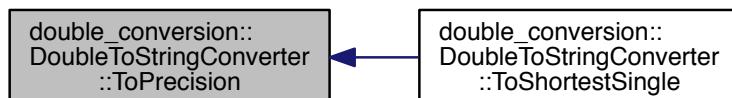
```
6.8.3.4 bool double_conversion::DoubleToStringConverter::ToFixed ( double value, int requested_digits, StringBuilder *\nresult_builder ) const
```

Here is the caller graph for this function:



```
6.8.3.5 bool double_conversion::DoubleToStringConverter::ToPrecision ( double value, int precision, StringBuilder *\nresult_builder ) const
```

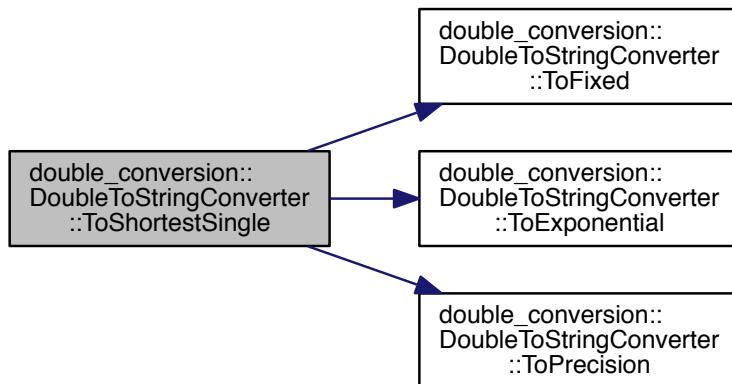
Here is the caller graph for this function:



```
6.8.3.6 bool double_conversion::DoubleToStringConverter::ToShortest( double value, StringBuilder *result_builder )
const [inline]

6.8.3.7 bool double_conversion::DoubleToStringConverter::ToShortestSingle( float value, StringBuilder *result_builder )
const [inline]
```

Here is the call graph for this function:



6.8.4 Member Data Documentation

```
6.8.4.1 const int double_conversion::DoubleToStringConverter::kBase10MaximalLength = 17 [static]

6.8.4.2 const int double_conversion::DoubleToStringConverter::kMaxExponentialDigits = 120 [static]

6.8.4.3 const int double_conversion::DoubleToStringConverter::kMaxFixedDigitsAfterPoint = 60 [static]

6.8.4.4 const int double_conversion::DoubleToStringConverter::kMaxFixedDigitsBeforePoint = 60 [static]

6.8.4.5 const int double_conversion::DoubleToStringConverter::kMaxPrecisionDigits = 120 [static]

6.8.4.6 const int double_conversion::DoubleToStringConverter::kMinPrecisionDigits = 1 [static]
```

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

- include/kenlm/util/double-conversion/[double-conversion.h](#)

6.9 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 bound 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.
- [int getBP \(\)](#)
Accessor to the determined best point.

6.9.1 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.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Eval::Eval ()

Default constructor.

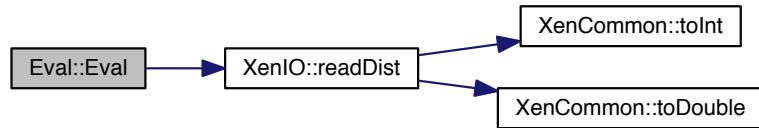
6.9.2.2 Eval::Eval (std::string distFile)

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:



6.9.2.3 Eval::~Eval()

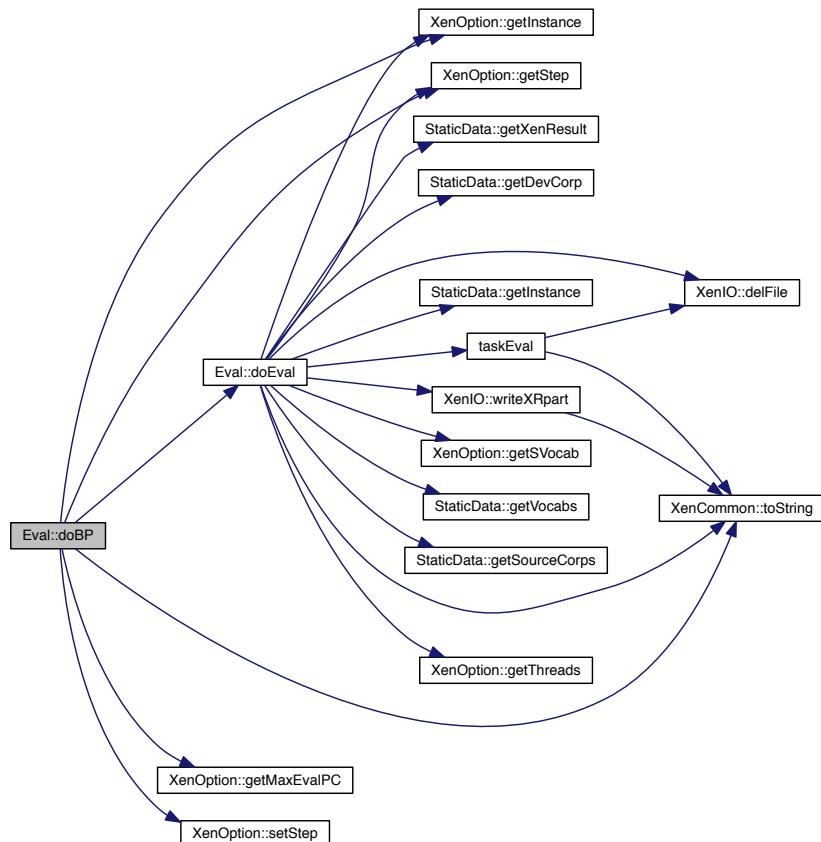
Default destructor.

6.9.3 Member Function Documentation

6.9.3.1 void Eval::doBP()

Computes the best theoretical point based on current evaluation.

Here is the call graph for this function:



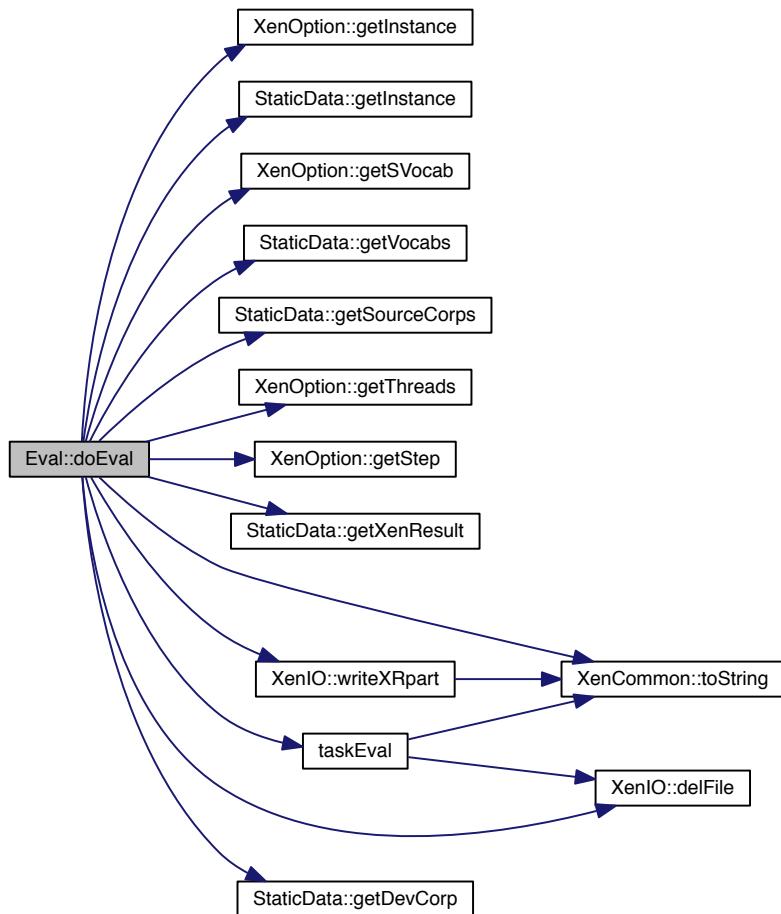
6.9.3.2 void Eval::doEval (int *high*, int *low*)

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:



6.9.3.3 int Eval::getBP()

Accessor to the determined best point.

Returns

integer representing the percentage for the best point

6.9.3.4 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](#)

6.10 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<XenLMken> getPtrInLM\(\)** const](#)
Accessor to the in-domain language model.
- [**boost::shared_ptr<XenLMken> getPtrOutLM\(\)** const](#)
Accessor to the out-of-domain language model.

6.10.1 Detailed Description

Tiny class holding two related language models.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 LMPair::LMPair() [inline]

Default constructor.

6.10.2.2 LMPair::~LMPair() [inline]

Default destructor.

6.10.3 Member Function Documentation

6.10.3.1 boost::shared_ptr< XenLMken > LMPair::getPtrInLM() const [inline]

Accessor to the in-domain language model.

Returns

the in-domain language model

6.10.3.2 boost::shared_ptr< XenLMken > 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](#)

6.11 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< XenLMken > getPtrOutLM2 \(\) const](#)
Accessor to the second out-of-domain language model.
- [boost::shared_ptr< XenLMken > getPtrOutLM3 \(\) const](#)
Accessor to the third out-of-domain language model.

6.11.1 Detailed Description

Tiny class holding two additional LMs for mean scoring feature.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 MeanLMPair::MeanLMPair() [inline]

Default constructor.

6.11.2.2 MeanLMPair::~MeanLMPair() [inline]

Default Destructor.

6.11.3 Member Function Documentation

6.11.3.1 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

6.11.3.2 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](#)

6.12 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.

6.12.1 Detailed Description

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

6.12.2 Constructor & Destructor Documentation

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

Default constructor.

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

Default Destructor.

6.12.3 Member Function Documentation

6.12.3.1 `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

6.12.3.2 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:

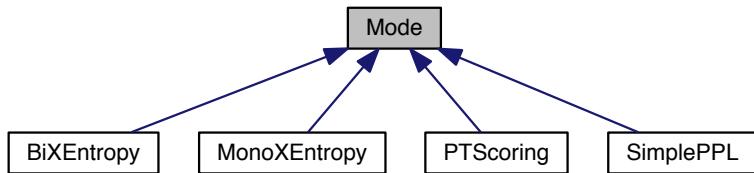
- include/utils/[StaticData.h](#)

6.13 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](#).

6.13.1 Detailed Description

Filtering modes interface.

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

6.13.2 Constructor & Destructor Documentation

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

Pure virtual destructor.

6.13.3 Member Function Documentation

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

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

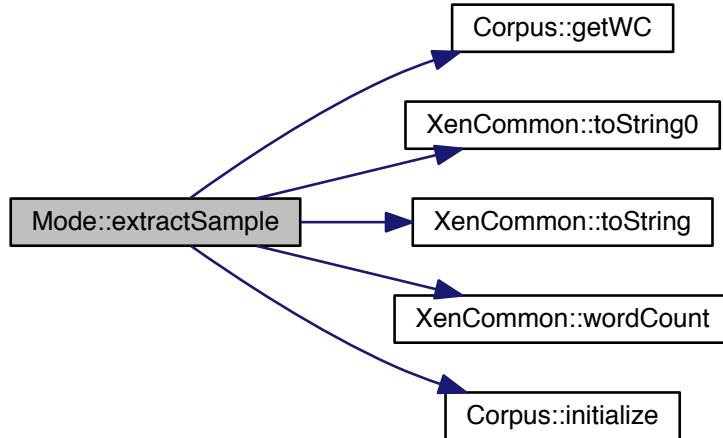
Parameters

<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)

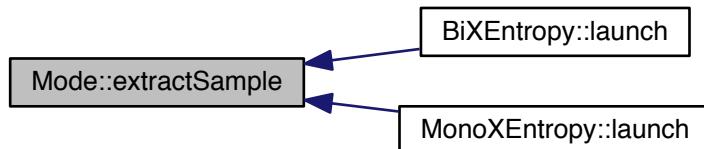
Returns

extracted [Corpus](#) sample

Here is the call graph for this function:



Here is the caller graph for this function:



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

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

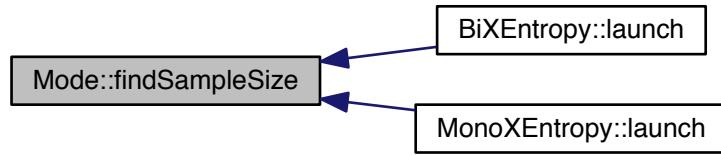
Parameters

<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:



6.13.3.3 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:

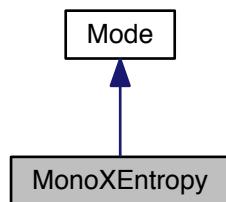
- [include/mode.h](#)
- [src/mode.cpp](#)

6.14 MonoXEntropy Class Reference

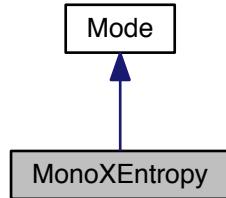
Filtering mode 2: monolingual cross-entropy.

```
#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

6.14.1 Detailed Description

Filtering mode 2: monolingual cross-entropy.

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

6.14.2 Constructor & Destructor Documentation

6.14.2.1 [MonoXEntropy::MonoXEntropy \(\)](#)

Default constructor.

6.14.2.2 [MonoXEntropy::~MonoXEntropy \(\)](#)

Default destructor.

6.14.3 Member Function Documentation

6.14.3.1 int MonoXEntropy::launch() [virtual]

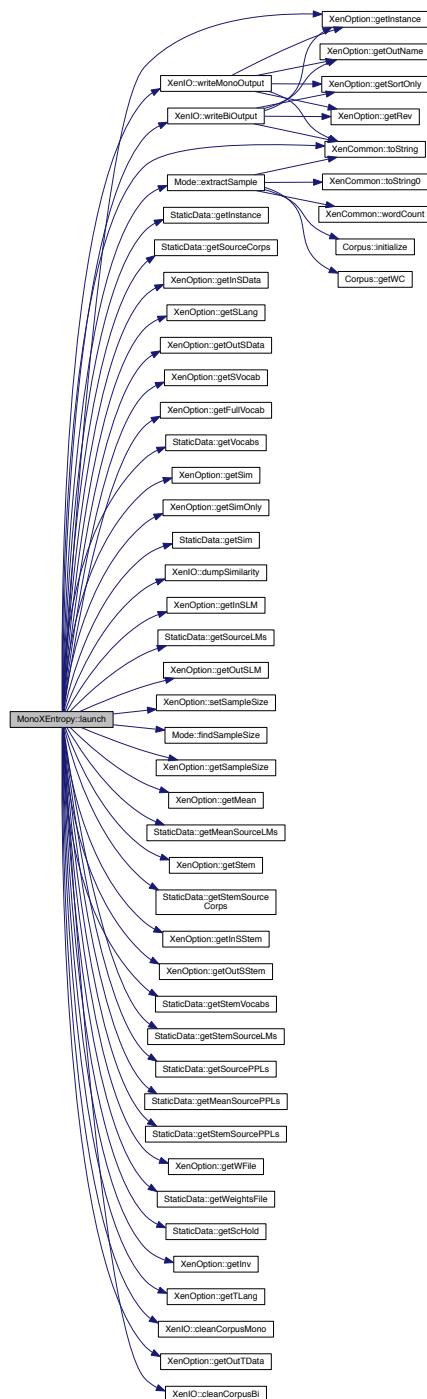
Function in charge of launching the filtering mode.

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](#)

6.15 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 instanciated 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.

6.15.1 Detailed Description

Class handling phrase-table related functionalities.

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

6.15.2 Constructor & Destructor Documentation

6.15.2.1 PhraseTable::PhraseTable()

Default constructor.

6.15.2.2 PhraseTable::~PhraseTable()

Default destructor.

6.15.3 Member Function Documentation

6.15.3.1 std::string PhraseTable::getAlignment(int n)

Accessor to the nth alignments for the source/target phrase pair.

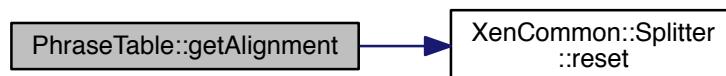
Parameters

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

Returns

string containing the alignment

Here is the call graph for this function:



Here is the caller graph for this function:

**6.15.3.2 std::string PhraseTable::getCounts(int n)**

Accessor to the nth counts for the source/target phrase pair.

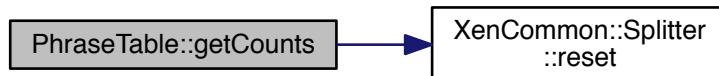
Parameters

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

Returns

string containing the counts

Here is the call graph for this function:



Here is the caller graph for this function:



6.15.3.3 std::string PhraseTable::getScores (int n)

Accessor to the nth scores for the source/target phrase pair.

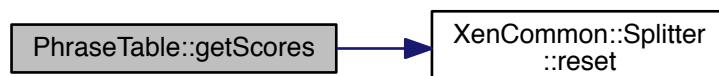
Parameters

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

Returns

string containing the scores

Here is the call graph for this function:



Here is the caller graph for this function:



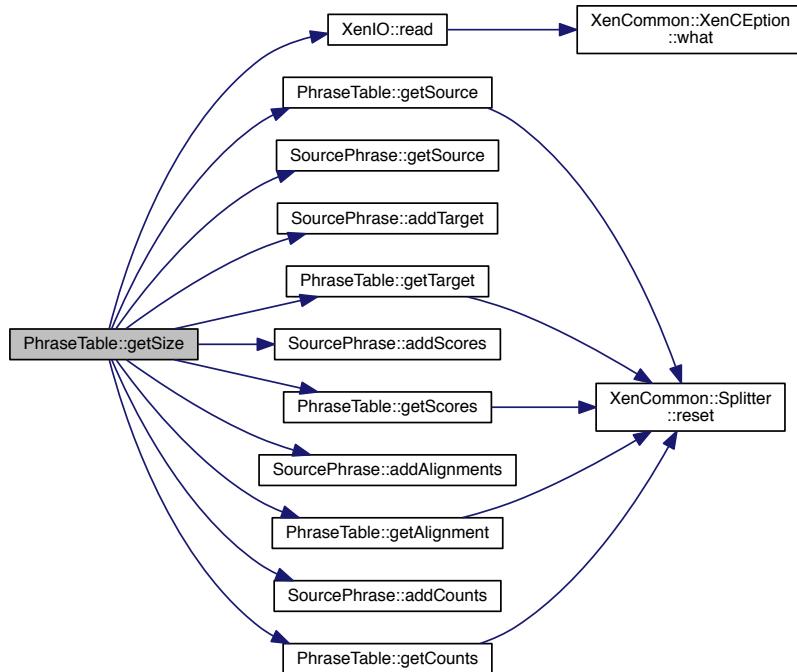
6.15.3.4 unsigned int PhraseTable::getSize () const

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

Returns

unsigned int representing the size

Here is the call graph for this function:



6.15.3.5 std::string PhraseTable::getSource (int n)

Accessor to the nth source phrase.

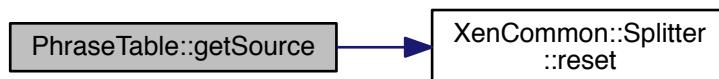
Parameters

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

Returns

string containing the source phrase

Here is the call graph for this function:



Here is the caller graph for this function:

**6.15.3.6 std::vector<SourcePhrase> PhraseTable::getSrcPhrases()**

Accessor to the vector of merged source phrases.

Returns

vector of merged [SourcePhrase](#)

6.15.3.7 std::string PhraseTable::getTarget(int n)

Accessor to the nth target phrase.

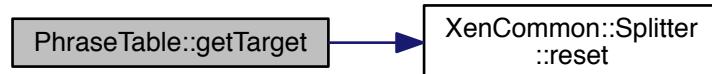
Parameters

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

Returns

string containing the target phrase

Here is the call graph for this function:



Here is the caller graph for this function:

**6.15.3.8 boost::shared_ptr< XenFile > PhraseTable::getXenFile () const**

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

Returns

shared pointer to the [XenFile](#)

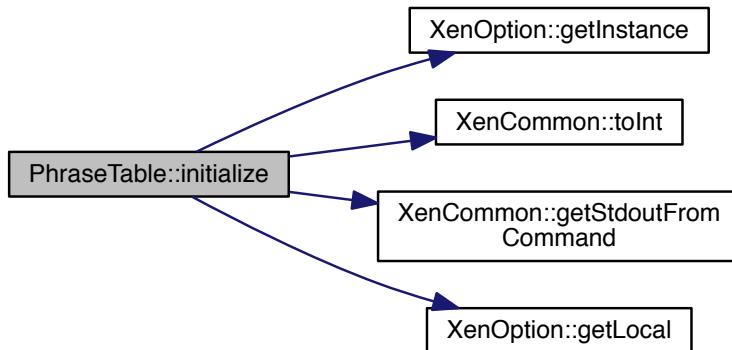
6.15.3.9 void PhraseTable::initialize (boost::shared_ptr< XenFile > ptrData)

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

Parameters

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

Here is the call graph for this function:



6.15.3.10 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](#)

6.16 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.

6.16.1 Detailed Description

Tiny class holding the two phrase-tables.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 PhraseTablePair::PhraseTablePair() [inline]

Default constructor.

6.16.2.2 PhraseTablePair::~PhraseTablePair() [inline]

Default destructor.

6.16.3 Member Function Documentation

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

Accessor to the in-domain phrase-table.

Returns

the in-domain phrase-table

6.16.3.2 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/utils/[StaticData.h](#)

6.17 double_conversion::PowersOfTenCache Class Reference

```
#include <cached-powers.h>
```

Static Public Member Functions

- static void [GetCachedPowerForBinaryExponentRange](#) (int min_exponent, int max_exponent, [DiyFp](#) *power, int *decimal_exponent)
- static void [GetCachedPowerForDecimalExponent](#) (int requested_exponent, [DiyFp](#) *power, int *found_exponent)

Static Public Attributes

- static const int [kDecimalExponentDistance](#)
- static const int [kMinDecimalExponent](#)
- static const int [kMaxDecimalExponent](#)

6.17.1 Member Function Documentation

6.17.1.1 static void [double_conversion::PowersOfTenCache::GetCachedPowerForBinaryExponentRange](#) (int *min_exponent*, int *max_exponent*, [DiyFp](#) * *power*, int * *decimal_exponent*) [static]

6.17.1.2 static void [double_conversion::PowersOfTenCache::GetCachedPowerForDecimalExponent](#) (int *requested_exponent*, [DiyFp](#) * *power*, int * *found_exponent*) [static]

6.17.2 Member Data Documentation

6.17.2.1 const int [double_conversion::PowersOfTenCache::kDecimalExponentDistance](#) [static]

6.17.2.2 const int [double_conversion::PowersOfTenCache::kMaxDecimalExponent](#) [static]

6.17.2.3 const int [double_conversion::PowersOfTenCache::kMinDecimalExponent](#) [static]

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

- include/kenlm/util/double-conversion/[cached-powers.h](#)

6.18 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< [XenLMken](#) > ptrLM)
• void **initialize** (boost::shared_ptr< [PhraseTable](#) > ptrPT, boost::shared_ptr< [XenLMken](#) > ptrLM, bool source)
• **~PPL ()**
Default destructor.
- unsigned int **getSize** () const
Accessor to the size of the perplexity/cross-entropy vector.
- double **getPPL** (int n)
Accessor to the nth perplexity score.
- double **getXE** (int n)
Accessor to the nth 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.

6.18.1 Detailed Description

Perplexity/Cross-entropy computations.

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

6.18.2 Constructor & Destructor Documentation

6.18.2.1 PPL::PPL()

Default constructor.

6.18.2.2 PPL::~PPL()

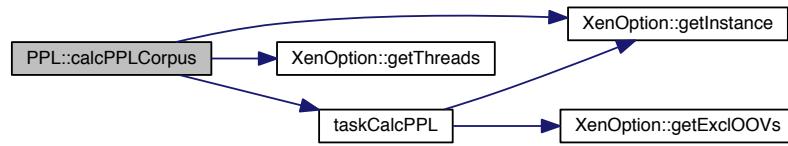
Default destructor.

6.18.3 Member Function Documentation

6.18.3.1 void PPL::calcPPLCorpus()

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

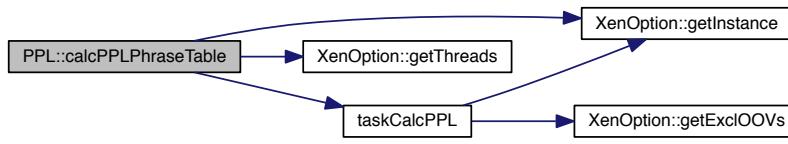
Here is the call graph for this function:



6.18.3.2 void PPL::calcPPLPhraseTable()

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

Here is the call graph for this function:



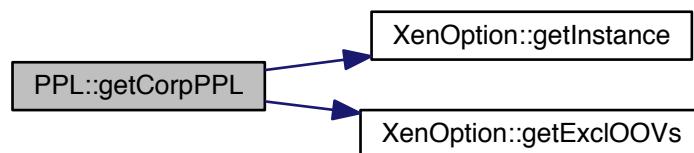
6.18.3.3 double PPL::getCorpPPL()

Accessor to the document-level perplexity score.

Returns

double representing the document perplexity score

Here is the call graph for this function:



6.18.3.4 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

6.18.3.5 unsigned int PPL::getSize () const

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

Returns

unsigned int representing the size

6.18.3.6 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

6.18.3.7 void PPL::initialize (boost::shared_ptr< Corpus > ptrCorp, boost::shared_ptr< XenLMken > ptrLM)**6.18.3.8 void PPL::initialize (boost::shared_ptr< PhraseTable > ptrPT, boost::shared_ptr< XenLMken > ptrLM, bool source)**

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

- include/ppl.h
- src/ppl.cpp

6.19 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.

6.19.1 Detailed Description

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

6.19.2 Constructor & Destructor Documentation

6.19.2.1 PPLPair::PPLPair() [inline]

Default constructor.

6.19.2.2 PPLPair::~PPLPair() [inline]

Default destructor.

6.19.3 Member Function Documentation

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

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

Returns

the in-domain [PPL](#) object

6.19.3.2 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:

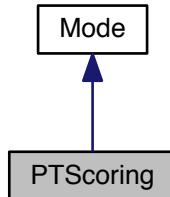
- include/utils/[StaticData.h](#)

6.20 PTScoring Class Reference

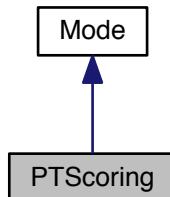
Filtering mode 4: phrase-table cross-entropy.

```
#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

6.20.1 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

6.20.2 Constructor & Destructor Documentation

6.20.2.1 PTScoreing::PTScoreing()

Default constructor.

6.20.2.2 PTScoreing::~PTScoreing()

Default destructor.

