

Y!LDA

1.0

Generated by Doxygen 1.6.3

Wed May 25 15:17:00 2011

Contents

1	Y!LDA Topic Modelling Framework	1
2	Y!LDA Architecture	7
2.1	Introduction	8
2.2	Goals	8
2.2.1	Adding new Models	8
2.2.2	Common Infrastructure	8
2.2.3	Scalability	8
2.3	Main components of the System	8
2.3.1	Builder Pattern	9
2.4	Distributed Set Up	9
2.4.1	Distributed_Map	10
2.4.2	Synchronizer	10
2.4.3	Synchronizer_Helper	11
2.5	Default Implementations provided	11
2.6	Unigram Model	12
2.7	Adding a new Model	12
2.8	Checkpoints	12
3	Multi-Machine Setup	15
4	Single Machine Setup	21
4.1	Learning a Model	22
4.1.1	Tokenization and Formatting	22

4.1.2	Learning the topic mixtures	24
4.1.3	Viewing the word mixtures for each topic	27
4.1.4	Viewing the topic assignments	29
4.2	Using the Model	30
4.2.1	Batch Mode	30
4.2.2	Streaming Mode	34