6.20.3 Member Function Documentation

6.20.3.1 int PTScoreing::launch() [virtual]

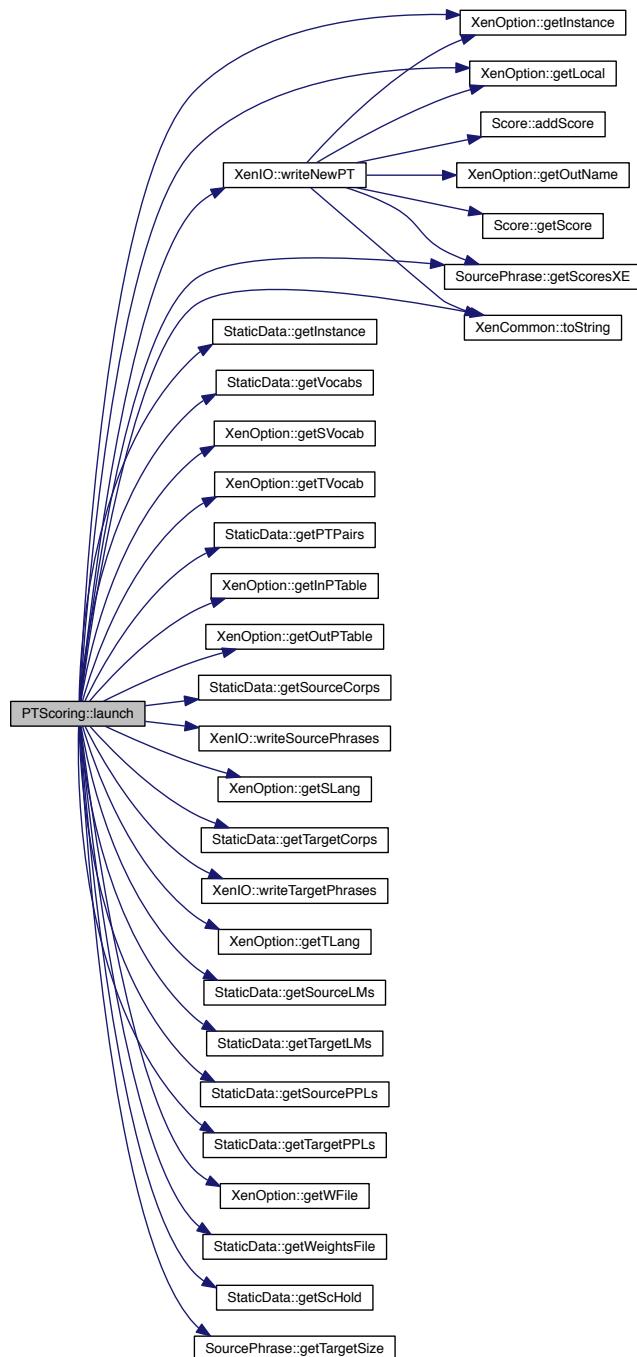
Function in charge of launching the filtering mode.

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](#)

6.21 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)

6.21.1 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.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 [Score::Score \(\)](#)

Default constructor.

6.21.2.2 [Score::~Score \(\)](#)

Default destructor.

6.21.3 Member Function Documentation

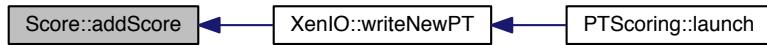
6.21.3.1 [void Score::addScore \(double sc \)](#)

Adds a score to the vector of doubles.

Parameters

<code>sc</code>	: score to add to the Score holder
-----------------	--

Here is the caller graph for this function:

**6.21.3.2 void Score::calibrate()**

Calibrates the scores distribution between 0 and 1.

6.21.3.3 bool Score::getPrint(int n) const

Accessor to the output status of the nth score.

Parameters

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

Returns

true if the score should be outputted

6.21.3.4 double Score::getScore(int n) const

Accessor to the nth score.

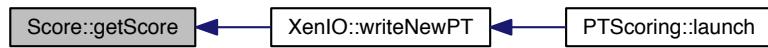
Parameters

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

Returns

double representing the requested score

Here is the caller graph for this function:

**6.21.3.5 unsigned int Score::getSize () const**

Accessor to the size of the scores vector.

Returns

unsigned int representing the size

6.21.3.6 void Score::inverse ()

Inverts the calibrated score distribution ($1 - \text{score}$)

6.21.3.7 void Score::removeScore (int n)

Removes the nth 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](#)

6.22 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.

6.22.1 Detailed Description

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

6.22.2 Constructor & Destructor Documentation

6.22.2.1 [ScoreHolder::ScoreHolder\(\) \[inline\]](#)

Default constructor.

6.22.2.2 [ScoreHolder::~ScoreHolder\(\) \[inline\]](#)

Default Destructor.

6.22.3 Member Function Documentation

6.22.3.1 [boost::shared_ptr< Score > ScoreHolder::getPtrScores\(\) const \[inline\]](#)

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

Returns

the global [Score](#) object

6.22.3.2 [boost::shared_ptr< Score > ScoreHolder::getPtrScSimil\(\) const \[inline\]](#)

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

Returns

the similarity measures [Score](#) object

6.22.3.3 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](#)

6.23 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 nth sentence similarity measure.
- unsigned int [getSize](#) () const
Accessor to the size of the similarity map.

6.23.1 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

6.23.2 Constructor & Destructor Documentation

6.23.2.1 [Similarity::Similarity\(\)](#)

Default constructor.

6.23.2.2 [Similarity::~Similarity\(\)](#)

Default destructor.

6.23.3 Member Function Documentation

6.23.3.1 [float Similarity::getSim\(int n \)](#)

Accessor to the nth sentence similarity measure.

Parameters

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

Returns

float representing the similarity measure of the nth sentence

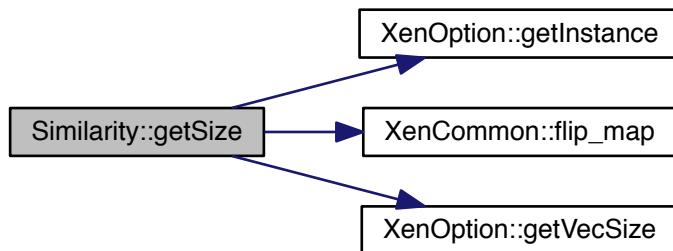
6.23.3.2 unsigned int Similarity::getSize () const

Accessor to the size of the similarity map.

Returns

unsigned int representing the size

Here is the call graph for this function:

**6.23.3.3 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:

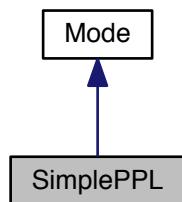
- include/[similarity.h](#)
- src/[similarity.cpp](#)

6.24 SimplePPL Class Reference

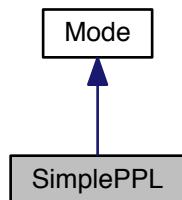
Filtering mode 1: simple perplexity.

```
#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

6.24.1 Detailed Description

Filtering mode 1: simple perplexity.

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

6.24.2 Constructor & Destructor Documentation

6.24.2.1 SimplePPL::SimplePPL()

Default constructor.

6.24.2.2 SimplePPL::~SimplePPL()

Default destructor.

6.24.3 Member Function Documentation

6.24.3.1 int SimplePPL::launch() [virtual]

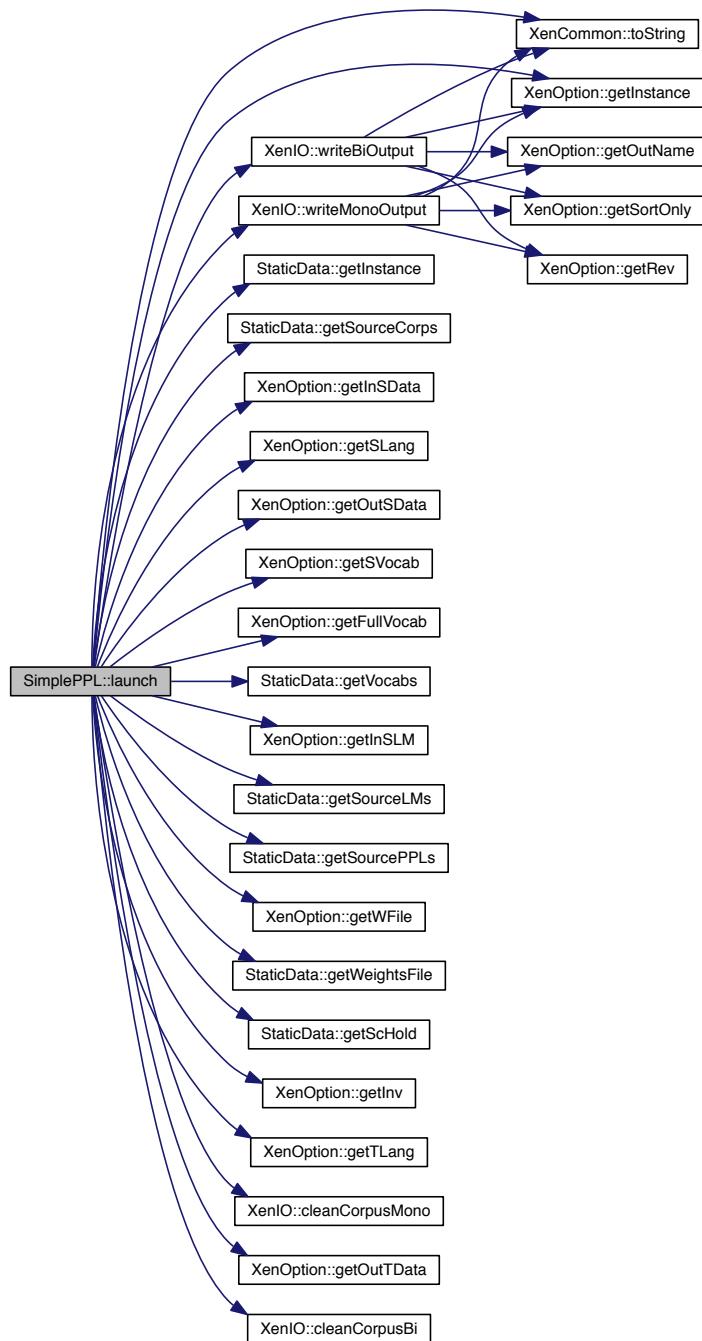
Function in charge of launching the filtering mode.

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](#)

6.25 double_conversion::Single Class Reference

```
#include <ieee.h>
```

Public Member Functions

- [Single \(\)](#)
- [Single \(float f\)](#)
- [Single \(uint32_t d32\)](#)
- [DiyFp AsDiyFp \(\) const](#)
- [uint32_t AsUint32 \(\) const](#)
- [int Exponent \(\) const](#)
- [uint32_t Significand \(\) const](#)
- [bool IsDenormal \(\) const](#)
- [bool IsSpecial \(\) const](#)
- [bool Isnan \(\) const](#)
- [bool IsInfinite \(\) const](#)
- [int Sign \(\) const](#)
- [void NormalizedBoundaries \(DiyFp *out_m_minus, DiyFp *out_m_plus\) const](#)
- [DiyFp UpperBoundary \(\) const](#)
- [bool LowerBoundaryIsCloser \(\) const](#)
- [float value \(\) const](#)

Static Public Member Functions

- [static float Infinity \(\)](#)
- [static float NaN \(\)](#)

Static Public Attributes

- [static const uint32_t kSignMask = 0x80000000](#)
- [static const uint32_t kExponentMask = 0x7F800000](#)
- [static const uint32_t kSignificandMask = 0x007FFFFF](#)
- [static const uint32_t kHiddenBit = 0x00800000](#)
- [static const int kPhysicalSignificandSize = 23](#)
- [static const int kSignificandSize = 24](#)

6.25.1 Constructor & Destructor Documentation

6.25.1.1 [double_conversion::Single\(\) \[inline\]](#)

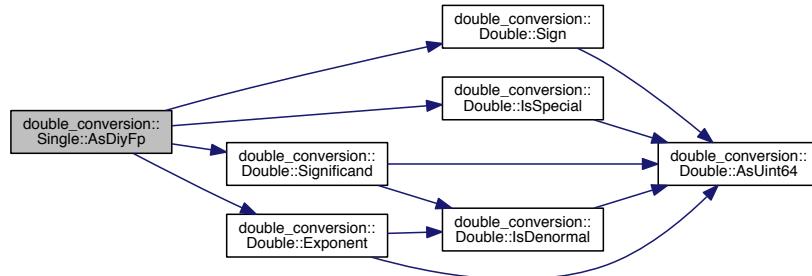
6.25.1.2 [double_conversion::Single\(float f \) \[inline\], \[explicit\]](#)

6.25.1.3 [double_conversion::Single\(uint32_t d32 \) \[inline\], \[explicit\]](#)

6.25.2 Member Function Documentation

6.25.2.1 `DiyFp double_conversion::Single::AsDiyFp() const [inline]`

Here is the call graph for this function:



6.25.2.2 `uint32_t double_conversion::Single::AsUInt32() const [inline]`

6.25.2.3 `int double_conversion::Single::Exponent() const [inline]`

Here is the call graph for this function:



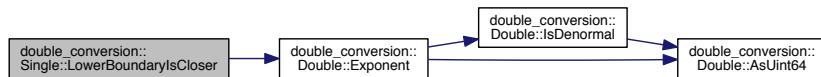
6.25.2.4 `static float double_conversion::Single::Infinity() [inline], [static]`

Here is the call graph for this function:



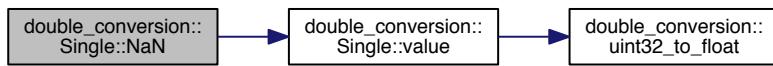
- 6.25.2.5 `bool double_conversion::Single::IsDenormal() const [inline]`
- 6.25.2.6 `bool double_conversion::Single::IsInfinite() const [inline]`
- 6.25.2.7 `bool double_conversion::Single::IsNaN() const [inline]`
- 6.25.2.8 `bool double_conversion::Single::IsSpecial() const [inline]`
- 6.25.2.9 `bool double_conversion::Single::LowerBoundaryIsCloser() const [inline]`

Here is the call graph for this function:



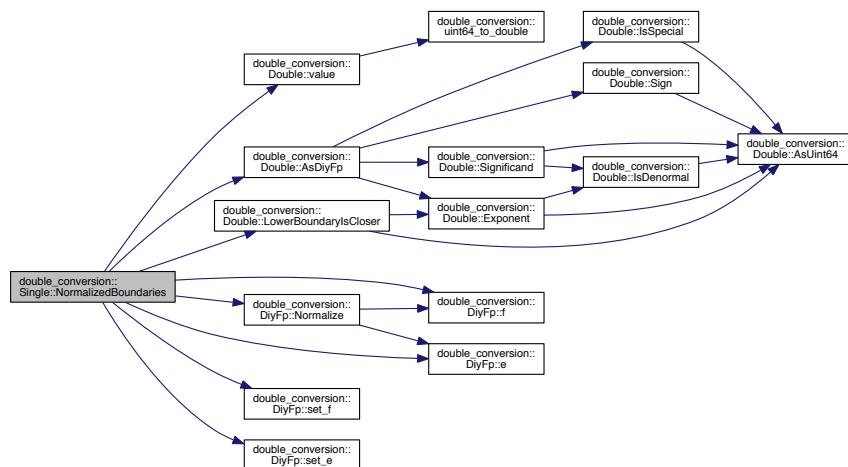
- 6.25.2.10 `static float double_conversion::Single::NaN() [inline], [static]`

Here is the call graph for this function:



- 6.25.2.11 `void double_conversion::Single::NormalizedBoundaries(DiyFp * out_m_minus, DiyFp * out_m_plus) const [inline]`

Here is the call graph for this function:



6.25.2.12 `int double_conversion::Single::Sign() const [inline]`

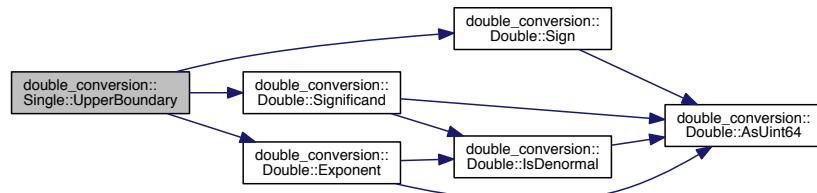
6.25.2.13 `uint32_t double_conversion::Single::Significand() const [inline]`

Here is the call graph for this function:



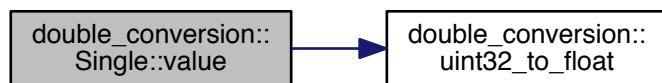
6.25.2.14 `DiyFp double_conversion::Single::UpperBoundary() const [inline]`

Here is the call graph for this function:

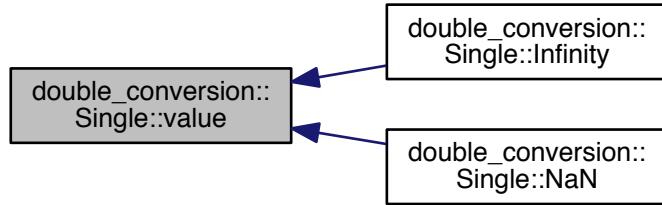


6.25.2.15 `float double_conversion::Single::value() const [inline]`

Here is the call graph for this function:



Here is the caller graph for this function:



6.25.3 Member Data Documentation

6.25.3.1 `const uint32_t double_conversion::Single::kExponentMask = 0x7F800000 [static]`

6.25.3.2 `const uint32_t double_conversion::Single::kHiddenBit = 0x00800000 [static]`

6.25.3.3 `const int double_conversion::Single::kPhysicalSignificandSize = 23 [static]`

6.25.3.4 `const uint32_t double_conversion::Single::kSignificandMask = 0x007FFFFF [static]`

6.25.3.5 `const int double_conversion::Single::kSignificandSize = 24 [static]`

6.25.3.6 `const uint32_t double_conversion::Single::kSignMask = 0x80000000 [static]`

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

- include/kenlm/util/double-conversion/[ieee.h](#)

6.26 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.

6.26.1 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.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 SourcePhrase::SourcePhrase (std::string src)

Constructor from a string.

Parameters

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

6.26.2.2 SourcePhrase::~SourcePhrase ()

Default destructor.

6.26.3 Member Function Documentation

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

Associates alignments to the source phrase.

Parameters

<code>s</code>	: the alignments to add to the source phrase
----------------	--

Here is the caller graph for this function:

**6.26.3.2 void SourcePhrase::addCounts (std::string s)**

Associates counts to the source phrase.

Parameters

<code>s</code>	: the counts to add to the source phrase
----------------	--

Here is the caller graph for this function:

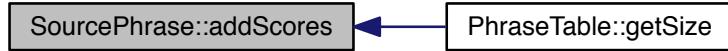
**6.26.3.3 void SourcePhrase::addScores (std::string s)**

Associates a phrase table scores sequence to the source phrase.

Parameters

<code>s</code>	: the scores sequence to add to the source phrase
----------------	---

Here is the caller graph for this function:



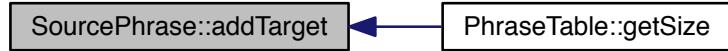
6.26.3.4 void SourcePhrase::addTarget(std::string s)

Associates a target phrase to the source phrase.

Parameters

<code>s</code>	: the target phrase to add to the source phrase
----------------	---

Here is the caller graph for this function:



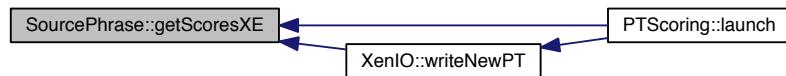
6.26.3.5 boost::shared_ptr< Score > SourcePhrase::getScoresXE() const

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:



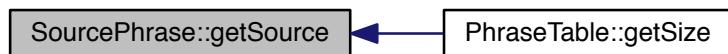
6.26.3.6 std::string SourcePhrase::getSource () const

Accessor to the source phrase.

Returns

the source phrase

Here is the caller graph for this function:



6.26.3.7 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

6.27 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)`

6.27.1 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)

6.27.2 Member Typedef Documentation

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

6.27.3 Constructor & Destructor Documentation

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

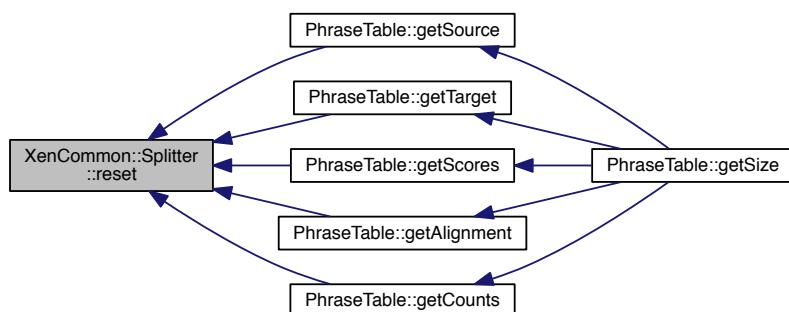
6.27.3.2 `XenCommon::Splitter::Splitter (const std::string & src, const std::string & delim) [inline]`

6.27.4 Member Function Documentation

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

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

Here is the caller graph for this function:



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

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

- include/utils/common.h

6.28 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](#).

6.28.1 Detailed Description

Class gathering all data used and generated by XenC.

6.28.2 Member Function Documentation

6.28.2.1 void [StaticData::deleteInstance](#)() [static]

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

Here is the caller graph for this function:



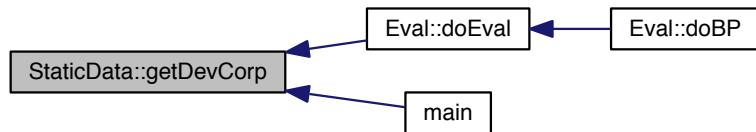
6.28.2.2 boost::shared_ptr<Corpus> StaticData::getDevCorp() [static]

Accessor to the development [Corpus](#).

Returns

the development [Corpus](#)

Here is the caller graph for this function:

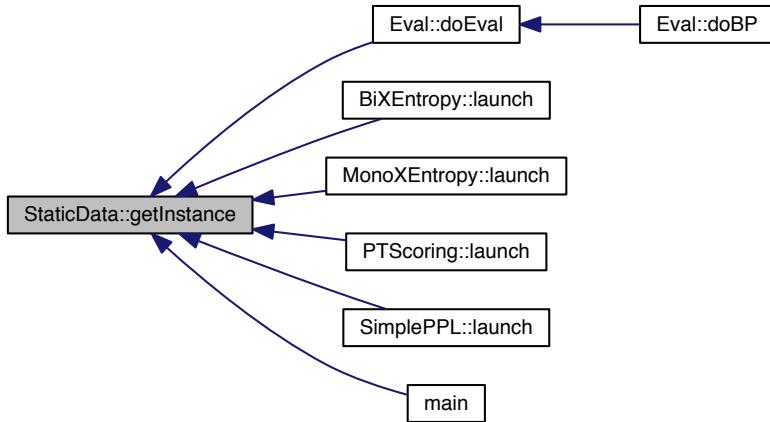
**6.28.2.3 StaticData * StaticData::getInstance() [static]**

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

Returns

the [StaticData](#) unique instance

Here is the caller graph for this function:



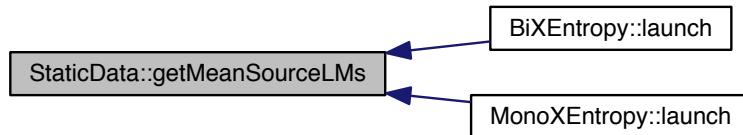
6.28.2.4 `boost::shared_ptr< MeanLMPair > StaticData::getMeanSourceLMs() [static]`

Accessor to the mean source language models.

Returns

the mean source language models

Here is the caller graph for this function:



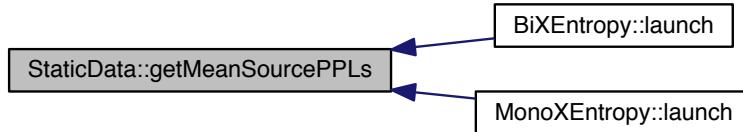
6.28.2.5 `boost::shared_ptr< MeanPPLPair > StaticData::getMeanSourcePPLs() [static]`

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

Returns

the mean source [PPL](#) objects

Here is the caller graph for this function:



6.28.2.6 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:

**6.28.2.7 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:

**6.28.2.8 boost::shared_ptr< PhraseTablePair > StaticData::getPTPairs() [static]**

Accessor to the phrase-tables.

Returns

the phrase-tables

Here is the caller graph for this function:



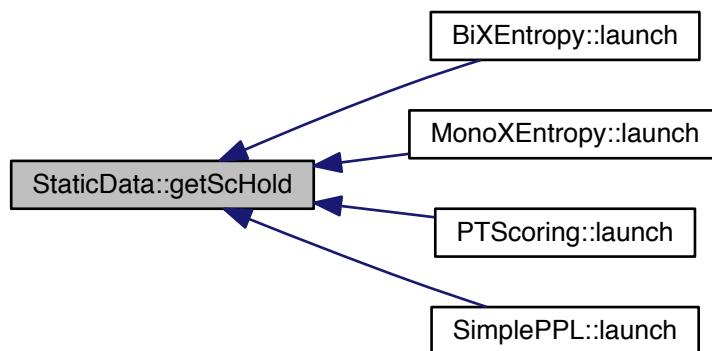
6.28.2.9 `boost::shared_ptr< ScoreHolder > StaticData::getScHold() [static]`

Accessor to the [ScoreHolder](#) object.

Returns

the [ScoreHolder](#) object

Here is the caller graph for this function:



6.28.2.10 `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:



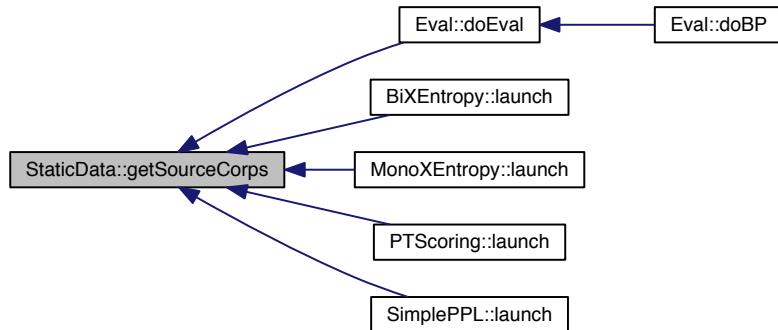
6.28.2.11 boost::shared_ptr< CorpusPair > StaticData::getSourceCorps() [static]

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

Returns

the source language [Corpus](#) Pair

Here is the caller graph for this function:

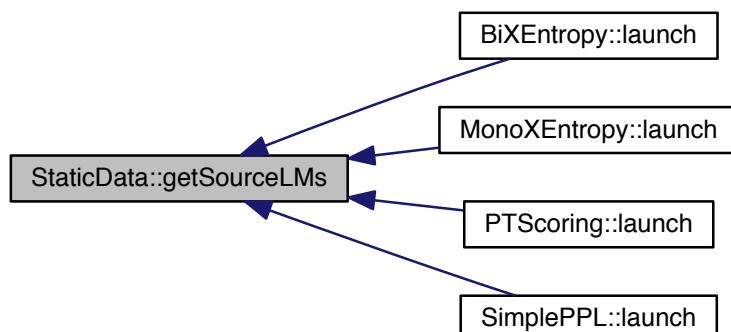
**6.28.2.12 boost::shared_ptr< LMPair > StaticData::getSourceLMS() [static]**

Accessor to the source language models.

Returns

the source language models

Here is the caller graph for this function:



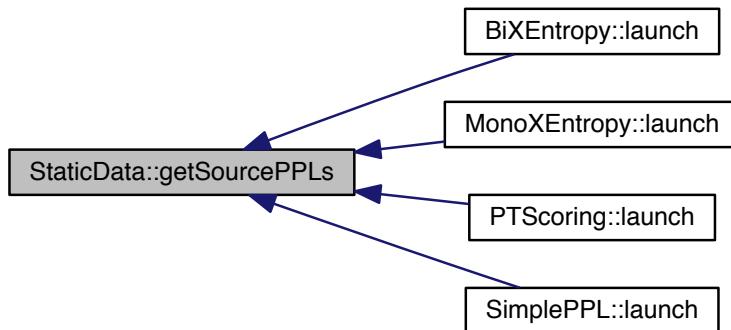
6.28.2.13 boost::shared_ptr< PPLPair > StaticData::getSourcePPLs() [static]

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

Returns

the source language [PPL](#) objects

Here is the caller graph for this function:



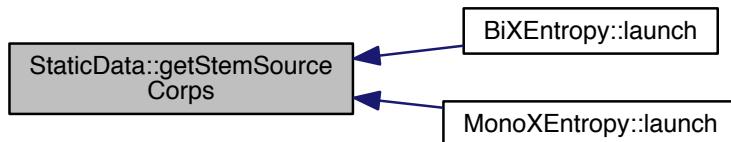
6.28.2.14 boost::shared_ptr< CorpusPair > StaticData::getStemSourceCorps() [static]

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

Returns

the source language stem [Corpus](#) Pair

Here is the caller graph for this function:



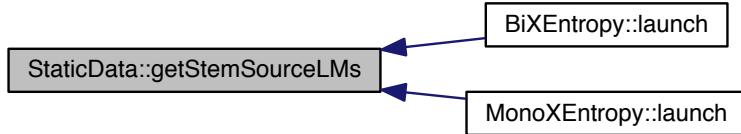
6.28.2.15 boost::shared_ptr< LMPair > StaticData::getStemSourceLMs() [static]

Accessor to the source language stem language models.

Returns

the source language stem language models

Here is the caller graph for this function:

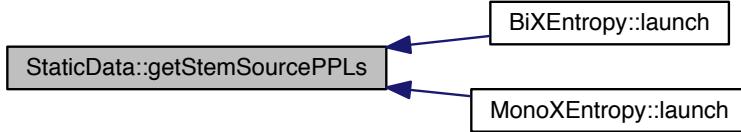
**6.28.2.16 boost::shared_ptr< PPLPair > StaticData::getStemSourcePPLs() [static]**

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

Returns

the source language stem [PPL](#) objects

Here is the caller graph for this function:



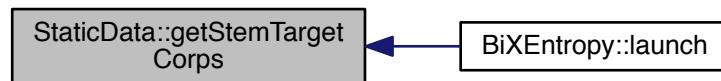
6.28.2.17 boost::shared_ptr< CorpusPair > StaticData::getStemTargetCorps() [static]

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

Returns

the target language stem [Corpus](#) Pair

Here is the caller graph for this function:

**6.28.2.18 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:

**6.28.2.19 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:

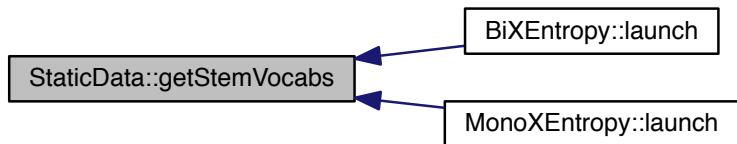
**6.28.2.20 boost::shared_ptr< VocabPair > StaticData::getStemVocabs() [static]**

Accessor to the stem vocabularies.

Returns

the stem vocabularies

Here is the caller graph for this function:

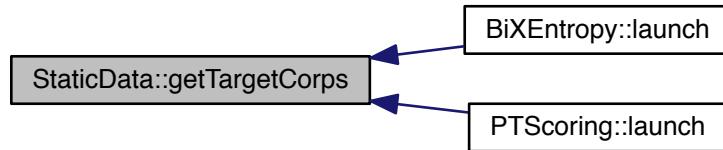
**6.28.2.21 boost::shared_ptr< CorpusPair > StaticData::getTargetCorps() [static]**

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

Returns

the target language [Corpus](#) Pair

Here is the caller graph for this function:

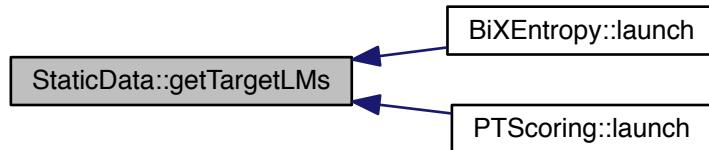
**6.28.2.22 boost::shared_ptr< LMPair > StaticData::getTargetLMS() [static]**

Accessor to the target language models.

Returns

the target language models

Here is the caller graph for this function:

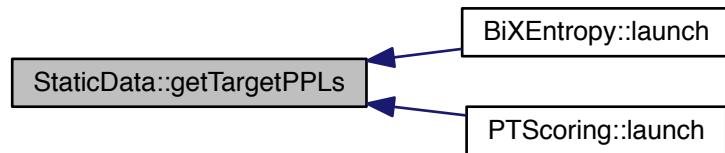
**6.28.2.23 boost::shared_ptr< PPLPair > StaticData::getTargetPPLs() [static]**

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

Returns

the target language [PPL](#) objects

Here is the caller graph for this function:

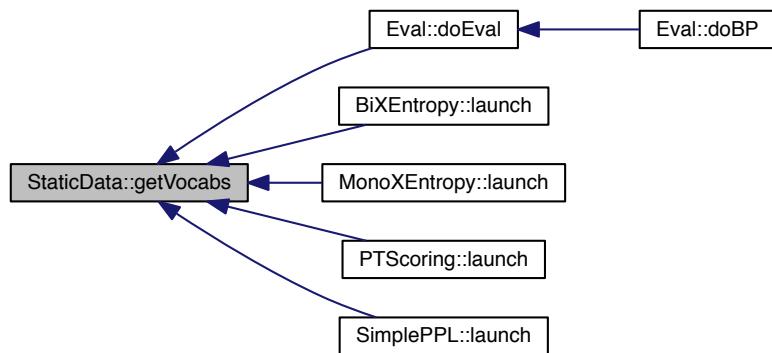
**6.28.2.24 boost::shared_ptr< VocabPair > StaticData::getVocabs() [static]**

Accessor to the vocabularies.

Returns

the vocabularies

Here is the caller graph for this function:

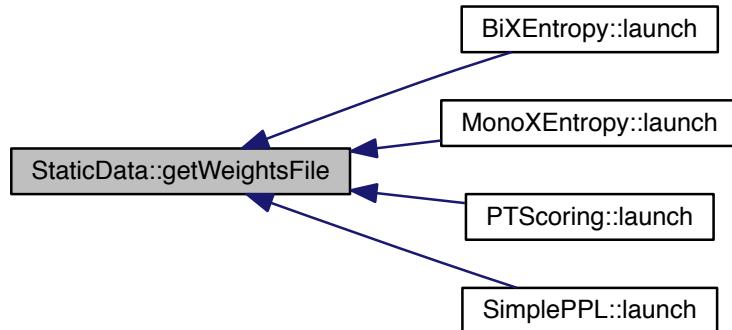
**6.28.2.25 boost::shared_ptr< Wfile > StaticData::getWeightsFile() [static]**

Accessor to the weights file.

Returns

the weights file

Here is the caller graph for this function:



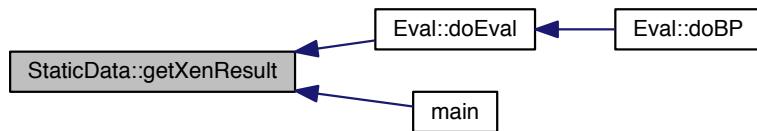
6.28.2.26 boost::shared_ptr< XenResult > StaticData::getXenResult () [static]

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](#)

6.29 double_conversion::StringBuilder Class Reference

```
#include <utils.h>
```

Public Member Functions

- `StringBuilder (char *buffer, int size)`
- `~StringBuilder ()`
- `int size () const`
- `int position () const`
- `void Reset ()`
- `void AddCharacter (char c)`
- `void AddString (const char *s)`
- `void AddSubstring (const char *s, int n)`
- `void AddPadding (char c, int count)`
- `char * Finalize ()`

6.29.1 Constructor & Destructor Documentation

6.29.1.1 `double_conversion::StringBuilder (char * buffer, int size) [inline]`

6.29.1.2 `double_conversion::StringBuilder::~StringBuilder () [inline]`

6.29.2 Member Function Documentation

6.29.2.1 `void double_conversion::StringBuilder::AddCharacter (char c) [inline]`

6.29.2.2 `void double_conversion::StringBuilder::AddPadding (char c, int count) [inline]`

6.29.2.3 `void double_conversion::StringBuilder::AddString (const char * s) [inline]`

Here is the call graph for this function:



6.29.2.4 void double_conversion::StringBuilder::AddSubstring(const char * s, int n) [inline]

6.29.2.5 char* double_conversion::StringBuilder::Finalize() [inline]

6.29.2.6 int double_conversion::StringBuilder::position() const [inline]

6.29.2.7 void double_conversion::StringBuilder::Reset() [inline]

6.29.2.8 int double_conversion::StringBuilder::size() const [inline]

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

- include/kenlm/util/double-conversion/utils.h

6.30 double_conversion::StringToDoubleConverter Class Reference

```
#include <double-conversion.h>
```

Public Types

- enum Flags {
 NO_FLAGS = 0, ALLOW_HEX = 1, ALLOW_OCTALS = 2, ALLOW_TRAILING_JUNK = 4,
 ALLOW.LEADING_SPACES = 8, ALLOW.TRAILING_SPACES = 16, ALLOW.SPACES.AFTER_SIGN =
 32 }

Public Member Functions

- `StringToDoubleConverter` (int flags, double empty_string_value, double junk_string_value, const char *infinity_symbol, const char *nan_symbol)
- double `StringToDouble` (const char *buffer, int length, int *processed_characters_count) const
- float `StringToFloat` (const char *buffer, int length, int *processed_characters_count) const

6.30.1 Member Enumeration Documentation

6.30.1.1 enum double_conversion::StringToDoubleConverter::Flags

Enumerator

```
NO_FLAGS
ALLOW_HEX
ALLOW_OCTALS
ALLOW_TRAILING_JUNK
ALLOW.LEADING_SPACES
ALLOW.TRAILING_SPACES
ALLOW.SPACES.AFTER_SIGN
```

6.30.2 Constructor & Destructor Documentation

6.30.2.1 `double_conversion::StringToDoubleConverter::StringToDoubleConverter (int flags, double empty_string_value, double junk_string_value, const char * infinity_symbol, const char * nan_symbol) [inline]`

6.30.3 Member Function Documentation

6.30.3.1 `double double_conversion::StringToDoubleConverter::StringToDouble (const char * buffer, int length, int * processed_characters_count) const [inline]`

6.30.3.2 `float double_conversion::StringToDoubleConverter::StringToFloat (const char * buffer, int length, int * processed_characters_count) const [inline]`

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

- `include/kenlm/util/double-conversion/double-conversion.h`

6.31 TxtStats Struct Reference

```
#include <XenLMken.h>
```

Public Attributes

- float `prob`
- float `zeroprobs`
- uint64_t `numwords`
- uint64_t `numoov`
- uint64_t `numsentences`

6.31.1 Member Data Documentation

6.31.1.1 `uint64_t TxtStats::numoov`

6.31.1.2 `uint64_t TxtStats::numsentences`

6.31.1.3 `uint64_t TxtStats::numwords`

6.31.1.4 `float TxtStats::prob`

6.31.1.5 `float TxtStats::zeroprobs`

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

- `include/XenLMken.h`

6.32 double_conversion::Vector< T > Class Template Reference

```
#include <utils.h>
```

Public Member Functions

- [Vector \(\)](#)
- [Vector \(T *data, int length\)](#)
- [Vector< T > SubVector \(int from, int to\)](#)
- [int length \(\) const](#)
- [bool is_empty \(\) const](#)
- [T * start \(\) const](#)
- [T & operator\[\] \(int index\) const](#)
- [T & first \(\)](#)
- [T & last \(\)](#)

6.32.1 Constructor & Destructor Documentation

6.32.1.1 `template<typename T> double_conversion::Vector< T >::Vector() [inline]`

6.32.1.2 `template<typename T> double_conversion::Vector< T >::Vector (T * data, int length) [inline]`

6.32.2 Member Function Documentation

6.32.2.1 `template<typename T> T& double_conversion::Vector< T >::first() [inline]`

6.32.2.2 `template<typename T> bool double_conversion::Vector< T >::is_empty() const [inline]`

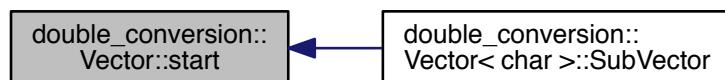
6.32.2.3 `template<typename T> T& double_conversion::Vector< T >::last() [inline]`

6.32.2.4 `template<typename T> int double_conversion::Vector< T >::length() const [inline]`

6.32.2.5 `template<typename T> T& double_conversion::Vector< T >::operator[] (int index) const [inline]`

6.32.2.6 `template<typename T> T* double_conversion::Vector< T >::start() const [inline]`

Here is the caller graph for this function:



6.32.2.7 template<typename T> Vector<T> double_conversion::Vector< T >::SubVector(int from, int to)
[inline]

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

- include/kenlm/util/double-conversion/utils.h

6.33 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.

6.33.1 Detailed Description

Tiny class holding the two vocabularies.

6.33.2 Constructor & Destructor Documentation

6.33.2.1 [VocabPair::VocabPair\(\)](#) [inline]

Default constructor.

6.33.2.2 [VocabPair::~VocabPair\(\)](#) [inline]

Default destructor.

6.33.3 Member Function Documentation

6.33.3.1 boost::shared_ptr<[XenVocab](#)> [VocabPair::getPtrSourceVoc\(\)](#) const [inline]

Accessor to the source vocabulary.

Returns

the source vocabulary

6.33.3.2 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](#)

6.34 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 nth weight of the file.
- unsigned int [getSize \(\) const](#)
Accessor to the size of the weights file.

6.34.1 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.

6.34.2 Constructor & Destructor Documentation

6.34.2.1 Wfile::Wfile ()

Default constructor.

6.34.2.2 Wfile::~Wfile ()

Default destructor.

6.34.3 Member Function Documentation

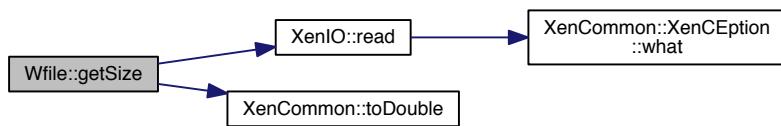
6.34.3.1 unsigned int Wfile::getSize () const

Accessor to the size of the weights file.

Returns

the size of the weights file

Here is the call graph for this function:



6.34.3.2 double Wfile::getWeight (int n)

Accessor to the nth weight of the file.

Parameters

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

Returns

the requested weight

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

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

Parameters

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

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

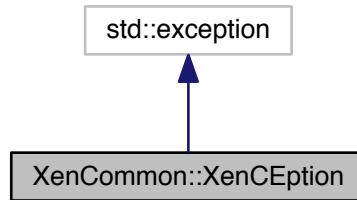
- include/[wfile.h](#)
- src/[wfile.cpp](#)

6.35 XenCommon::XenCEption Struct Reference

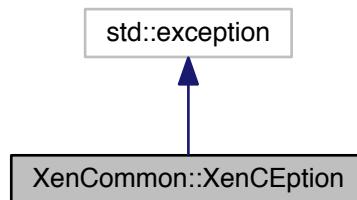
XenC exception structure.

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

Inheritance diagram for XenCommon::XenCEption:



Collaboration diagram for XenCommon::XenCEption:



Public Member Functions

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

Public Attributes

- std::string [s](#)
The exception message.

6.35.1 Detailed Description

XenC exception structure.

6.35.2 Constructor & Destructor Documentation

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

Exception constructor.

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

Exception desctructor.

6.35.3 Member Function Documentation

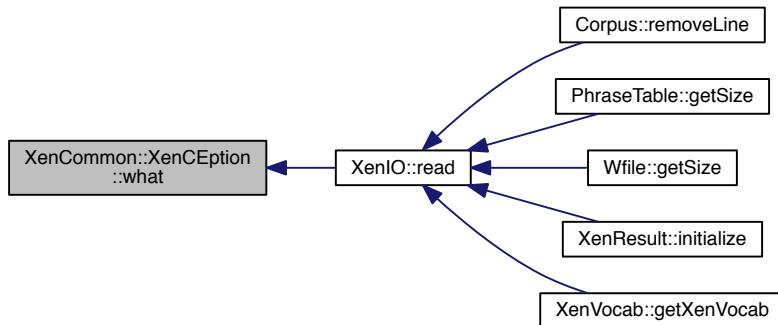
6.35.3.1 const char * XenCommon::XenCEption::what () const throw [inline]

Accessor to the exception message.

Returns

the exception message

Here is the caller graph for this function:



6.35.4 Member Data Documentation

6.35.4.1 std::string XenCommon::XenCEption::s

The exception message.

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

- include/utils/common.h

6.36 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.

6.36.1 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.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 XenFile::XenFile ()

Default constructor.

6.36.2.2 XenFile::~XenFile ()

Default destructor.

6.36.3 Member Function Documentation

6.36.3.1 std::string XenFile::getDirName() const

Accessor to the file's directory name.

Returns

the file's directory name

6.36.3.2 std::string XenFile::getExt()

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

Returns

the file name's extension

6.36.3.3 std::string XenFile::getFileName() const

Accessor to the file name.

Returns

the file name (and only the file name)

6.36.3.4 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:



6.36.3.5 std::string XenFile::getPrefix()

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

Returns

the file name's prefix

6.36.3.6 void XenFile::initialize(std::string name)

Initialization function from a string.

Parameters

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

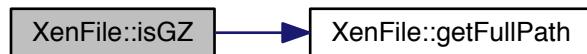
6.36.3.7 bool XenFile::isGZ() const

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

6.37 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`> ptr← CorpTarget, 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`> ptr← CorpTarget, 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 `writeVocab` (std::map< std::string, int > `voc`, std::string `fileName`)
Writes a vocab to file.
- static void `writeXRpart` (boost::shared_ptr< `XenResult` > `ptrXR`, int `pc`, std::string `fileName=""`)
Writes a part of the text of a `XenResult` given a percentage.
- 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.
- static void `delFile` (std::string `fileName`)
Deletes a file from the filesystem.

6.37.1 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...

6.37.2 Member Function Documentation

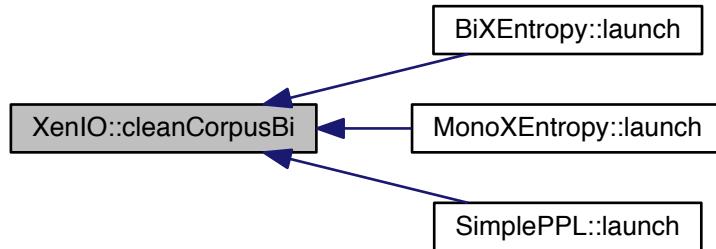
6.37.2.1 void XenIO::cleanCorpusBi (boost::shared_ptr< `Corpus` > `ptrCorpSource`, boost::shared_ptr< `Corpus` > `ptrCorpTarget`, boost::shared_ptr< `Score` > `ptrScore`) [static]

Bilingual corpus cleaning (ensures no empty lines)

Parameters

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

Here is the caller graph for this function:



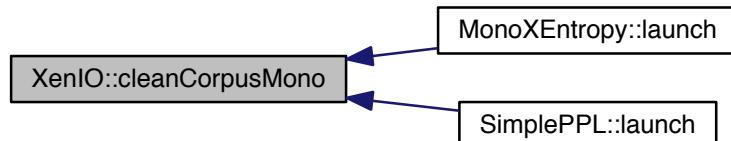
6.37.2.2 `void XenIO::cleanCorpusMono (boost::shared_ptr< Corpus > ptrCorp, boost::shared_ptr< Score > ptrScore) [static]`

Monolingual corpus cleaning (ensures no empty lines)

Parameters

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

Here is the caller graph for this function:



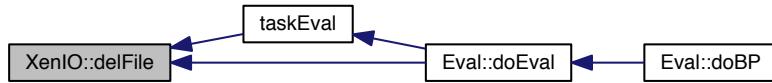
6.37.2.3 `void XenIO::delFile (std::string fileName) [static]`

Deletes a file from the filesystem.

Parameters

<code>fileName</code>	: file to delete
-----------------------	------------------

Here is the caller graph for this function:



6.37.2.4 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:



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

Reads a file (plain text/gzipped)

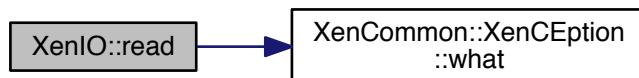
Parameters

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

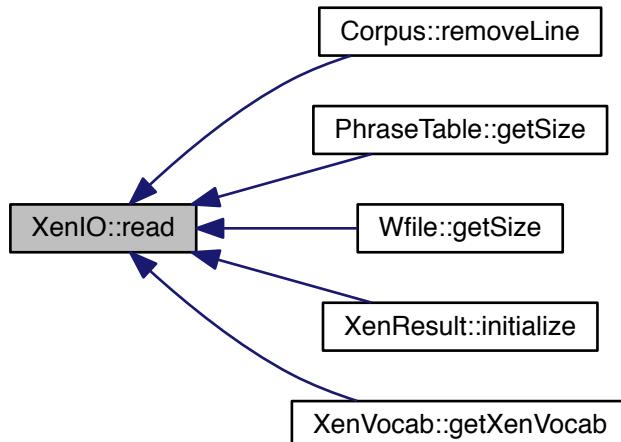
Returns

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:



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

Reads a evaluation/best point distribution file.

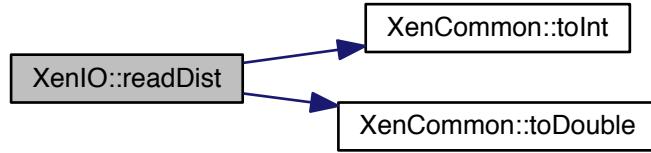
Parameters

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

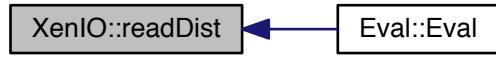
Returns

an EvalMap containing the already computed scores

Here is the call graph for this function:



Here is the caller graph for this function:



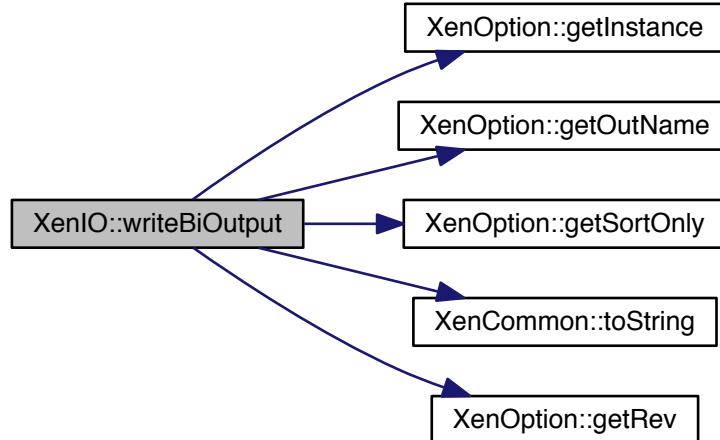
6.37.2.7 void XenIO::writeBiOutput (boost::shared_ptr< Corpus > ptrCorpSource, boost::shared_ptr< Corpus > ptrCorpTarget, boost::shared_ptr< Score > ptrScore) [static]

Writes bilingual scored/sorted result files.

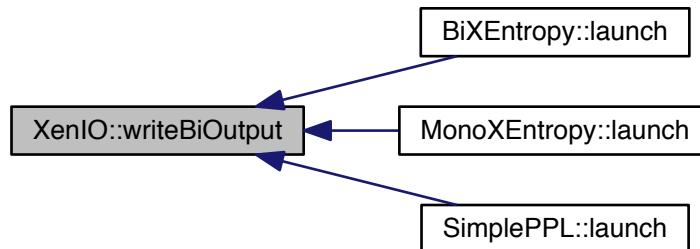
Parameters

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

Here is the call graph for this function:



Here is the caller graph for this function:



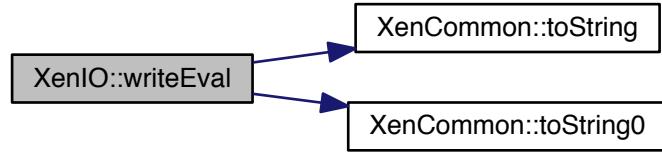
6.37.2.8 void XenIO::writeEval (boost::shared_ptr< EvalMap > ptrEvalMap, std::string distName) [static]

Writes an evaluation/best point distribution file.

Parameters

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

Here is the call graph for this function:



Here is the caller graph for this function:



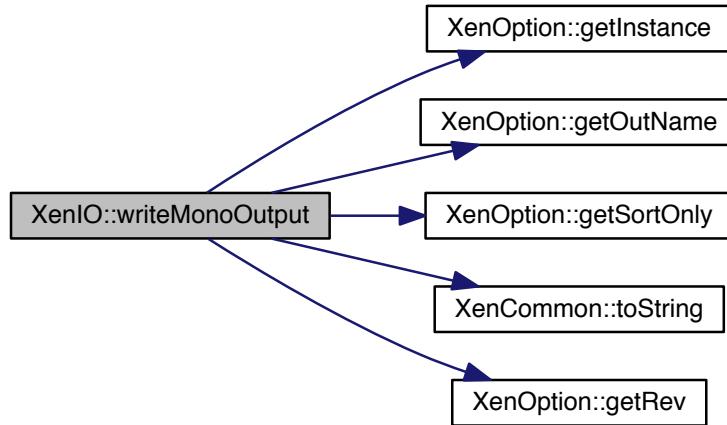
6.37.2.9 void XenIO::writeMonoOutput (boost::shared_ptr< Corpus > ptrCorp, boost::shared_ptr< Score > ptrScore) [static]

Writes monolingual scored/sorted result files.

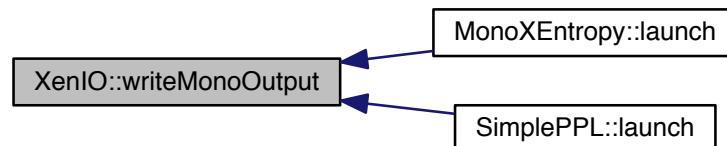
Parameters

<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:



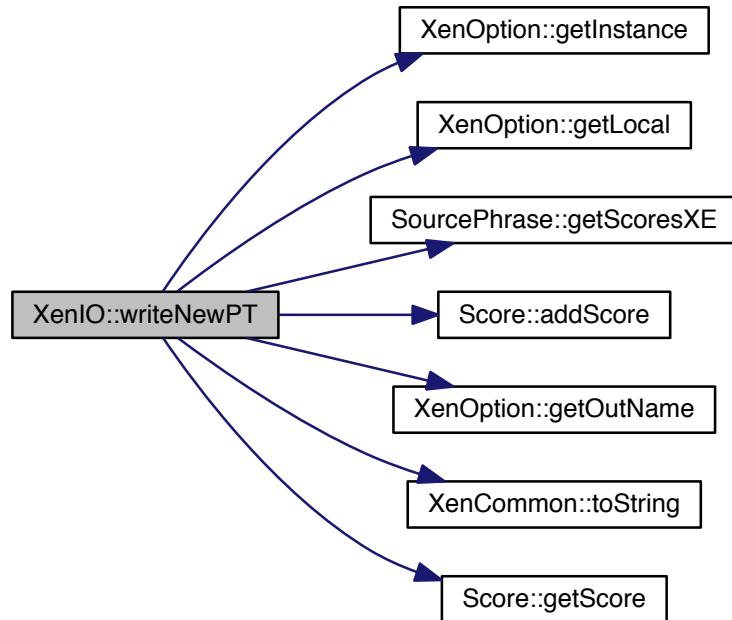
6.37.2.10 void `XenIO::writeNewPT (boost::shared_ptr< PhraseTable > ptrPT, boost::shared_ptr< Score > ptrScore)`
`[static]`

Writes a new rescored phrase-table.

Parameters

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Writes a phrase-table's source phrases.

Parameters

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

Returns

the written source phrases file name

Here is the caller graph for this function:



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

Writes a phrase-table's target phrases.

Parameters

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

Returns

the written target phrases file name

Here is the caller graph for this function:



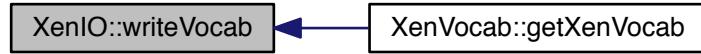
6.37.2.13 void XenIO::writeVocab (std::map<std::string, int> voc, std::string fileName) [static]

Writes a vocab to file.

Parameters

<code>ptrEvalMap</code>	: the map containing the vocab to write
<code>distName</code>	: the vocab file name

Here is the caller graph for this function:



6.37.2.14 void XenIO::writeXRpart (boost::shared_ptr< XenResult > ptrXR, int pc, std::string fileName = " ") [static]

Writes a part of the text of a [XenResult](#) given a percentage.

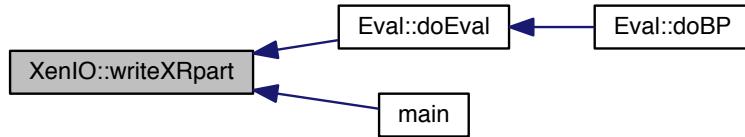
Parameters

<i>ptrXR</i>	: the XenResult file to extract the part from
<i>pc</i>	: the percentage to extract
<i>fileName</i>	: the file name for the extraction (optional)

Here is the call graph for this function:



Here is the caller graph for this function:



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

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

6.38 XenLMken Class Reference

Class handling KenLM estimation, loading, querying...

```
#include <XenLMken.h>
```

Public Member Functions

- [XenLMken \(\)](#)
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.
- [~XenLMken \(\)](#)
Default destructor.
- [int createLM \(\)](#)
Estimates a language model based on the provided data.
- [int loadLM \(\)](#)
Loads a language model from a provided file name.
- [std::string getFileName \(\) const](#)
Accessor to the language model file name.
- [TxtStats getSentenceStats \(std::string sent\)](#)
Computes the KenLM stats of a given sentence.
- [TxtStats getDocumentStats \(boost::shared_ptr< Corpus > c\)](#)
Computes the KenLM stats of a [Corpus](#) at a document level.

6.38.1 Detailed Description

Class handling KenLM estimation, loading, querying...

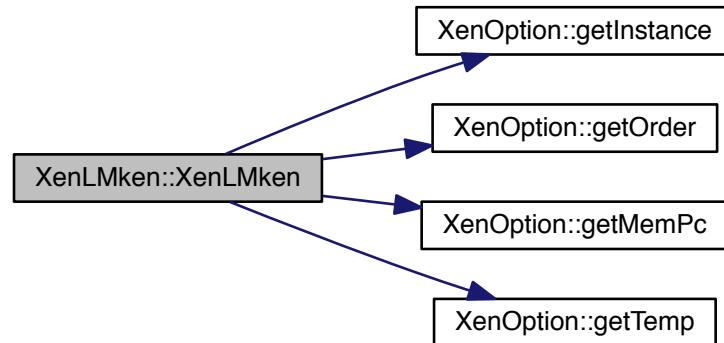
This class is in charge of handling all KenLM-related operations.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 XenLMken::XenLMken ()

Default constructor.

Here is the call graph for this function:



6.38.2.2 XenLMken::~XenLMken()

Default destructor.

6.38.3 Member Function Documentation

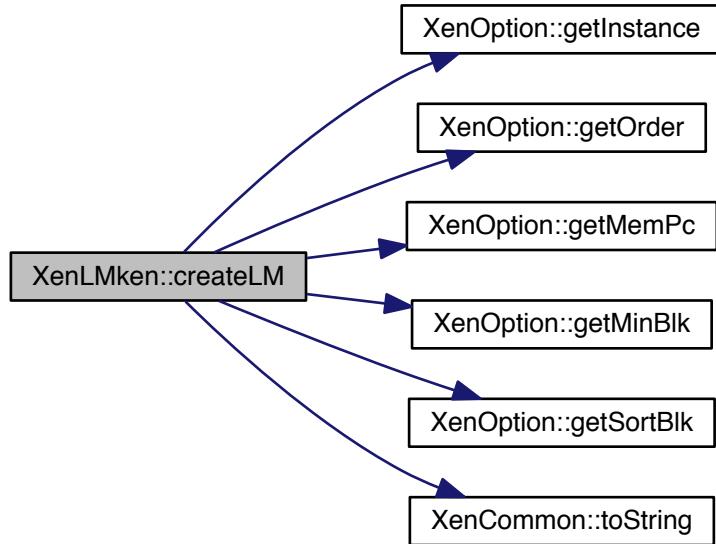
6.38.3.1 int XenLMken::createLM()

Estimates a language model based on the provided data.

Returns

0 if all goes well

Here is the call graph for this function:



6.38.3.2 `TxtStats XenLMken::getDocumentStats (boost::shared_ptr< Corpus > ptrCorp)`

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

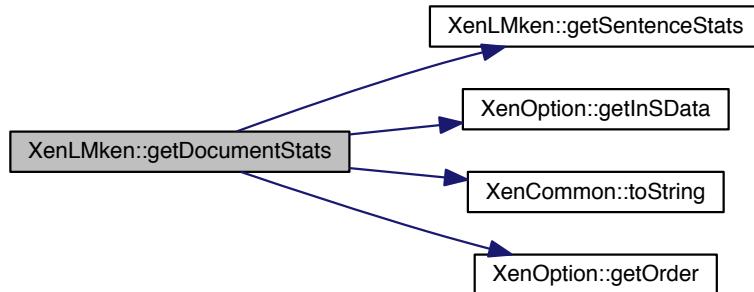
Parameters

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

Returns

the computed document-level KenLM stats

Here is the call graph for this function:

**6.38.3.3 std::string XenLMken::getFileName () const**

Accessor to the language model file name.

Returns

the language model file name

6.38.3.4 TxtStats XenLMken::getSentenceStats (std::string sent)

Computes the KenLM stats of a given sentence.

Parameters

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

Returns

the computed KenLM stats

Here is the caller graph for this function:



6.38.3.5 void XenLMken::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:



6.38.3.6 void XenLMken::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:



```
6.38.3.7 void XenLMken::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 <i>ptrXenRes</i> 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:



6.38.3.8 int XenLMken::loadLM ()

Loads a language model from a provided file name.

Returns

0 if all goes well

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

- include/XenLMken.h
- src/XenLMken.cpp

6.39 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](#)

- boost::shared_ptr< XenFile > **getWFile () const**
Accessor to the target language out-of-domain language model file.
- boost::shared_ptr< XenFile > **getDev () const**
Accessor to the weights file.
- int **getOrder () const**
Accessor to the development corpus file.
- std::size_t **getMemPc () const**
Accessor to the memory percentage for LM estimation.
- std::string **getTemp () const**
Accessor to the temporary files path.
- std::size_t **getMinBlk () const**
Accessor to the minimum block size.
- std::size_t **getSortBlk () const**
Accessor to the sort block size.
- bool **getExclOOVs () const**
Accessor to the OOVs exclusion state.
- 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.
- int **getMaxEvalPC () const**
- 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.

6.39.1 Detailed Description

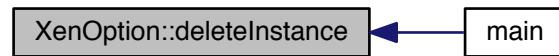
Singleton class handling XenC options accessors/mutators.

6.39.2 Member Function Documentation

6.39.2.1 void XenOption::deleteInstance () [static]

Deletes the unique instance of the `XenOption` singleton.

Here is the caller graph for this function:



6.39.2.2 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:



6.39.2.3 `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:



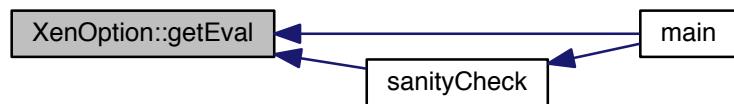
6.39.2.4 `bool XenOption::getEval() const`

Accessor to the evaluation execution state.

Returns

true if we have to perform an evaluation

Here is the caller graph for this function:



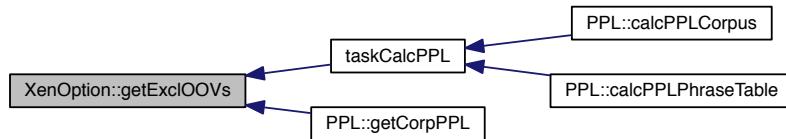
6.39.2.5 `bool XenOption::getExclOOVs() const`

Accessor to the OOVs exclusion state.

Returns

the OOVs exclusion state

Here is the caller graph for this function:

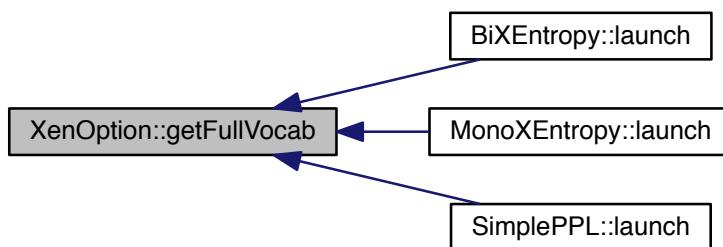
**6.39.2.6 bool XenOption::getFullVocab () const**

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:

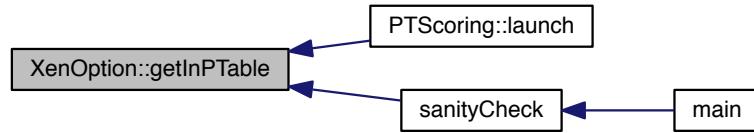
**6.39.2.7 boost::shared_ptr< XenFile > XenOption::getInPTable () const**

Accessor to the in-domain phrase-table file.

Returns

the in-domain phrase-table file

Here is the caller graph for this function:

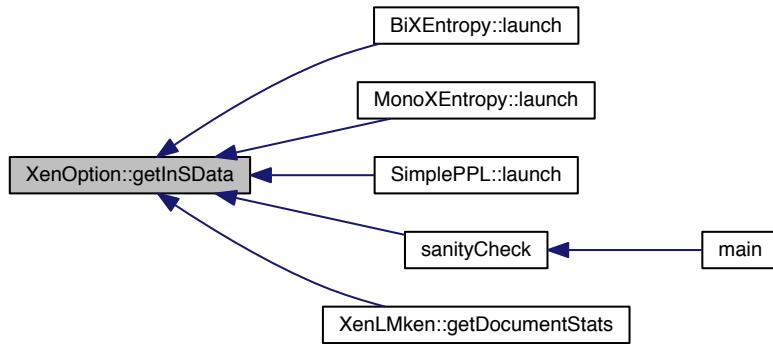
**6.39.2.8 boost::shared_ptr< XenFile > XenOption::getInSData() const**

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:

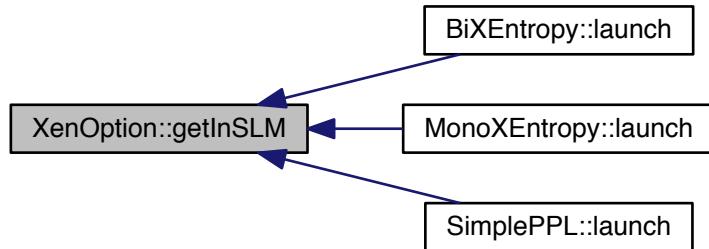
**6.39.2.9 boost::shared_ptr< XenFile > XenOption::getInSLM() const**

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:

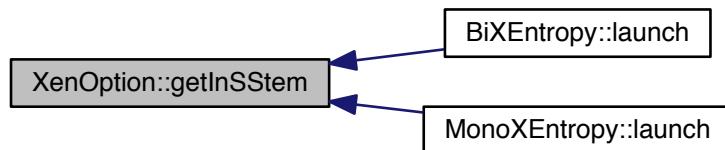
**6.39.2.10 boost::shared_ptr< XenFile > XenOption::getInSStem () const**

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:

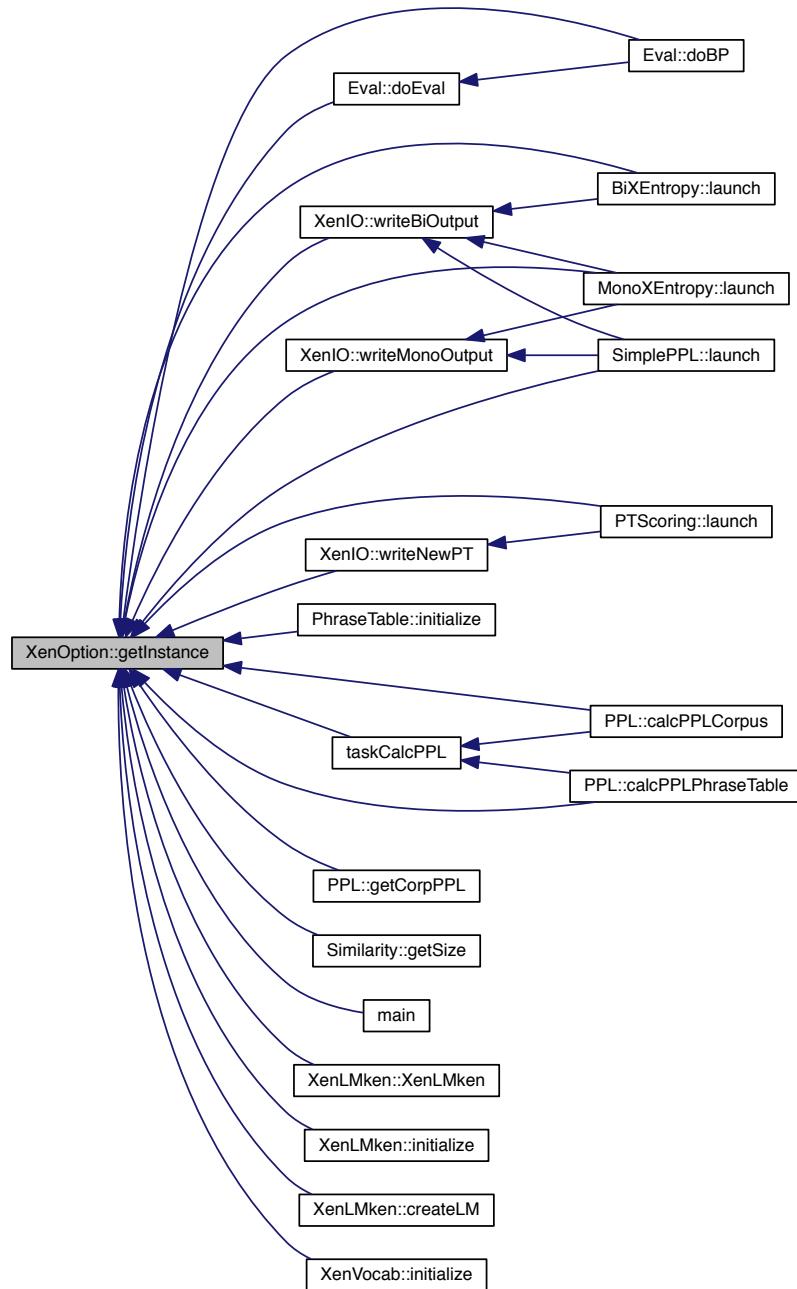
**6.39.2.11 XenOption * XenOption::getInstance () [static]**

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

Returns

the [XenOption](#) unique instance

Here is the caller graph for this function:



6.39.2.12 XenOption * XenOption::getInstance (LPOptions opt) [static]

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

Parameters

<i>opt</i>	: the LPOptions struct to build the XenOption object from
------------	---

Returns

the [XenOption](#) unique instance

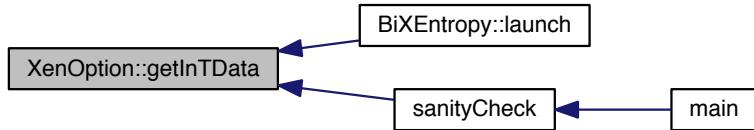
6.39.2.13 boost::shared_ptr< XenFile > XenOption::getInTData() const

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:

**6.39.2.14 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:



6.39.2.15 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:



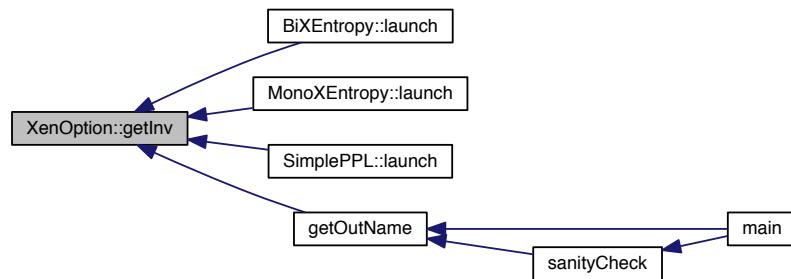
6.39.2.16 bool XenOption::getInv () const

Accessor to the inverted filtered output state.

Returns

true if we output inverted scores (1 - score)

Here is the caller graph for this function:



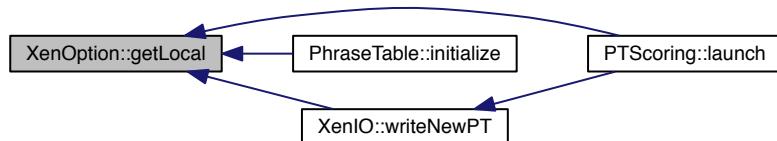
6.39.2.17 bool XenOption::getLocal () const

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:



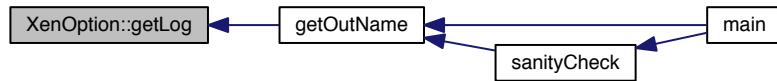
6.39.2.18 bool XenOption::getLog () const

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

Returns

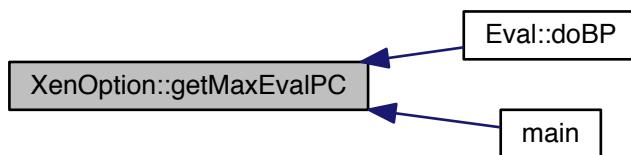
true if values in the weights file are in the log-domain

Here is the caller graph for this function:



6.39.2.19 int XenOption::getMaxEvalPC () const

Here is the caller graph for this function:



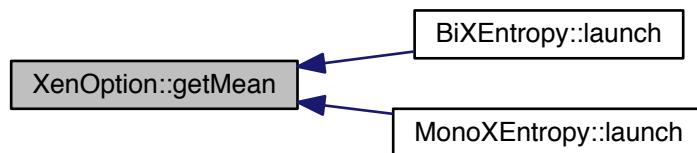
6.39.2.20 bool XenOption::getMean () const

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:

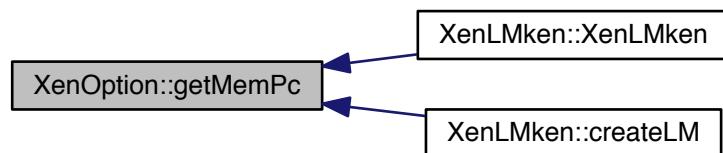
**6.39.2.21 std::size_t XenOption::getMemPc () const**

Accessor to the memory percentage for LM estimation.

Returns

the memory percentage for LM estimation

Here is the caller graph for this function:



6.39.2.22 std::size_t XenOption::getMinBlk() const

Accessor to the minimum block size.

Returns

the minimum block size

Here is the caller graph for this function:

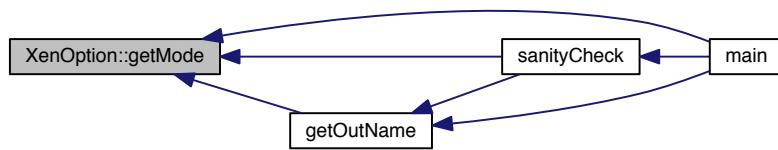
**6.39.2.23 int XenOption::getMode() const**

Accessor to the filtering mode.

Returns

the filtering mode

Here is the caller graph for this function:

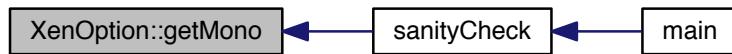
**6.39.2.24 bool XenOption::getMono() const**

Accessor to the monolingual or bilingual execution state.

Returns

true if we work on monolingual data

Here is the caller graph for this function:

**6.39.2.25 std::string XenOption::getName() const**

Accessor to the program name.

Returns

the program name

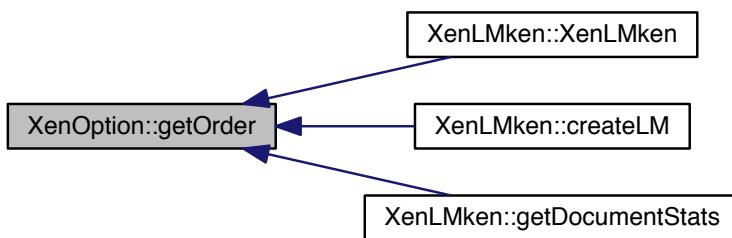
6.39.2.26 int XenOption::getOrder() const

Accessor to the order for language models estimation.

Returns

the order for language models estimation

Here is the caller graph for this function:



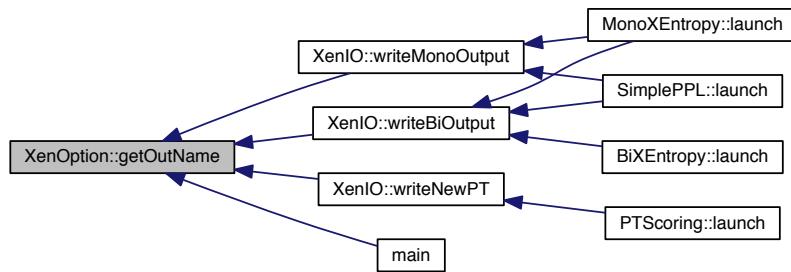
6.39.2.27 std::string XenOption::getOutName() const

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:

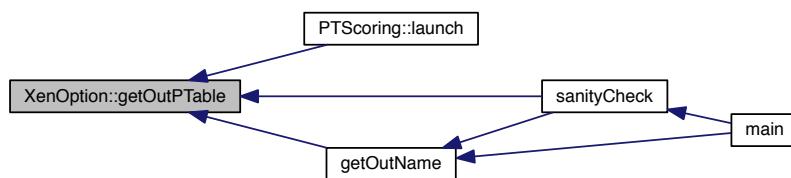
**6.39.2.28 boost::shared_ptr< XenFile > XenOption::getOutPTable() const**

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:



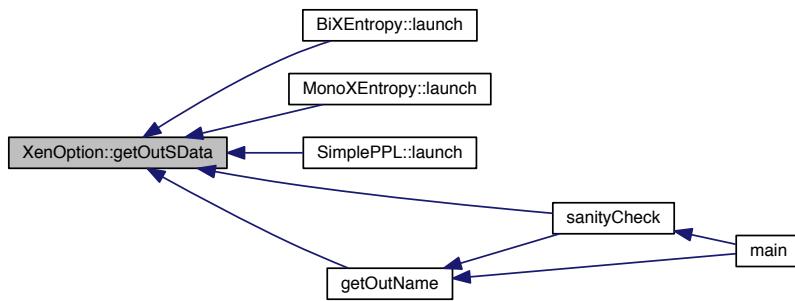
6.39.2.29 boost::shared_ptr< XenFile > XenOption::getOutSData() const

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:



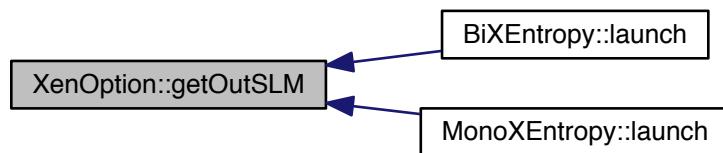
6.39.2.30 boost::shared_ptr< XenFile > XenOption::getOutSLM() const

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:



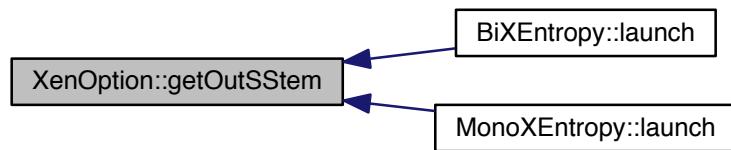
6.39.2.31 boost::shared_ptr< XenFile > XenOption::getOutSStem() const

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:

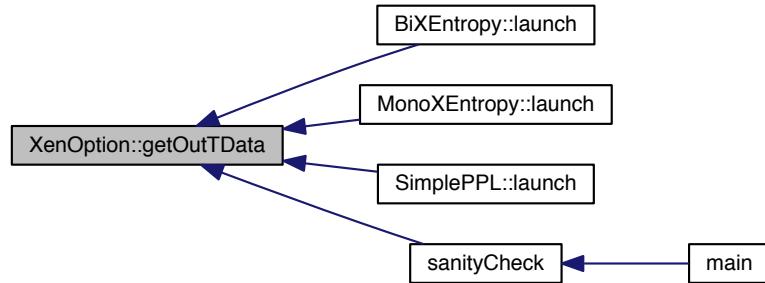
**6.39.2.32 boost::shared_ptr< XenFile > XenOption::getOutTData() const**

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:



6.39.2.33 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:

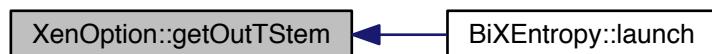
**6.39.2.34 boost::shared_ptr< XenFile > XenOption::getOutTStem () const**

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:

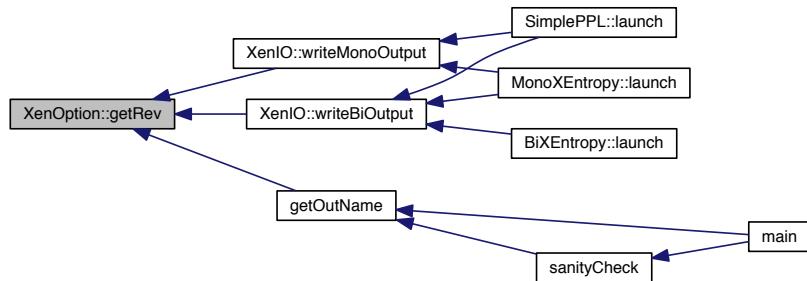
**6.39.2.35 bool XenOption::getRev () const**

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:

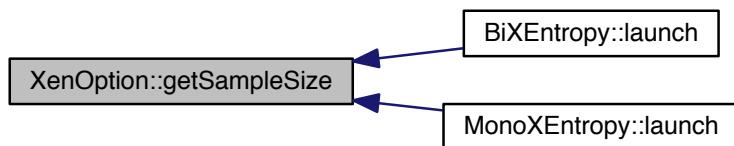
**6.39.2.36 int XenOption::getSampleSize () const**

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:

**6.39.2.37 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:

**6.39.2.38 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:

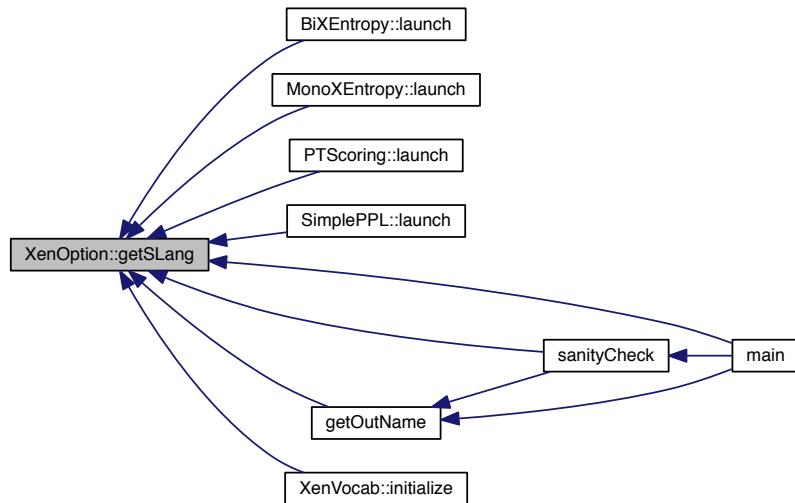
**6.39.2.39 std::string XenOption::getSLang () const**

Accessor to the source language.

Returns

the source language

Here is the caller graph for this function:

**6.39.2.40 std::size_t XenOption::getSortBlk() const**

Accessor to the sort block size.

Returns

the sort block size

Here is the caller graph for this function:



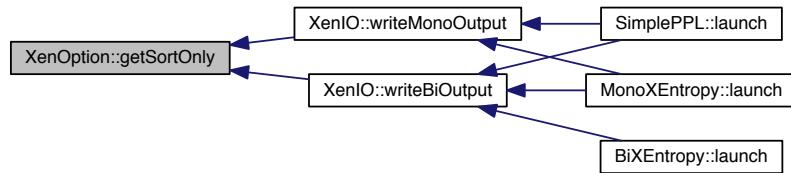
6.39.2.41 bool XenOption::getSortOnly () const

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:



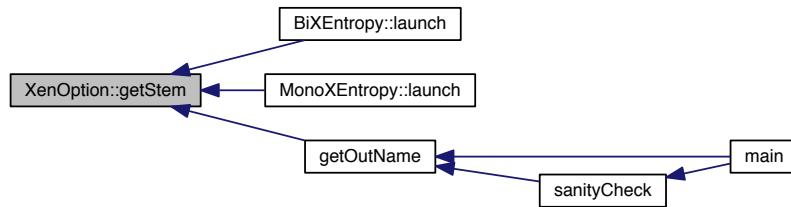
6.39.2.42 bool XenOption::getStem () const

Accessor to the stem mode execution state.

Returns

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

Here is the caller graph for this function:



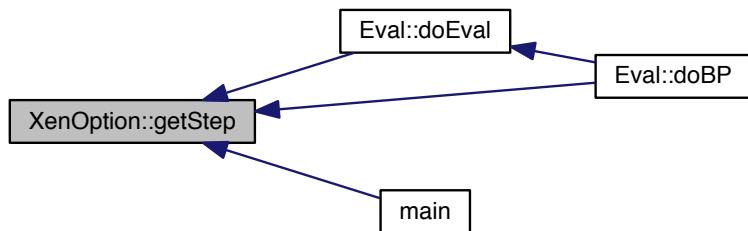
6.39.2.43 int XenOption::getStep () const

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:

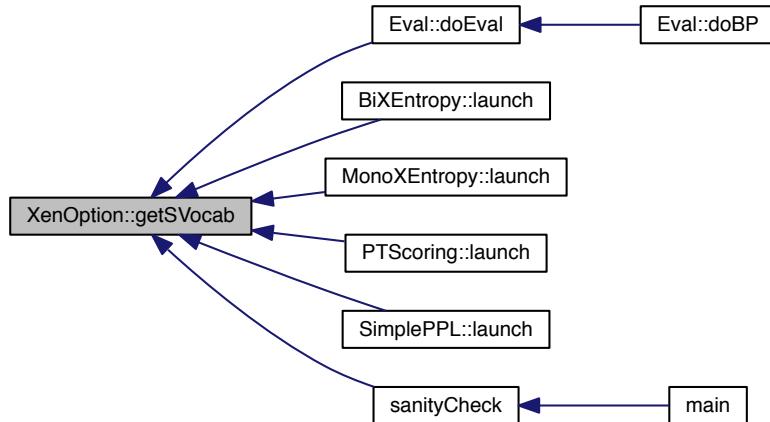
**6.39.2.44 boost::shared_ptr< XenFile > XenOption::getSVocab () const**

Accessor to the source language vocabulary file.

Returns

the source language vocabulary file

Here is the caller graph for this function:



6.39.2.45 std::string XenOption::getTemp() const

Accessor to the temporary files path.

Returns

the temporary files path

Here is the caller graph for this function:

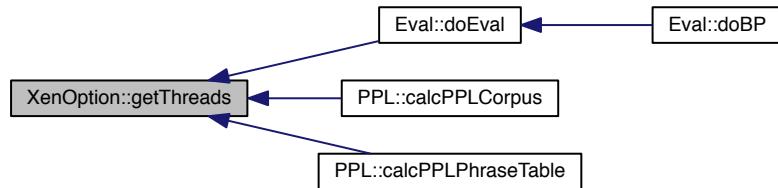
**6.39.2.46 int XenOption::getThreads() const**

Accessor to the requested number of threads.

Returns

the requested number of threads

Here is the caller graph for this function:

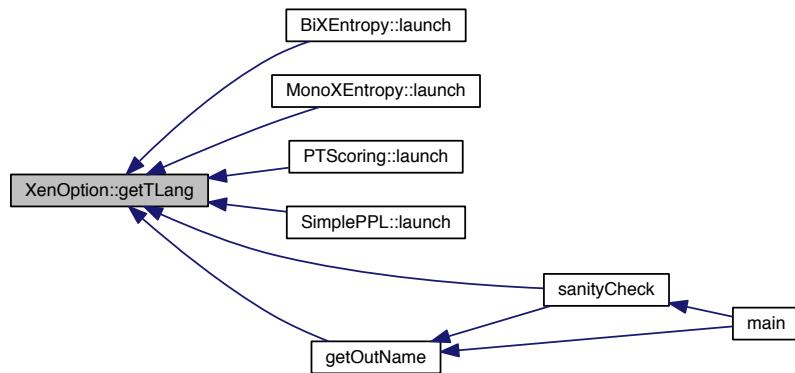
**6.39.2.47 std::string XenOption::getTLang() const**

Accessor to the target language.

Returns

the target language

Here is the caller graph for this function:

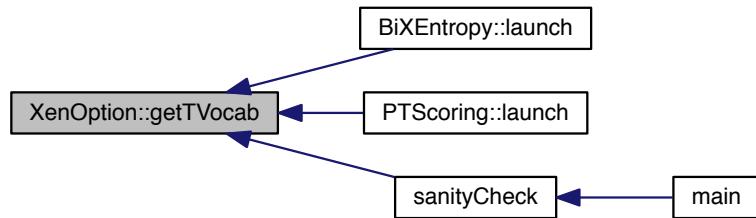
**6.39.2.48 boost::shared_ptr< XenFile > XenOption::getTVocab () const**

Accessor to the target language vocabulary file.

Returns

the target language vocabulary file

Here is the caller graph for this function:



6.39.2.49 int XenOption::getVecSize () const

Accessor to the similarity measures vector size.

Returns

the similarity measures vector size

Here is the caller graph for this function:



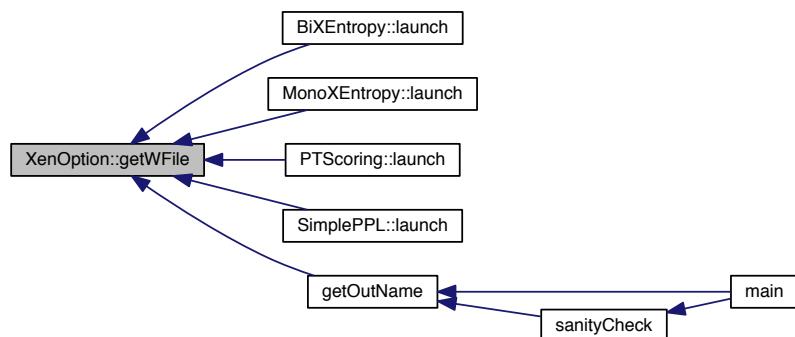
6.39.2.50 boost::shared_ptr< XenFile > XenOption::getWFile () const

Accessor to the weights file.

Returns

the weights file

Here is the caller graph for this function:



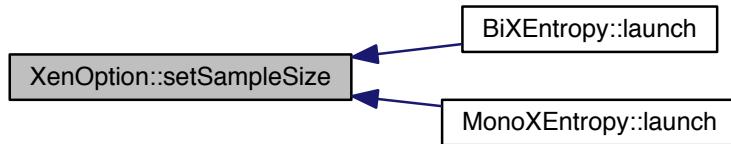
6.39.2.51 void XenOption::setSampleSize (int size)

Mutator to the out-of-domain sample size.

Parameters

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

Here is the caller graph for this function:



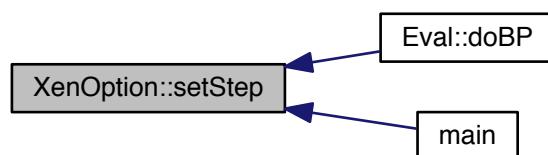
6.39.2.52 void XenOption::setStep (int step)

Mutator to the evaluation/best point step size.

Parameters

<code>step</code>	: 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

6.40 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 nth 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.

6.40.1 Detailed Description

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

6.40.2 Constructor & Destructor Documentation

6.40.2.1 XenResult::XenResult ()

Default constructor.

6.40.2.2 XenResult::~XenResult ()

Default destructor.

6.40.3 Member Function Documentation

6.40.3.1 unsigned int XenResult::getSize () const

Accessor to the size of the sorted corpus text.

Returns

the size of the sorted corpus text

6.40.3.2 std::vector< std::string > XenResult::getSortedText () const

Accessor to the sorted corpus text.

Returns

the sorted corpus text

6.40.3.3 std::string XenResult::getTextLine (int n)

Accessor to the nth line of the sorted corpus text.

Parameters

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

Returns

the requested nth line of text

6.40.3.4 boost::shared_ptr< XenFile > XenResult::getXenFile() const

Accessor to the sorted result file.

Returns

the sorted result file

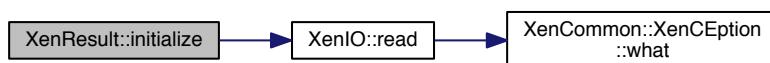
6.40.3.5 void XenResult::initialize(boost::shared_ptr< XenFile > ptrFile)

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

6.41 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.
- `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.

6.41.1 Detailed Description

Class handling a XenC vocabulary.

6.41.2 Constructor & Destructor Documentation

6.41.2.1 XenVocab::XenVocab()

Default constructor.

6.41.2.2 XenVocab::~XenVocab()

Default destructor.

6.41.3 Member Function Documentation

6.41.3.1 unsigned int XenVocab::getSize() const

Accessor to the size of the vocabulary text.

Returns

the size of the vocabulary text

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

Accessor to the vocabulary file.

Returns

the vocabulary file

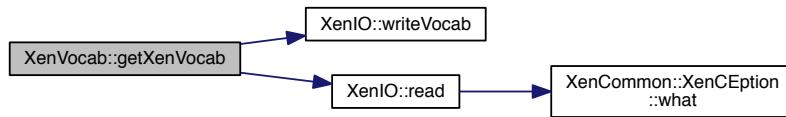
6.41.3.3 `std::map< std::string, int > XenVocab::getXenVocab() const`

Accessor to the XenC vocabulary object.

Returns

the XenC vocabulary object

Here is the call graph for this function:

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

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

Parameters

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

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

Initialization function from a [Corpus](#).

Parameters

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

6.41.3.6 void XenVocab::initialize (boost::shared_ptr<Corpus> ptrInCorp, boost::shared_ptr<Corpus> ptrOutCorp)

Initialization function from two [Corpus](#).

Parameters

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

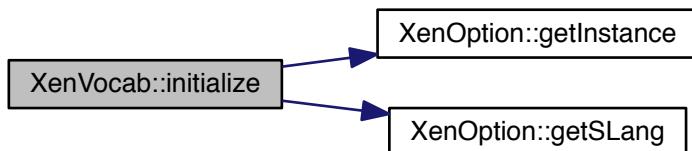
6.41.3.7 void XenVocab::initialize (boost::shared_ptr<XenResult> ptrXenRes)

Initialization function from a sorted result file.

Parameters

<code>ptrXenRes</code>	: 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](#)

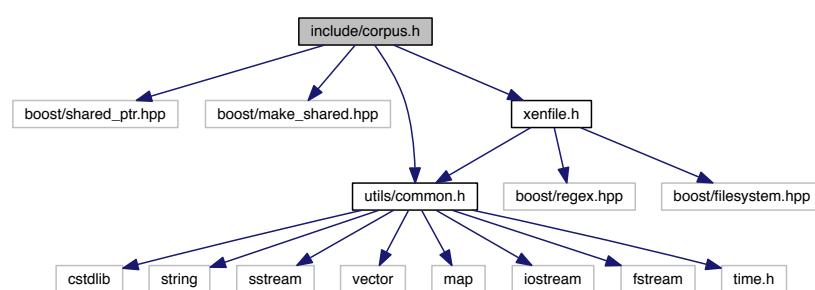
Chapter 7

File Documentation

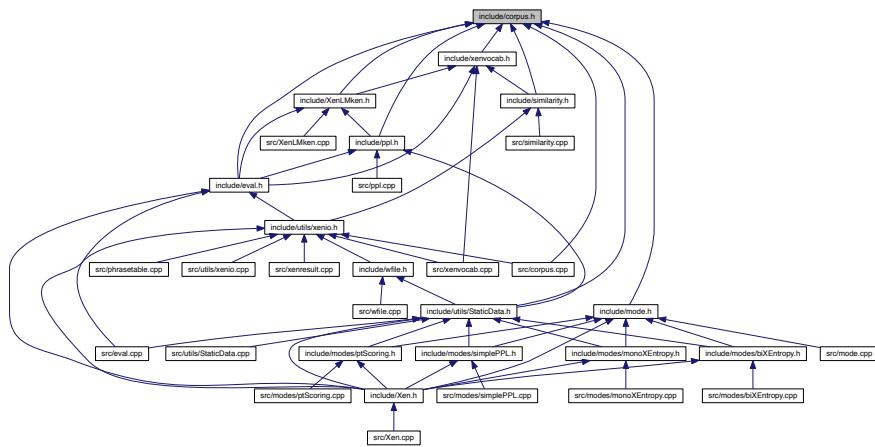
7.1 include/corpus.h File Reference

Class handling corpus-related functionalities.

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



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



Classes

- class **Corpus**

Corpus-related functionalities.

7.1.1 Detailed Description

Class handling corpus-related functionalities.

Author

Anthony Rousseau

Version

2.0.0

Date

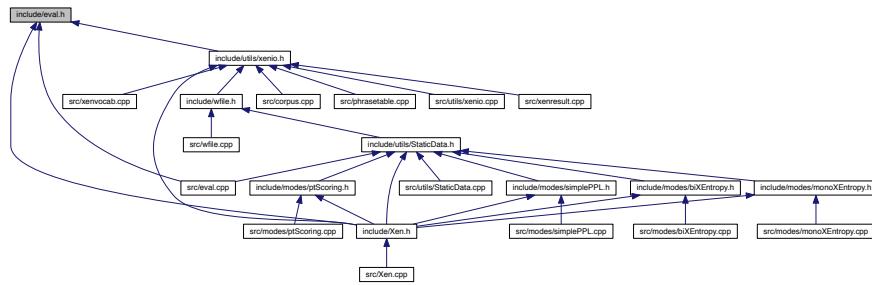
18 March 2016

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 "XenLMken.h"
#include "ppl.h"
#include "xenresult.h"
#include "xenvocab.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< Corpus > ptrCorp, 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

2.0.0

Date

18 March 2016

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

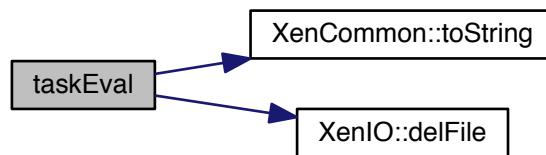
7.2.3.1 `void taskEval (int pc, boost::shared_ptr<Corpus> ptrCorp, boost::shared_ptr<XenVocab> ptrVoc, boost::shared_ptr<Corpus> ptrDevCorp, boost::shared_ptr<EvalMap> ptrDist)`

Thread-safe evaluation function.

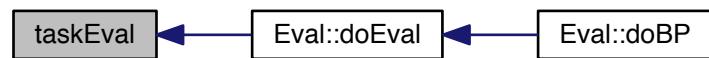
Parameters

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

Here is the call graph for this function:

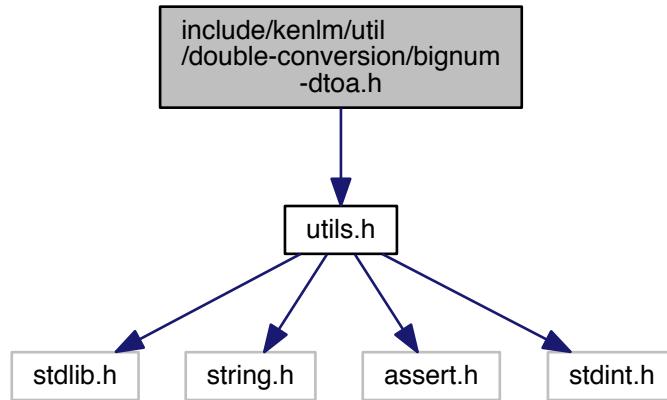


Here is the caller graph for this function:



7.3 include/kenlm/util/double-conversion/bignum-dtoa.h File Reference

```
#include "utils.h"
Include dependency graph for bignum-dtoa.h:
```



Namespaces

- [double_conversion](#)

Enumerations

- enum [double_conversion::BignumDtoaMode](#) { [double_conversion::BIGNUM_DTOA_SHORTEST](#), [double_conversion::BIGNUM_DTOA_SHORTEST_SINGLE](#), [double_conversion::BIGNUM_DTOA_FIXED](#), [double_conversion::BIGNUM_DTOA_PRECISION](#) }

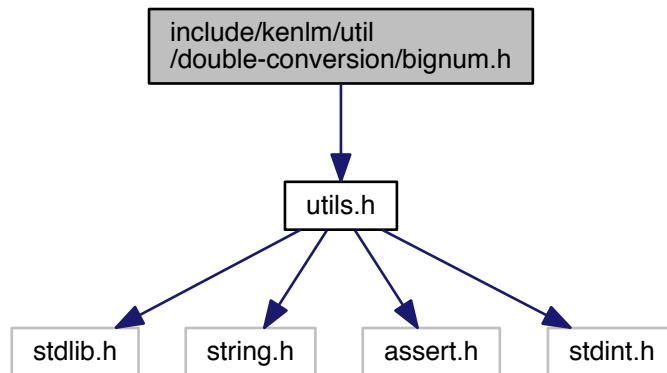
Functions

- void [double_conversion::BignumDtoa](#) (double v, BignumDtoaMode mode, int requested_digits, Vector< char > buffer, int *length, int *point)

7.4 include/kenlm/util/double-conversion/bignum.h File Reference

```
#include "utils.h"
```

Include dependency graph for bignum.h:



Classes

- class [double_conversion::Bignum](#)

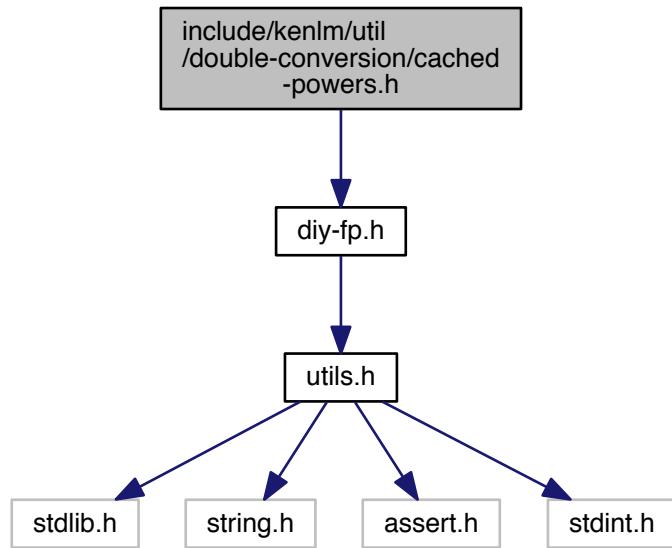
Namespaces

- [double_conversion](#)

7.5 include/kenlm/util/double-conversion/cached-powers.h File Reference

```
#include "diy-fp.h"
```

Include dependency graph for cached-powers.h:



Classes

- class [double_conversion::PowersOfTenCache](#)

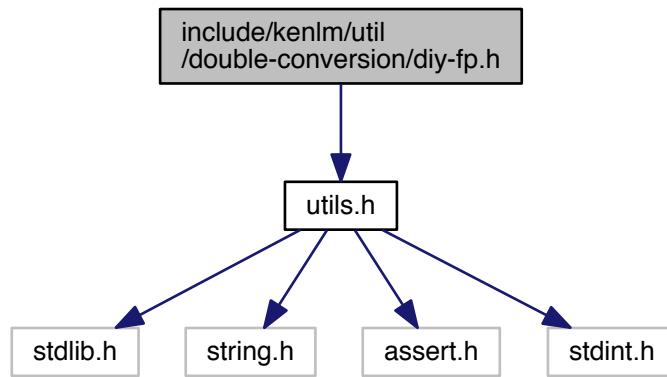
Namespaces

- [double_conversion](#)

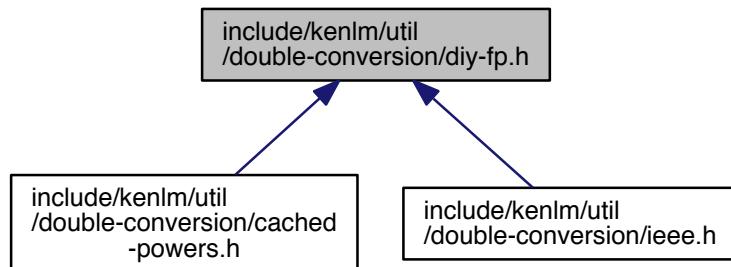
7.6 include/kenlm/util/double-conversion/diy-fp.h File Reference

```
#include "utils.h"
```

Include dependency graph for diy-fp.h:



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



Classes

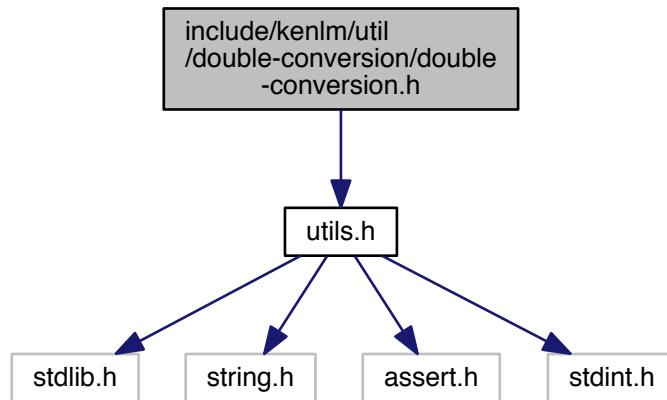
- class [double_conversion::DiyFp](#)

Namespaces

- [double_conversion](#)

7.7 include/kenlm/util/double-conversion/double-conversion.h File Reference

```
#include "utils.h"  
Include dependency graph for double-conversion.h:
```



Classes

- class [double_conversion::DoubleToStringConverter](#)
- class [double_conversion::StringToDoubleConverter](#)

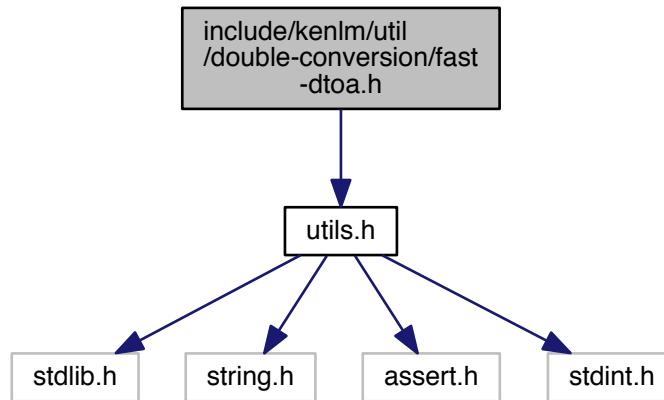
Namespaces

- [double_conversion](#)

7.8 include/kenlm/util/double-conversion/fast-dtoa.h File Reference

```
#include "utils.h"
```

Include dependency graph for fast-dtoa.h:



Namespaces

- `double_conversion`

Enumerations

- enum `double_conversion::FastDtoaMode` { `double_conversion::FAST_DTOA_SHORTEST`, `double_conversion::FAST_DTOA_SHORTEST_SINGLE`, `double_conversion::FAST_DTOA_PRECISION` }

Functions

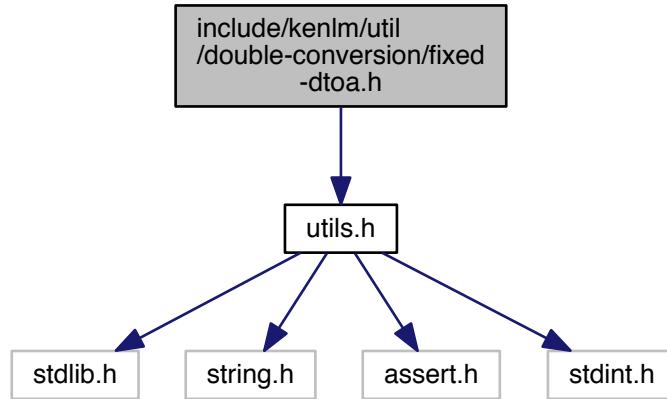
- bool `double_conversion::FastDtoa` (`double d`, `FastDtoaMode mode`, `int requested_digits`, `Vector< char > buffer`, `int *length`, `int *decimal_point`)

Variables

- static const int `double_conversion::kFastDtoaMaximalLength` = 17
- static const int `double_conversion::kFastDtoaMaximalSingleLength` = 9

7.9 include/kenlm/util/double-conversion/fixed-dtoa.h File Reference

```
#include "utils.h"
Include dependency graph for fixed-dtoa.h:
```



Namespaces

- [double_conversion](#)

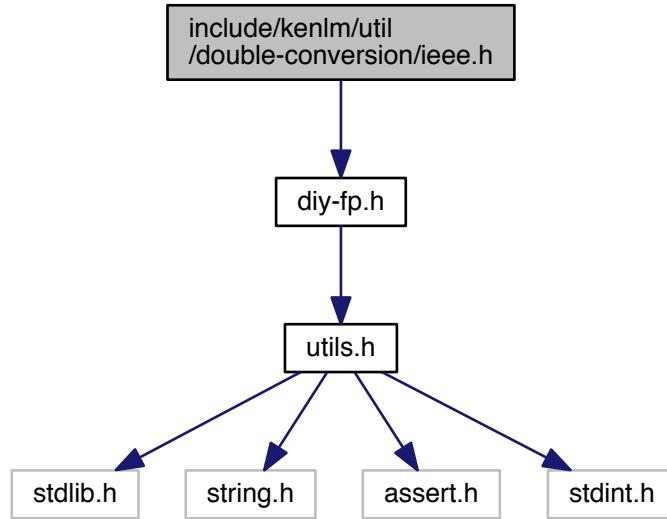
Functions

- bool [double_conversion::FastFixedDtoa](#) (double v, int fractional_count, Vector< char > buffer, int *length, int *decimal_point)

7.10 include/kenlm/util/double-conversion/ieee.h File Reference

```
#include "diy-fp.h"
```

Include dependency graph for ieee.h:



Classes

- class [double_conversion::Double](#)
- class [double_conversion::Single](#)

Namespaces

- [double_conversion](#)

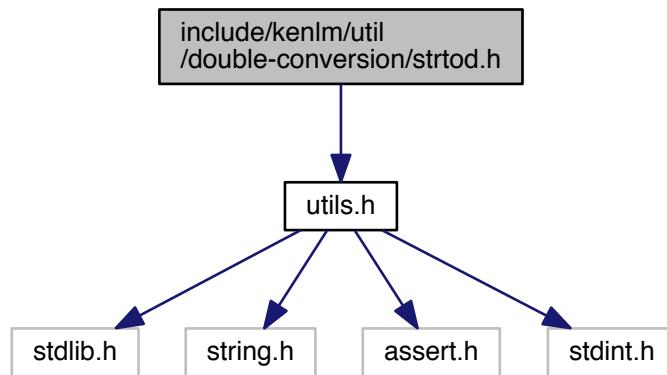
Functions

- static uint64_t [double_conversion::double_to_uint64](#) (double d)
- static double [double_conversion::uint64_to_double](#) (uint64_t d64)
- static uint32_t [double_conversion::float_to_uint32](#) (float f)
- static float [double_conversion::uint32_to_float](#) (uint32_t d32)

7.11 include/kenlm/util/double-conversion/strtod.h File Reference

```
#include "utils.h"
```

Include dependency graph for strtod.h:



Namespaces

- [double_conversion](#)

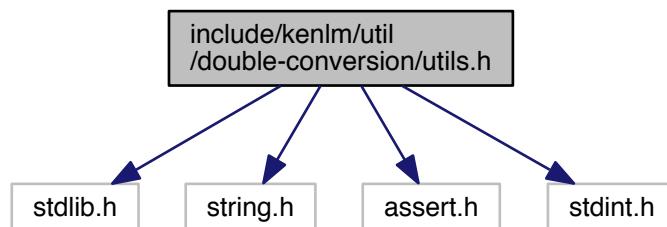
Functions

- `double double_conversion::Strtod (Vector< const char > buffer, int exponent)`
- `float double_conversion::Strtof (Vector< const char > buffer, int exponent)`

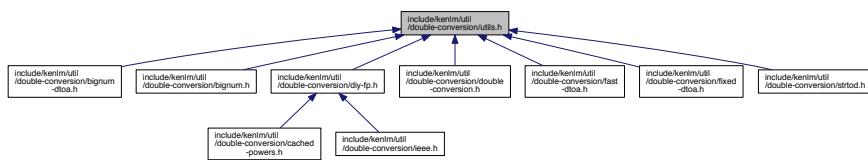
7.12 include/kenlm/util/double-conversion/utils.h File Reference

```
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <stdint.h>
```

Include dependency graph for utils.h:



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



Classes

- class `double_conversion::Vector< T >`
- class `double_conversion::StringBuilder`

Namespaces

- `double_conversion`

Macros

- `#define ASSERT(condition) (assert(condition))`
- `#define UNIMPLEMENTED() (abort())`
- `#define UNREACHABLE() (abort())`
- `#define UINT64_2PART_C(a, b) (((static_cast<uint64_t>(a) << 32) + 0x##b##u))`
- `#define ARRAY_SIZE(a)`
- `#define DISALLOW_COPY_AND_ASSIGN(TypeName)`
- `#define DISALLOW_IMPLICIT_CONSTRUCTORS(TypeName)`

Functions

- template<typename T >
 `static T double_conversion::Max (T a, T b)`
- template<typename T >
 `static T double_conversion::Min (T a, T b)`
- `int double_conversion::StrLength (const char *string)`
- template<class Dest , class Source >
 `Dest double_conversion::BitCast (const Source &source)`
- template<class Dest , class Source >
 `Dest double_conversion::BitCast (Source *source)`

Variables

- static const int `double_conversion::kCharSize = sizeof(char)`

7.12.1 Macro Definition Documentation

7.12.1.1 `#define ARRAY_SIZE(a)`

Value:

```
((sizeof(a) / sizeof(*((a)))) / \
static_cast<size_t>(! (sizeof(a) % sizeof(*((a)))))) \
```

7.12.1.2 `#define ASSERT(condition) (assert(condition))`

7.12.1.3 `#define DISALLOW_COPY_AND_ASSIGN(TypeName)`

Value:

```
TypeName (const TypeName&); \
void operator=(const TypeName&) \
```

7.12.1.4 `#define DISALLOW_IMPLICIT_CONSTRUCTORS(TypeName)`

Value:

```
TypeName (); \
DISALLOW_COPY_AND_ASSIGN(TypeName) \
```

7.12.1.5 `#define UINT64_2PART_C(a, b) (((static_cast<uint64_t>(a) << 32) + 0x##b##u))`

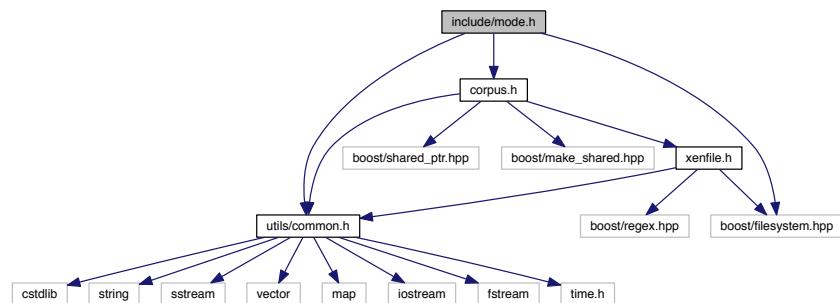
7.12.1.6 `#define UNIMPLEMENTED() (abort())`

7.12.1.7 `#define UNREACHABLE() (abort())`

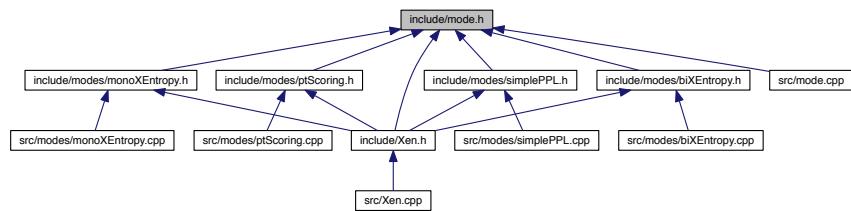
7.13 include/mode.h File Reference

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.

7.13.1 Detailed Description

Abstract class defining the filtering modes architecture.

Author

Anthony Rousseau

Version

2.0.0

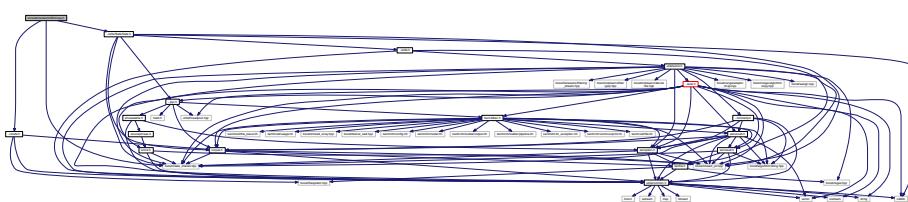
Date

18 March 2016

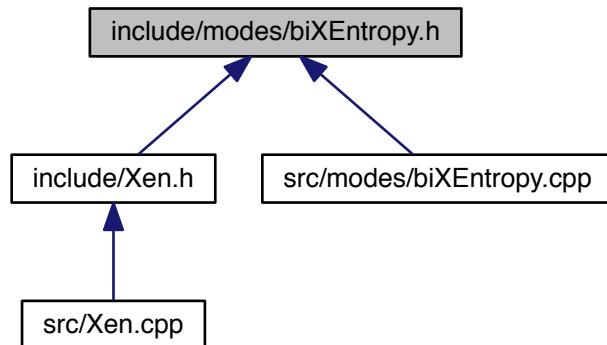
7.14 include/modes/biXEntropy.h File Reference

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.

7.14.1 Detailed Description

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

Author

Anthony Rousseau

Version

2.0.0

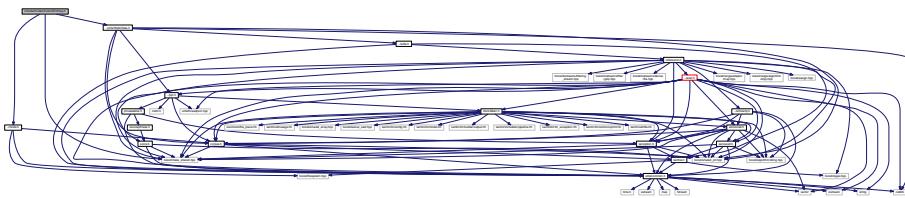
Date

18 March 2016

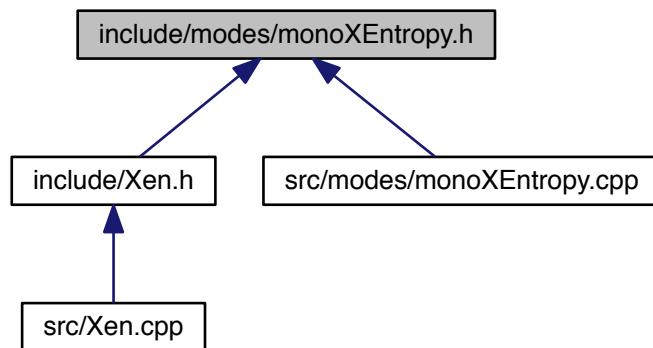
7.15 include/modes/monoXEntropy.h File Reference

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.

7.15.1 Detailed Description

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

Author

Anthony Rousseau

Version

2.0.0

Date

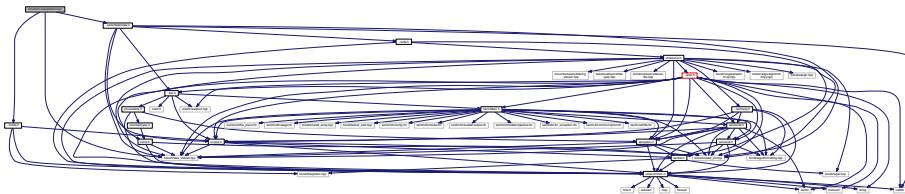
18 March 2016

7.16 include/modes/ptScoring.h File Reference

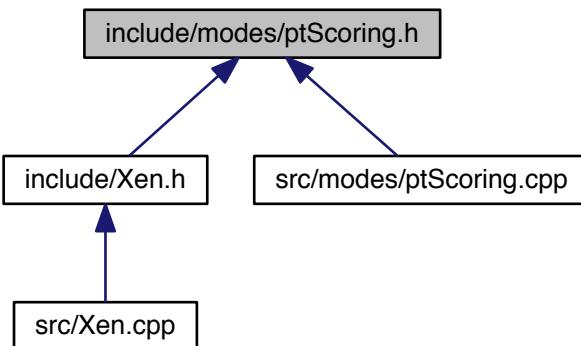
Derived class to handle filtering mode 4: phrase-table cross-entropy.

```
#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.

7.16.1 Detailed Description

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

Author

Anthony Rousseau

Version

2.0.0

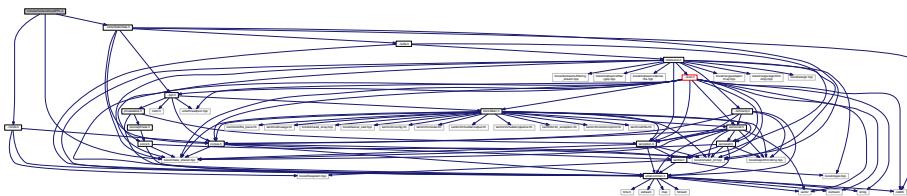
Date

18 March 2016

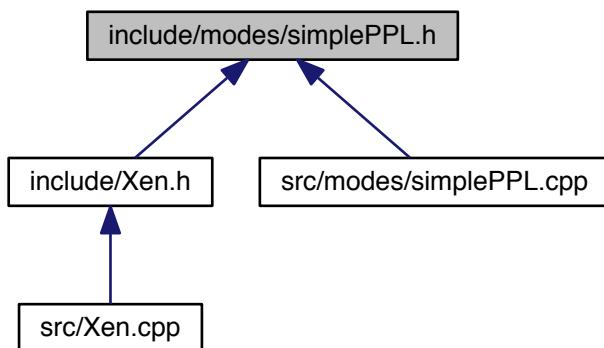
7.17 include/modes/simplePPL.h File Reference

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.

7.17.1 Detailed Description

Derived class to handle filtering mode 1: simple perplexity.

Author

Anthony Rousseau

Version

2.0.0

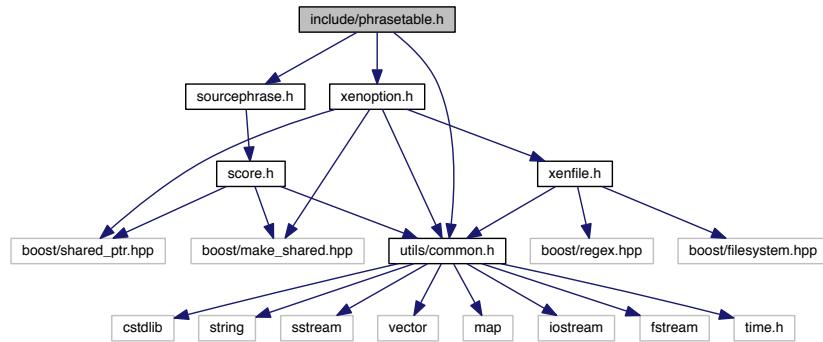
Date

18 March 2016

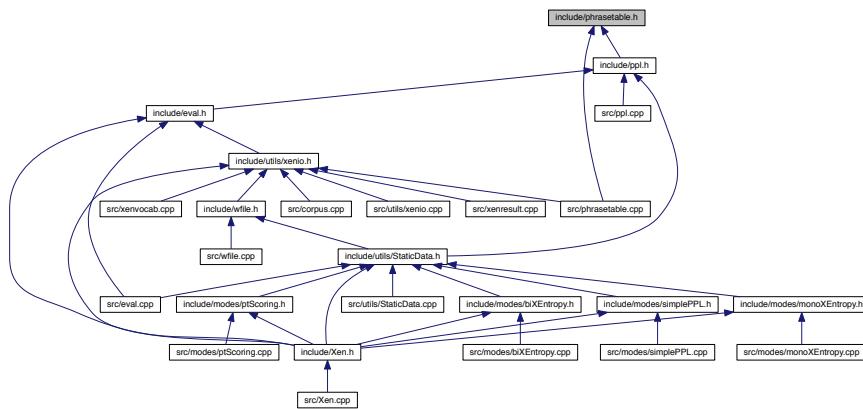
7.18 include/phrasetable.h File Reference

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:



Classes

- class [PhraseTable](#)

Class handling phrase-table related functionalities.

7.18.1 Detailed Description

Class handling phrase-table related functionalities.

Author

Anthony Rousseau

Version

2.0.0

Date

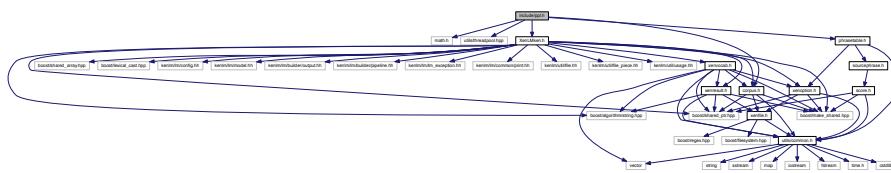
18 March 2016

7.19 include/ppl.h File Reference

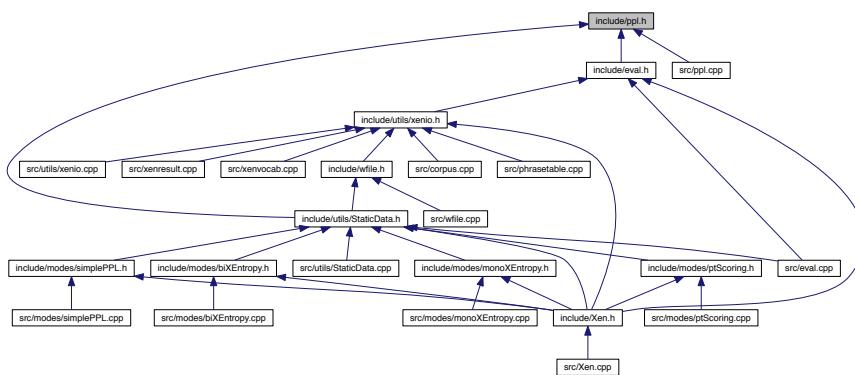
Class handling the perplexity/cross-entropy computations.

```
#include <math.h>
#include "utils/threadpool.hpp"
#include "corpus.h"
#include "phrasetable.h"
#include "XenLMken.h"
```

Include dependency graph for ppl.h:



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



Classes

- class [PPL](#)

Perplexity/Cross-entropy computations.

Macros

- `#define M_LN10 2.30258509299404568402`

Functions

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

Thread-safe perplexity computation function.

7.19.1 Detailed Description

Class handling the perplexity/cross-entropy computations.

Author

Anthony Rousseau

Version

2.0.0

Date

18 March 2016

7.19.2 Macro Definition Documentation

7.19.2.1 `#define M_LN10 2.30258509299404568402`

7.19.3 Function Documentation

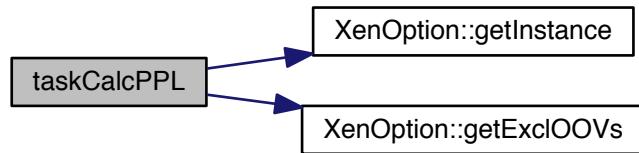
7.19.3.1 `void taskCalcPPL (int numLine, std::string line, boost::shared_ptr< std::vector< double > > ptrPPL, boost::shared_ptr< XenLMken > ptrLM)`

Thread-safe perplexity computation function.

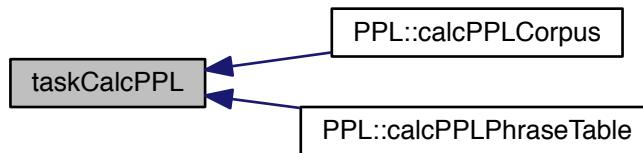
Parameters

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

Here is the call graph for this function:



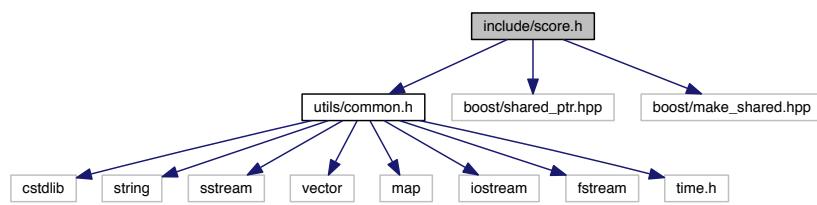
Here is the caller graph for this function:



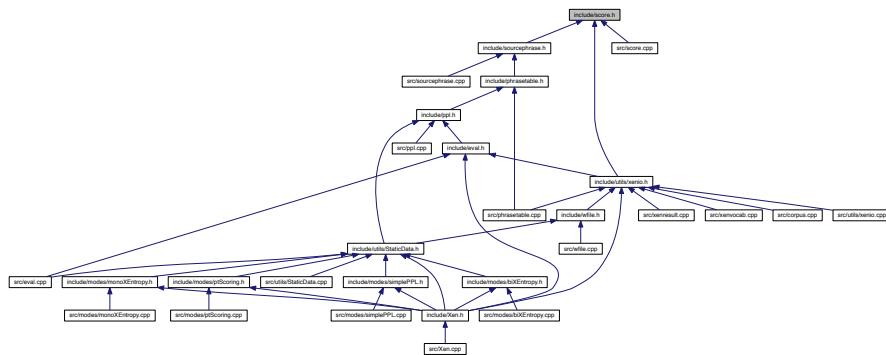
7.20 include/score.h File Reference

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:
```



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



Classes

- class Score

Class holding the XenC scores representation.

7.20.1 Detailed Description

Class holding the XenC scores representation.

Author

Anthony Rousseau

Version

2.0.0

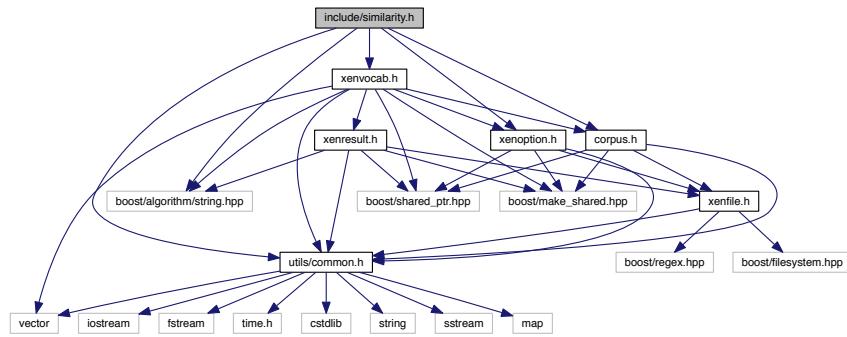
Date

18 March 2016

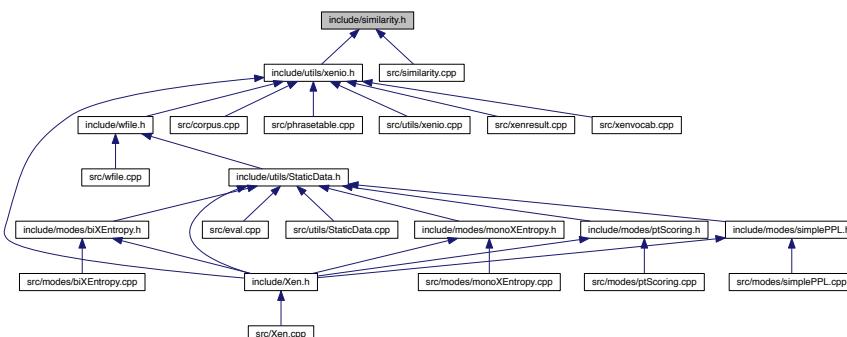
7.21 include/similarity.h File Reference

Class taking care of all the similarity measure computations.

```
#include <boost/algorithm/string.hpp>
#include "utils/common.h"
#include "corpus.h"
#include "xenvocab.h"
#include "xenoption.h"
Include dependency graph for similarity.h:
```



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



Classes

- class [Similarity](#)

Class taking care of all the similarity measure computations.

TypeDefs

- `typedef std::map< int, float > SimMap`

Map of integers as keys and floats as values to represent the similarity measures by sentence number.

7.21.1 Detailed Description

Class taking care of all the similarity measure computations.

Author

Anthony Rousseau

Version

2.0.0

Date

18 March 2016

7.21.2 Typedef Documentation

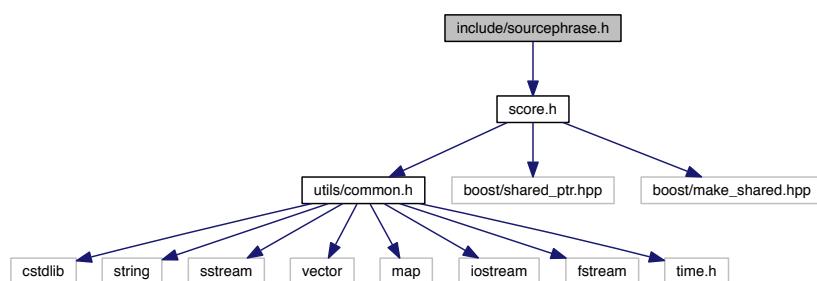
7.21.2.1 `typedef std::map<int, float> SimMap`

Map of integers as keys and floats as values to represent the similarity measures by sentence number.

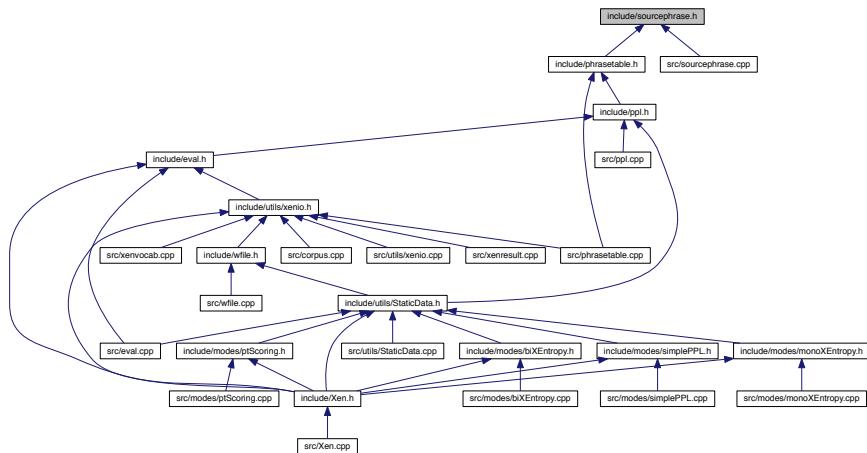
7.22 include/sourcephrase.h File Reference

Class holding a merged source phrase and all associated data.

```
#include "score.h"
Include dependency graph for sourcephrase.h:
```



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



Classes

- class [SourcePhrase](#)

Class holding a merged source phrase and all associated data.

7.22.1 Detailed Description

Class holding a merged source phrase and all associated data.

Author

Anthony Rousseau

Version

2.0.0

Date

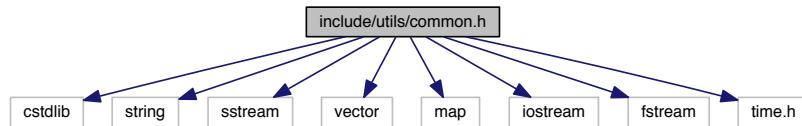
18 March 2016

7.23 include/utils/common.h File Reference

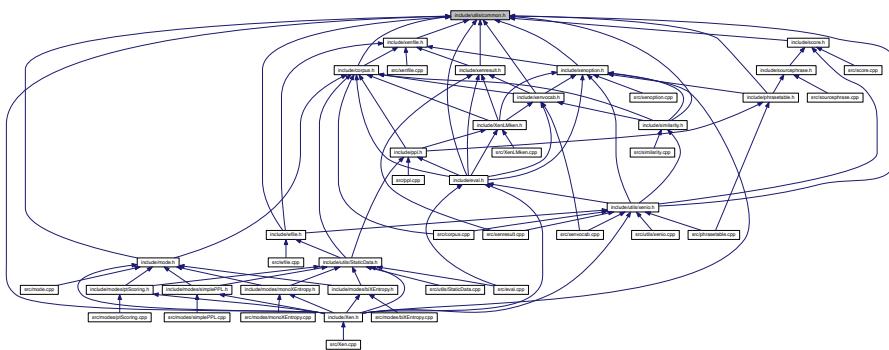
File containing all common classes/structures/functions of many classes of XenC.

```
#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

- [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 an 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.

7.23.1 Detailed Description

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

Author

Anthony Rousseau

Version

2.0.0

Date

18 March 2016

7.23.2 Typedef Documentation

7.23.2.1 `typedef struct _Options * LPOptions`

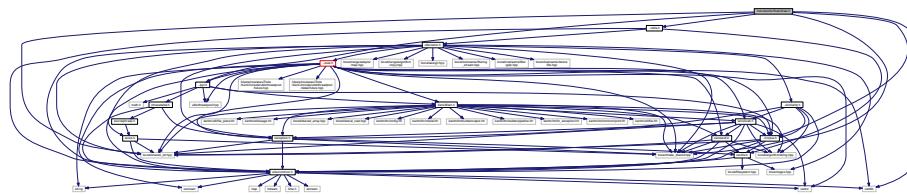
7.23.2.2 `typedef struct _Options Options`

7.24 include/utils/StaticData.h File Reference

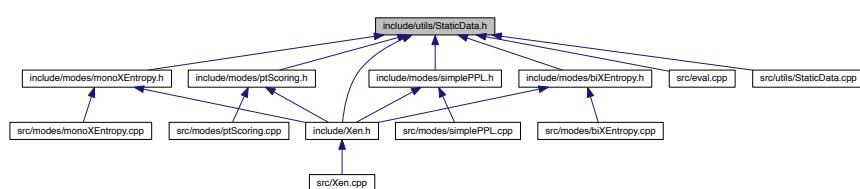
File handling all data objects used by XenC in a static way.

```
#include <cstdlib>
#include <boost/shared_ptr.hpp>
#include <boost/make_shared.hpp>
#include "../corpus.h"
#include "../ppl.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](#)

- class [MeanPPLPair](#)
Tiny class holding two additional LMs 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.

Macros

- `#define STATICDATA_H_`

7.24.1 Detailed Description

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

Author

Anthony Rousseau

Version

2.0.0

Date

18 March 2016

7.24.2 Macro Definition Documentation

7.24.2.1 `#define STATICDATA_H_`

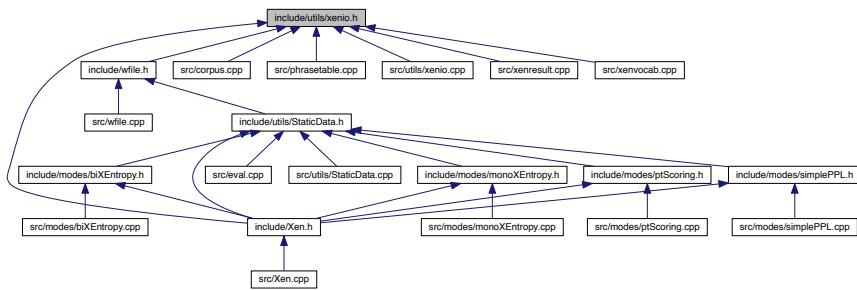
7.25 include/utils/xenio.h File Reference

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/range/adaptor/map.hpp>
#include <boost/range/algorithm/copy.hpp>
#include <boost/assign.hpp>
#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.

Macros

- `#define XENIO_H_`

Typedefs

- `typedef std::map< int, double, std::greater< int > > EvalMap`
descending ordered map on integers as keys and doubles as values

7.25.1 Detailed Description

Class handling all input/output operations of XenC.

Author

Anthony Rousseau

Version

2.0.0

Date

18 March 2016

7.25.2 Macro Definition Documentation

7.25.2.1 #define XENIO_H_

7.25.3 Typedef Documentation

7.25.3.1 typedef std::map<int, double, std::greater<int> > EvalMap

descending ordered map on integers as keys and doubles as values

7.26 include/wfile.h File Reference

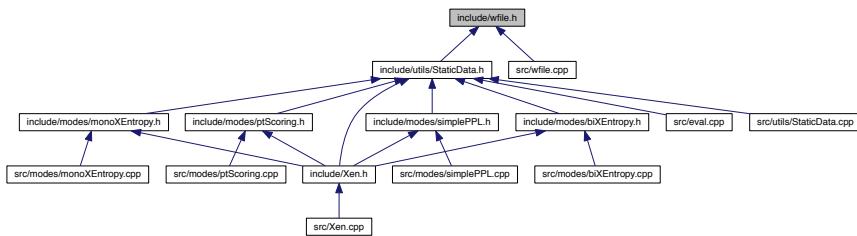
Class handling a file with values intended at weighting XenC scores.

```
#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.

7.26.1 Detailed Description

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

Author

Anthony Rousseau

Version

2.0.0

Date

18 March 2016

7.27 include/Xen.h File Reference

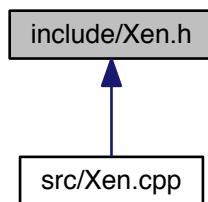
Main file of XenC, controls execution.

```
#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.

7.27.1 Detailed Description

Main file of XenC, controls execution.

Author

Anthony Rousseau

Version

2.0.0

Date

18 March 2016

7.27.2 Function Documentation

7.27.2.1 std::string getOutName (XenOption * opt)

Computes the output file name.

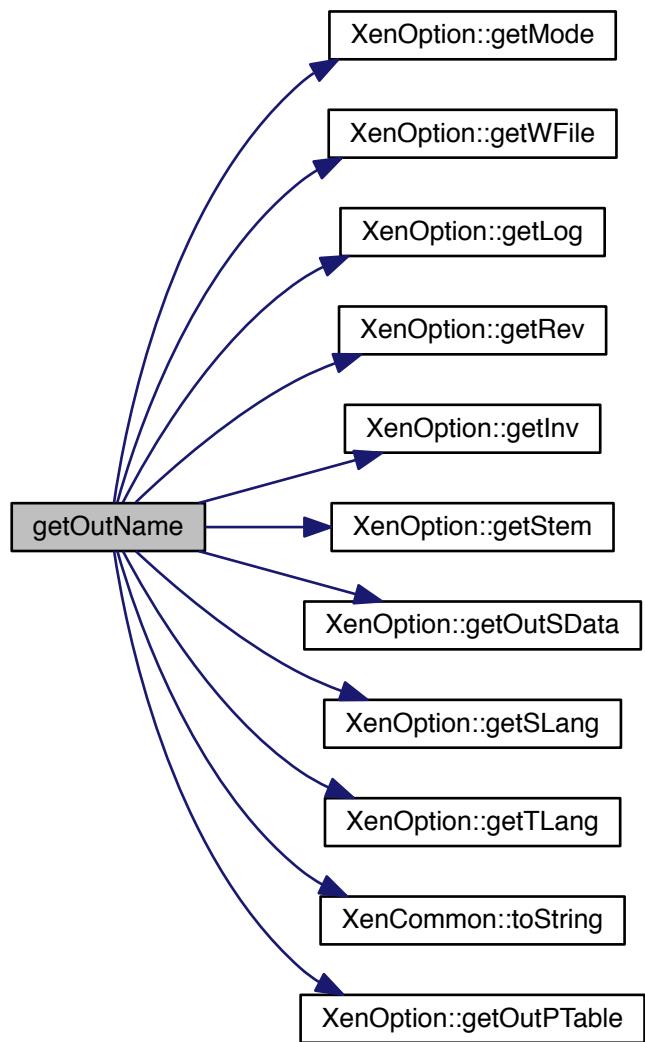
Parameters

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

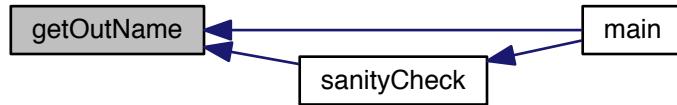
Returns

the output file name

Here is the call graph for this function:



Here is the caller graph for this function:



7.27.2.2 int main (int *argc*, char * *argv*[])

Main function of XenC.

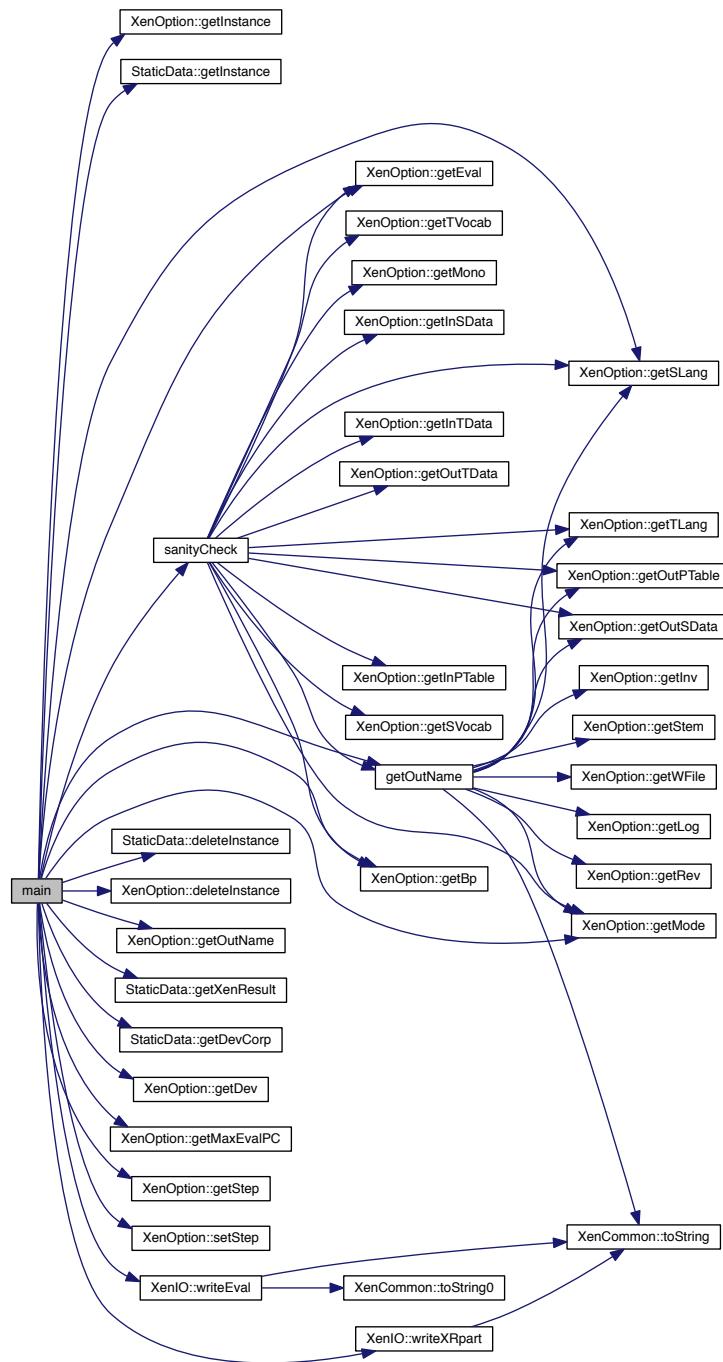
Parameters

<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:



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

Controls the mandatory options.

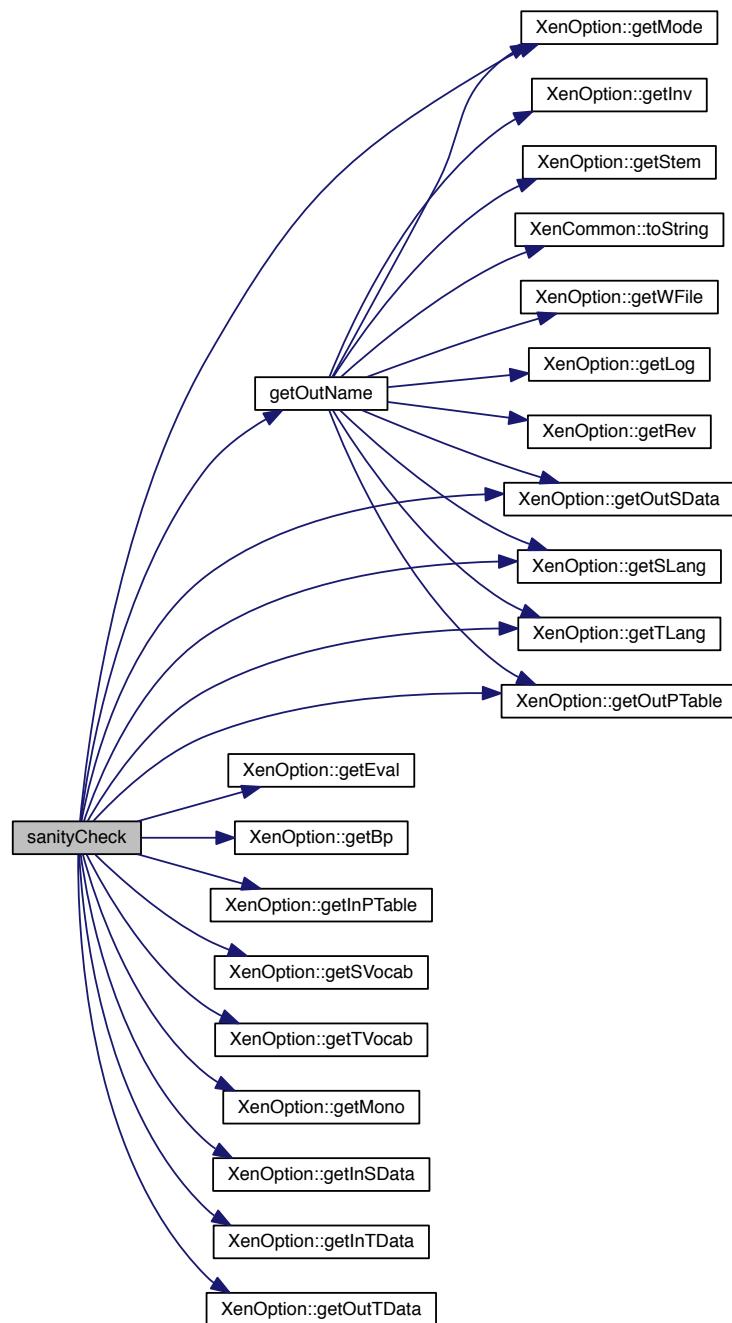
Parameters

<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:



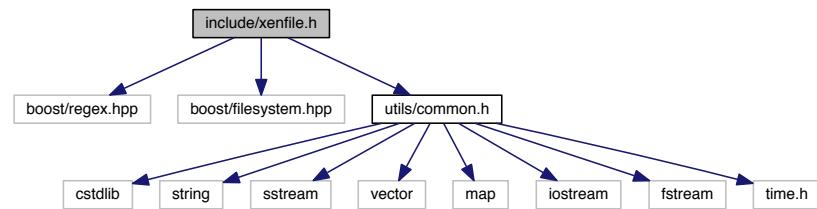
Here is the caller graph for this function:



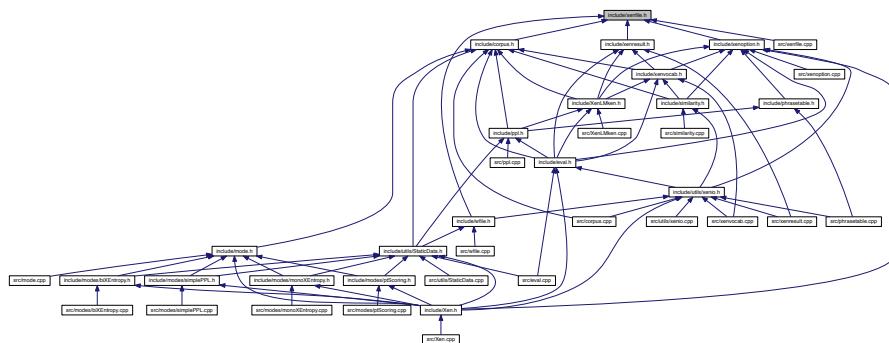
7.28 include/xenfile.h File Reference

Class providing some basic functions around files.

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

7.28.1 Detailed Description

Class providing some basic functions around files.

Author

Anthony Rousseau

Version

2.0.0

Date

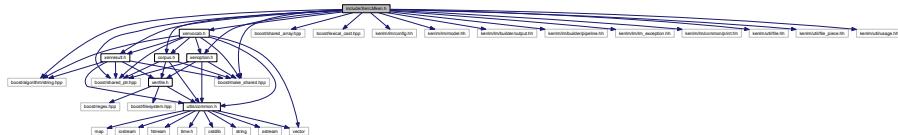
18 March 2016

7.29 include/XenLMken.h File Reference

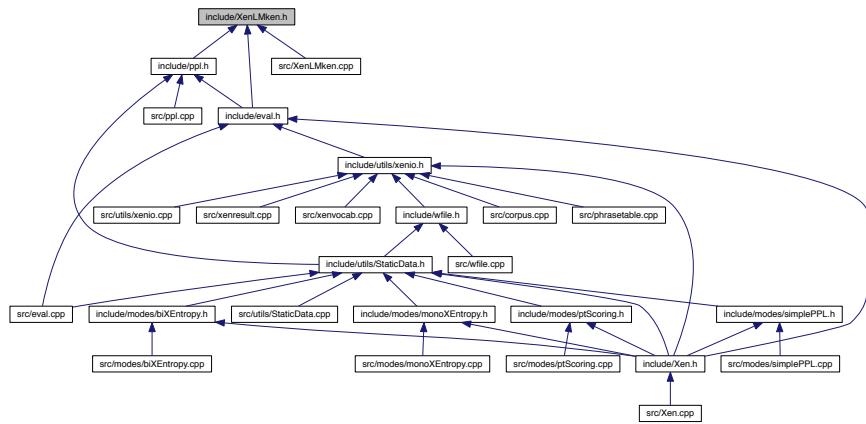
Class handling KenLM estimation, loading, querying...

```
#include <boost/algorithm/string.hpp>
#include <boost/shared_array.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/make_shared.hpp>
#include <boost/lexical_cast.hpp>
#include "kenlm/lm/config.hh"
#include "kenlm/lm/model.hh"
#include "kenlm/lm/builder/output.hh"
#include "kenlm/lm/builder/pipeline.hh"
#include "kenlm/lm/lm_exception.hh"
#include "kenlm/lm/common/print.hh"
#include "kenlm/util/file.hh"
#include "kenlm/util/file_piece.hh"
#include "kenlm/util/usage.hh"
#include "corpus.h"
#include "xenvocab.h"
#include "xenoption.h"
#include "xenresult.h"
Include dependency graph for XenLMken.h:
```

Include dependency graph for XenLMken.h:



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



Classes

- struct [TxtStats](#)
 - class [XenLMken](#)
- Class handling KenLM estimation, loading, querying...*

Macros

- `#define KENLM_MAX_ORDER 6`

7.29.1 Detailed Description

Class handling KenLM estimation, loading, querying...

Author

Anthony Rousseau

Version

2.0.0

Date

18 March 2016

7.29.2 Macro Definition Documentation

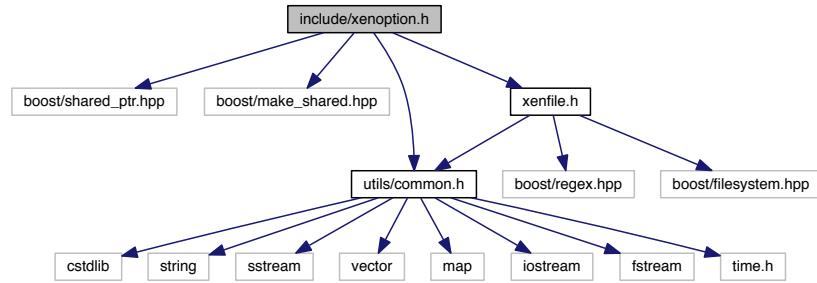
7.29.2.1 #define KENLM_MAX_ORDER 6

7.30 include/xenoption.h File Reference

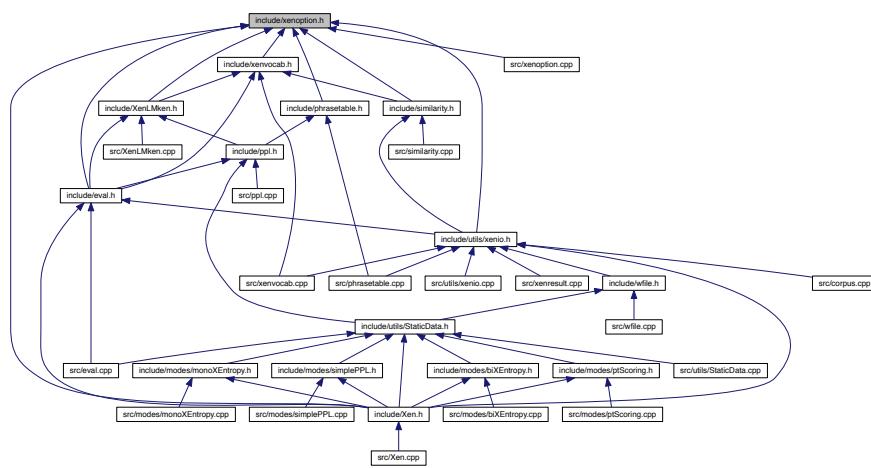
Singleton class handling XenC options accessors/mutators.

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

Include dependency graph for xenoption.h:



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



Classes

- class XenOption

Singleton class handling XenC options accessors/mutators.

7.30.1 Detailed Description

Singleton class handling XenC options accessors/mutators.

Author

Anthony Rousseau

Version

2.0.0

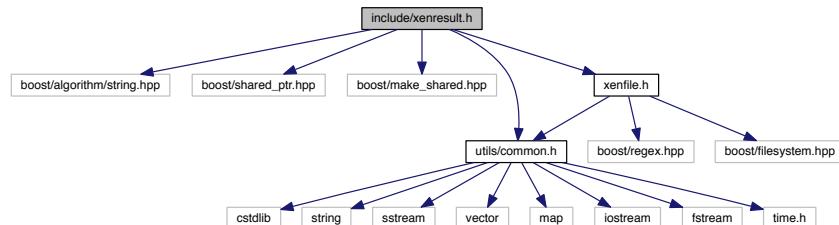
Date

18 March 2016

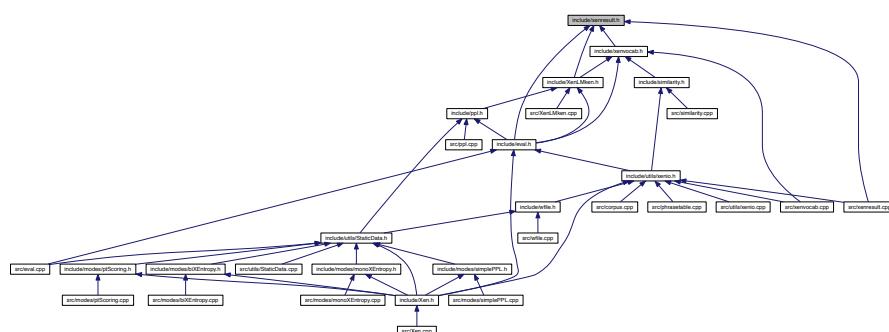
7.31 include/xenresult.h File Reference

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:
```



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



Classes

- class [XenResult](#)

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

7.31.1 Detailed Description

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

Author

Anthony Rousseau

Version

2.0.0

Date

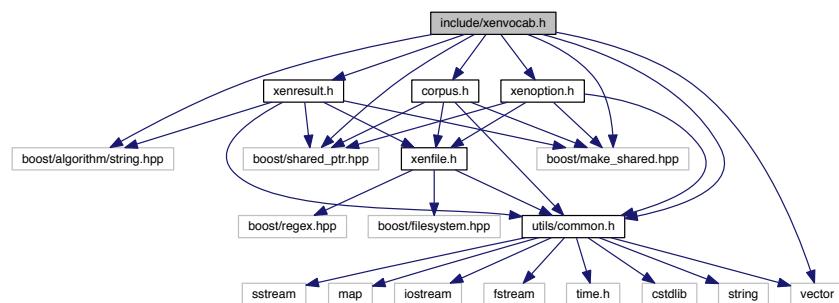
18 March 2016

7.32 include/xenvocab.h File Reference

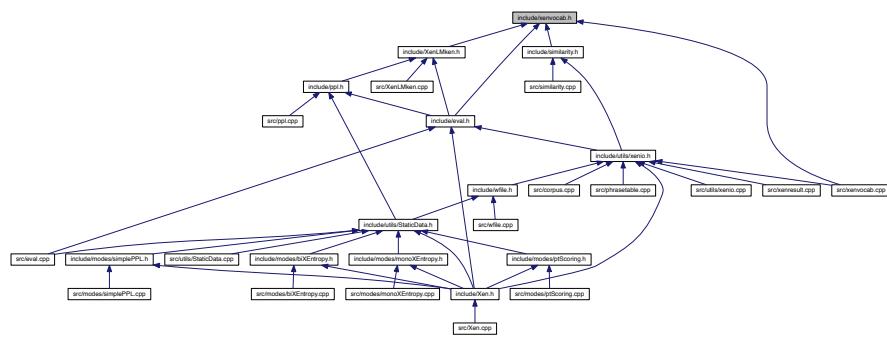
Class handling a XenC vocabulary.

```
#include <boost/algorithm/string.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/make_shared.hpp>
#include <vector>
#include "utils/common.h"
#include "corpus.h"
#include "xenresult.h"
#include "xenoption.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.

7.32.1 Detailed Description

Class handling a XenC vocabulary.

Author

Anthony Rousseau

Version

2.0.0

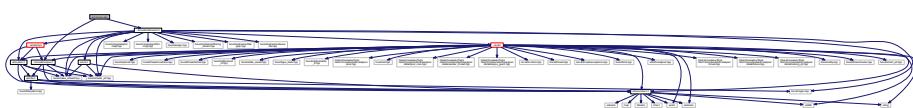
Date

18 March 2016

7.33 src/corpus.cpp File Reference

Class handling corpus-related functionalities.

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



7.33.1 Detailed Description

Class handling corpus-related functionalities.

Author

Anthony Rousseau

Version

2.0.0

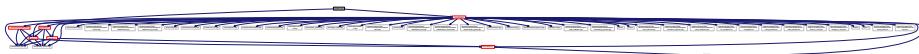
Date

18 March 2016

7.34 src/eval.cpp File Reference

Class handling evaluation system.

```
#include "../include/eval.h"
#include "../include/utils/StaticData.h"
Include dependency graph for eval.cpp:
```



Functions

- void `taskEval` (int pc, boost::shared_ptr<Corpus> ptrCorp, boost::shared_ptr<XenVocab> ptrVoc, boost::shared_ptr<Corpus> ptrDevCorp, boost::shared_ptr<EvalMap> ptrDist)
Thread-safe evaluation function.

7.34.1 Detailed Description

Class handling evaluation system.

Author

Anthony Rousseau

Version

2.0.0

Date

18 March 2016

7.34.2 Function Documentation

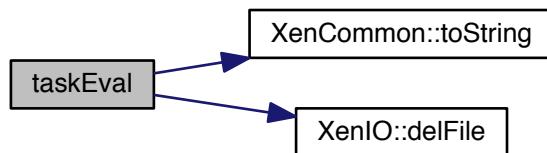
7.34.2.1 void `taskEval` (int pc, boost::shared_ptr<Corpus> ptrCorp, boost::shared_ptr<XenVocab> ptrVoc, boost::shared_ptr<Corpus> ptrDevCorp, boost::shared_ptr<EvalMap> ptrDist)

Thread-safe evaluation function.

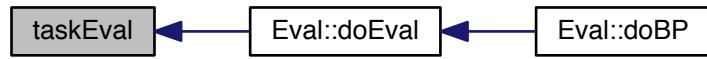
Parameters

<i>pc</i>	: integer representing the percentage of the scored out-of-domain corpus to take
<i>ptrCorp</i>	: shared pointer on the Corpus object representing the part of 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:



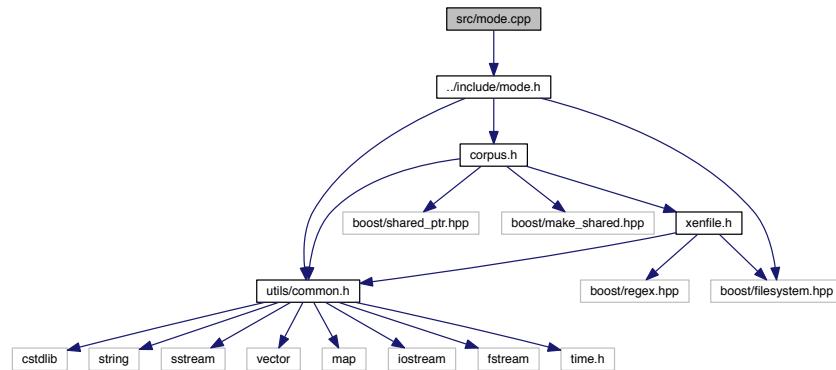
Here is the caller graph for this function:



7.35 src/mode.cpp File Reference

Abstract class defining the filtering modes architecture.

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



7.35.1 Detailed Description

Abstract class defining the filtering modes architecture.

Author

Anthony Rousseau

Version

2.0.0

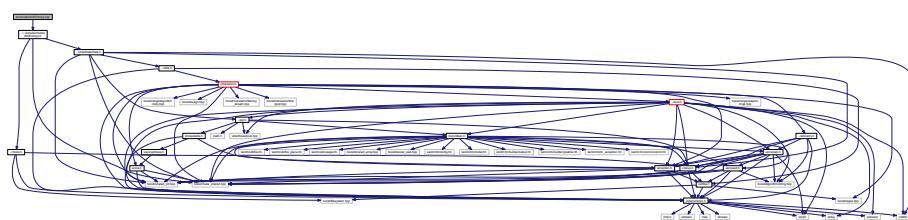
Date

18 March 2016

7.36 src/modes/biXEntropy.cpp File Reference

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

```
#include "../../include/modes/biXEntropy.h"
Include dependency graph for biXEntropy.cpp:
```



7.36.1 Detailed Description

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

Author

Anthony Rousseau

Version

2.0.0

Date

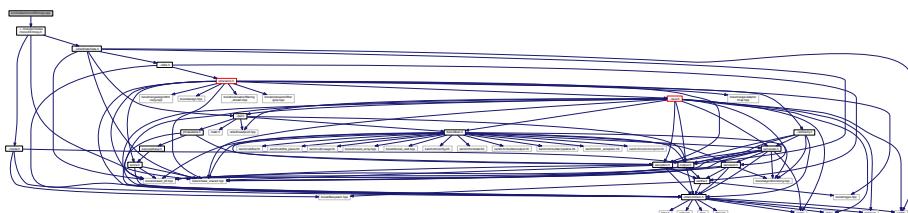
18 March 2016

7.37 src/modes/monoXEntropy.cpp File Reference

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

```
#include " ../../include/modes/monoXEntropy.h"
```

Include dependency graph for monoXEntropy.cpp:



7.37.1 Detailed Description

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

Author

Anthony Rousseau

Version

2.0.0

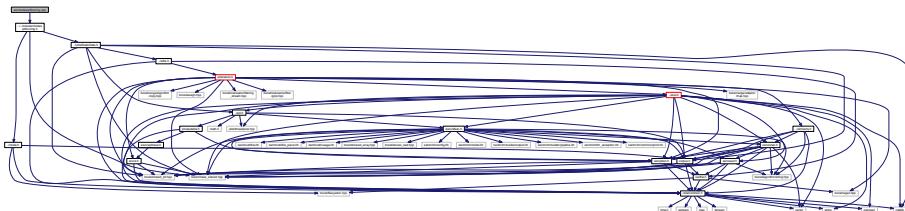
Date

18 March 2016

7.38 src/modes/ptScoring.cpp File Reference

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

```
#include "../../include/modes/ptScoring.h"
Include dependency graph for ptScoring.cpp:
```



7.38.1 Detailed Description

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

Author

Anthony Rousseau

Version

2.0.0

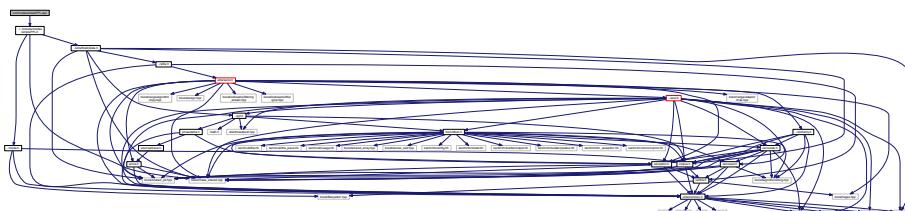
Date

18 March 2016

7.39 src/modes/simplePPL.cpp File Reference

Derived class to handle filtering mode 1: simple perplexity.

```
#include "../../include/modes/simplePPL.h"
Include dependency graph for simplePPL.cpp:
```



7.39.1 Detailed Description

Derived class to handle filtering mode 1: simple perplexity.

Author

Anthony Rousseau

Version

2.0.0

Date

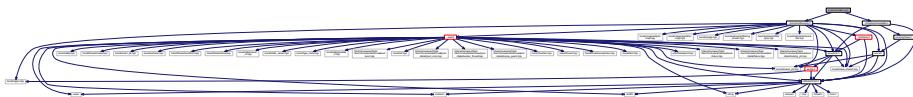
18 March 2016

7.40 src/phrasetable.cpp File Reference

Class handling phrase-table related functionalities.

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

Include dependency graph for phrasetable.cpp:



7.40.1 Detailed Description

Class handling phrase-table related functionalities.

Author

Anthony Rousseau

Version

2.0.0

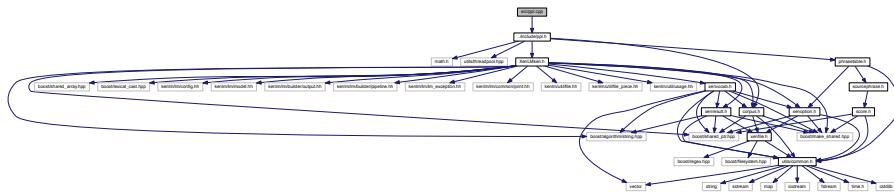
Date

18 March 2016

7.41 src/ppl.cpp File Reference

Class handling the perplexity/cross-entropy computations.

```
#include "../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<`XenLMken`> ptrLM)

Thread-safe perplexity computation function.

Variables

- boost::mutex [randy](#)

7.41.1 Detailed Description

Class handling the perplexity/cross-entropy computations.

Author

Anthony Rousseau

Version

200

Date

18 March 2016

7.41.2 Function Documentation

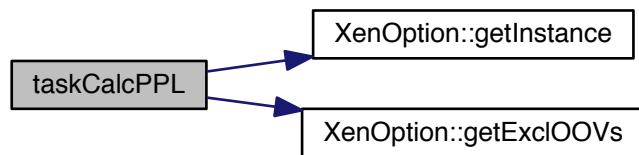
7.41.2.1 void taskCalcPPL (int numLine, std::string line, boost::shared_ptr< std::vector< double > > ptrPPL, boost::shared_ptr< XenLMken > ptrLM)

Thread-safe perplexity computation function.

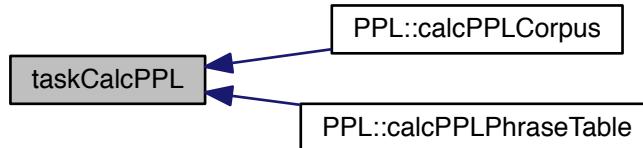
Parameters

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

Here is the call graph for this function:



Here is the caller graph for this function:



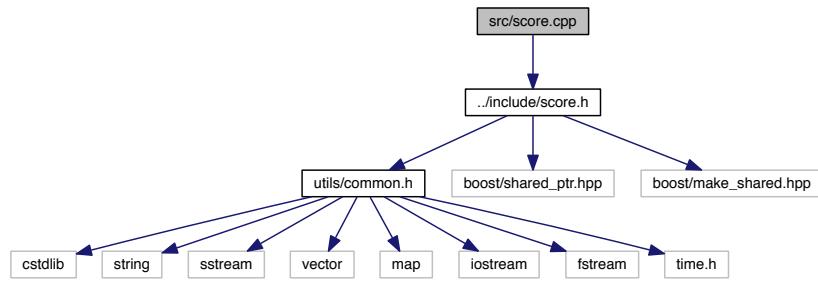
7.41.3 Variable Documentation

7.41.3.1 boost::mutex randy

7.42 src/score.cpp File Reference

Class holding the XenC scores representation.

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



7.42.1 Detailed Description

Class holding the XenC scores representation.

Author

Anthony Rousseau

Version

2.0.0

Date

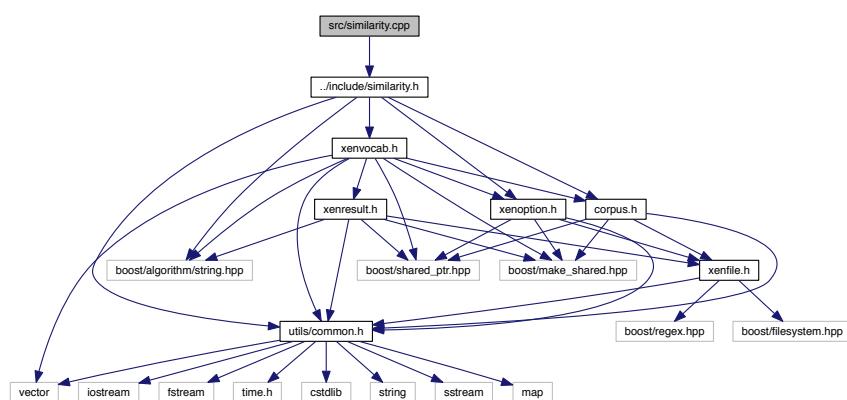
18 March 2016

7.43 src/similarity.cpp File Reference

Class taking care of all the similarity measure computations.

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

Include dependency graph for similarity.cpp:



7.43.1 Detailed Description

Class taking care of all the similarity measure computations.

Author

Anthony Rousseau

Version

2.0.0

Date

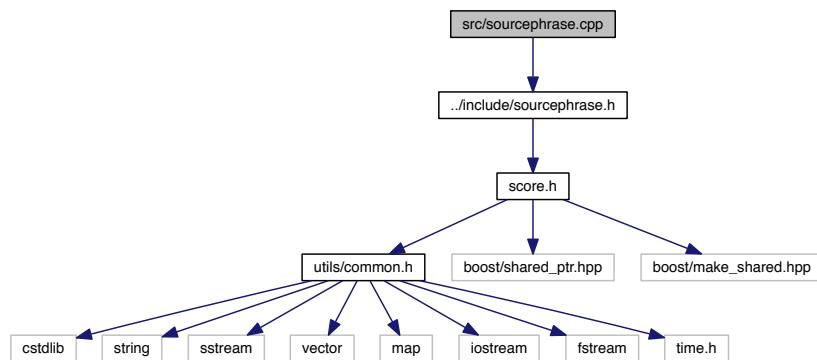
18 March 2016

7.44 src/sourcephrase.cpp File Reference

Class holding a merged source phrase and all associated data.

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

Include dependency graph for sourcephrase.cpp:



7.44.1 Detailed Description

Class holding a merged source phrase and all associated data.

Author

Anthony Rousseau

Version

2.0.0

Date

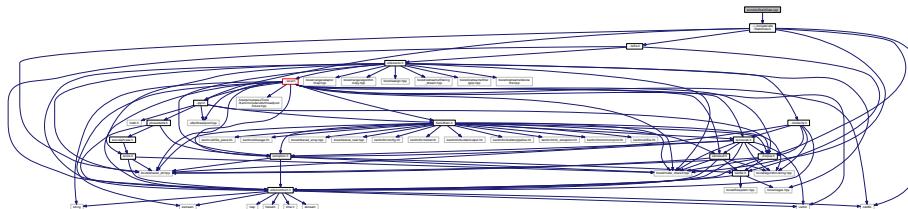
18 March 2016

7.45 src/utils/StaticData.cpp File Reference

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

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

Include dependency graph for StaticData.cpp:



7.45.1 Detailed Description

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

Author

Anthony Rousseau

Version

2.0.0

Date

18 March 2016

7.46 src/utils/xenio.cpp File Reference

Class handling all input/output operations of XenC.

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

Include dependency graph for xenio.cpp:



7.46.1 Detailed Description

Class handling all input/output operations of XenC.

Author

Anthony Rousseau

Version

2.0.0

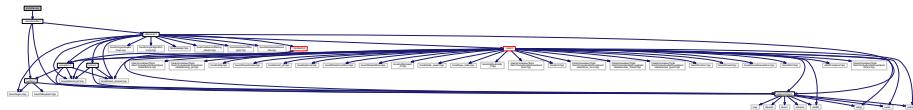
Date

18 March 2016

7.47 src/wfile.cpp File Reference

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

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



7.47.1 Detailed Description

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

Author

Anthony Rousseau

Version

2.0.0

Date

18 March 2016

7.48 src/Xen.cpp File Reference

Main file of XenC, controls execution.

```
#include "../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 std::string `version` = "2.0.0"

7.48.1 Detailed Description

Main file of XenC, controls execution.

Author

Anthony Rousseau

Version

2.0.0

Date

18 March 2016

7.48.2 Function Documentation

7.48.2.1 std::string getOutName (XenOption * opt)

Computes the output file name.

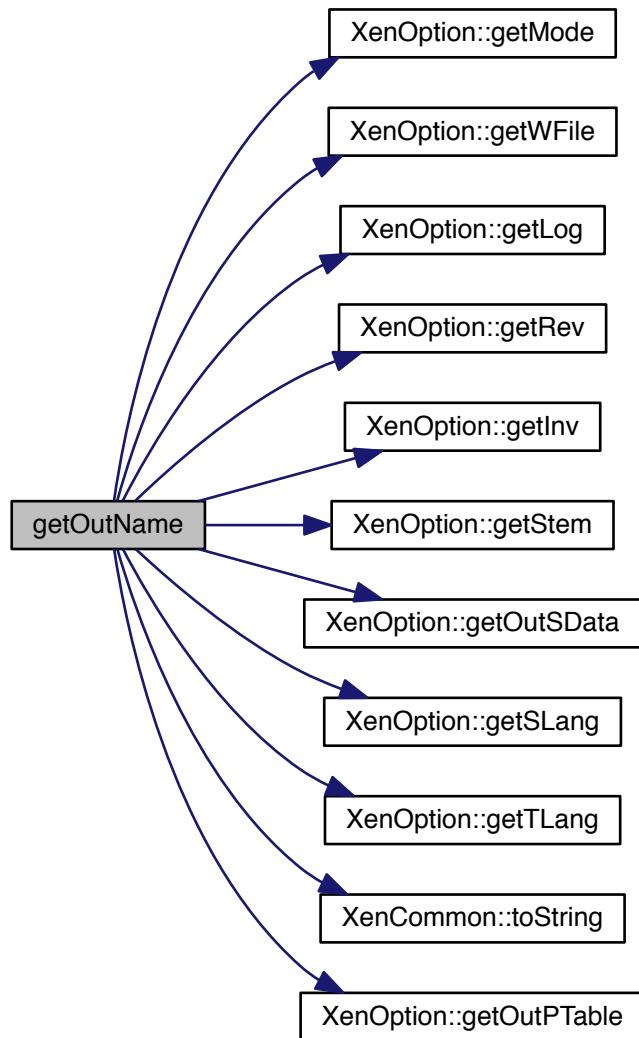
Parameters

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

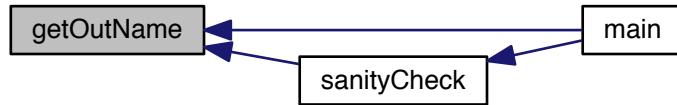
Returns

the output file name

Here is the call graph for this function:



Here is the caller graph for this function:



7.48.2.2 int main (int *argc*, char * *argv*[])

Main function of XenC.

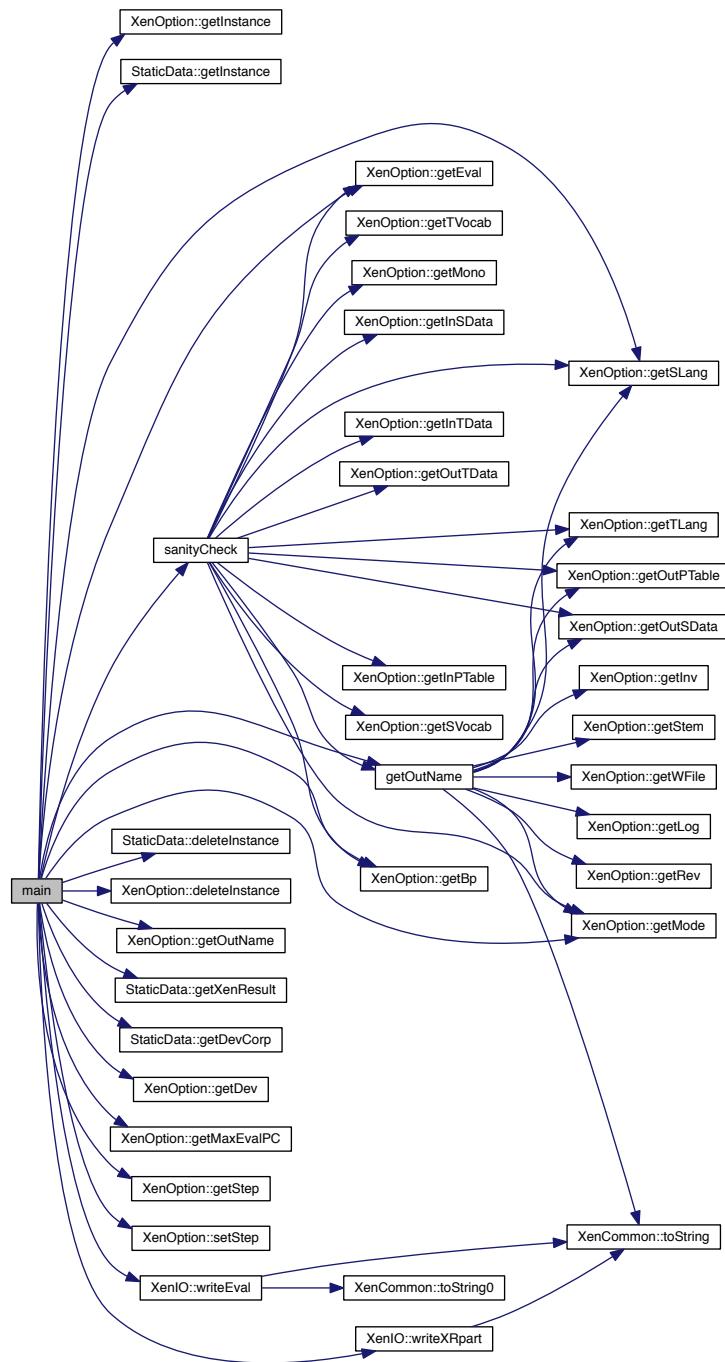
Parameters

<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:



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

Controls the mandatory options.

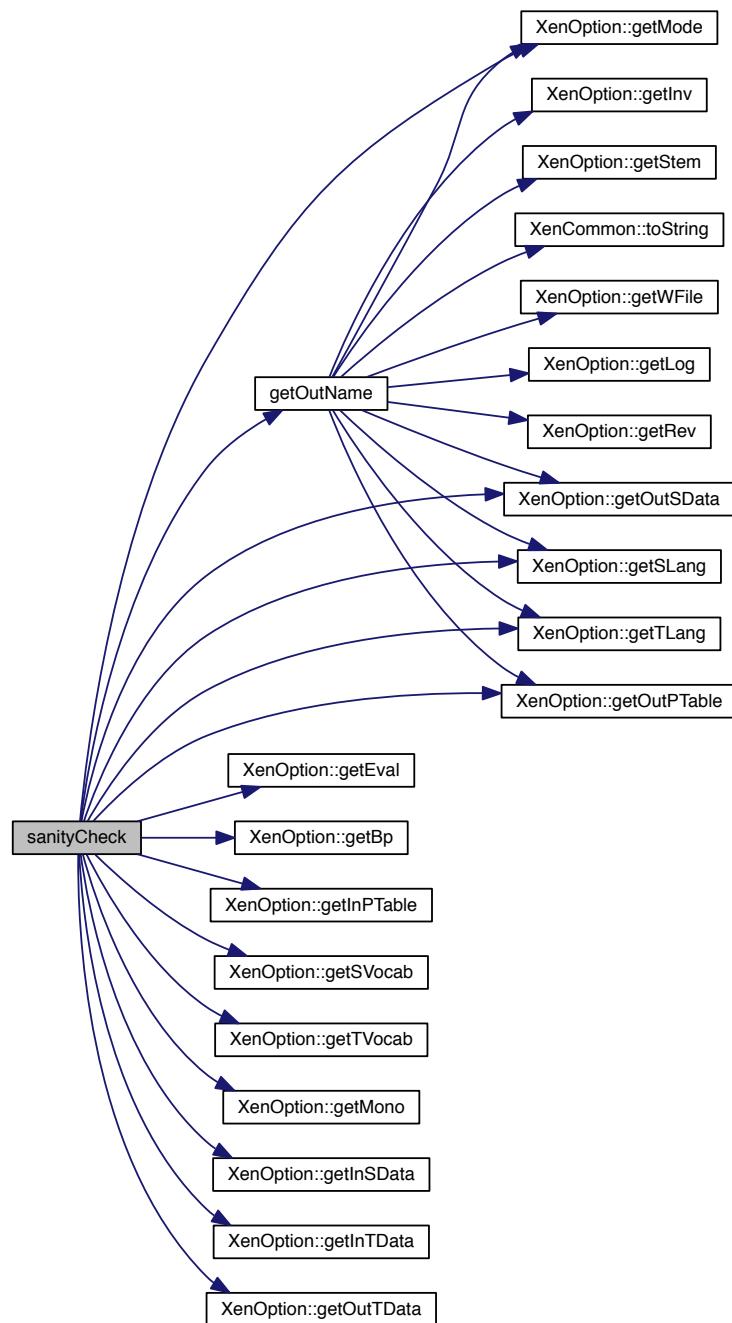
Parameters

<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:



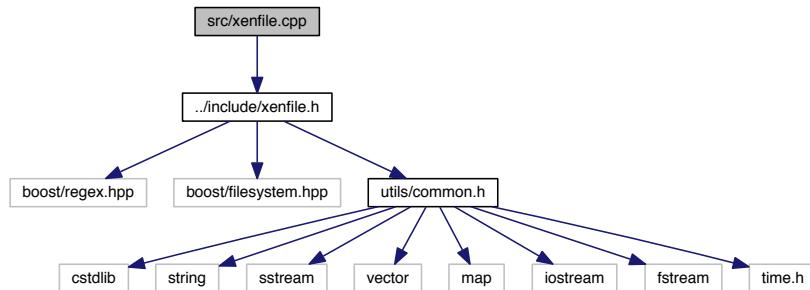
7.48.3 Variable Documentation

7.48.3.1 const std::string version = "2.0.0"

7.49 src/xenfile.cpp File Reference

Class providing some basic functions around files.

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



7.49.1 Detailed Description

Class providing some basic functions around files.

Author

Anthony Rousseau

Version

2.0.0

Date

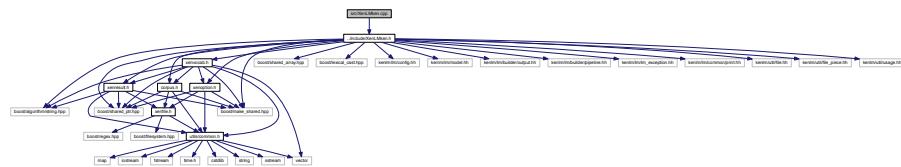
18 March 2016

7.50 src/XenLMken.cpp File Reference

Class handling KenLM estimation, loading, querying...

```
#include "../include/XenLMken.h"
```

Include dependency graph for XenLMken.cpp:



7.50.1 Detailed Description

Class handling KenLM estimation, loading, querying...

Author

Anthony Rousseau

Version

2.0.0

Date

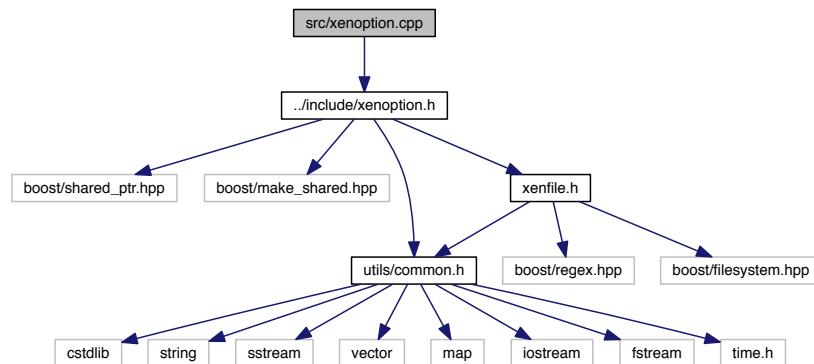
18 March 2016

7.51 src/xenoption.cpp File Reference

Singleton class handling XenC options accessors/mutators.

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

Include dependency graph for xenoption.cpp:



7.51.1 Detailed Description

Singleton class handling XenC options accessors/mutators.

Author

Anthony Rousseau

Version

2.0.0

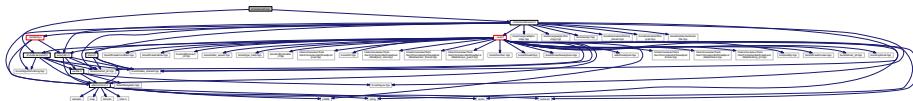
Date

18 March 2016

7.52 src/xenresult.cpp File Reference

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

```
#include "../include/xenresult.h"
#include "../include/utils/xenio.h"
Include dependency graph for xenresult.cpp:
```



7.52.1 Detailed Description

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

Author

Anthony Rousseau

Version

2.0.0

Date

18 March 2016

7.53 src/xenvocab.cpp File Reference

Class handling a XenC vocabulary.

```
#include "../include/xenvocab.h"
#include "../include/utils/xenio.h"
Include dependency graph for xenvocab.cpp:
```



7.53.1 Detailed Description

Class handling a XenC vocabulary.

Author

Anthony Rousseau

Version

2.0.0

Date

18 March 2016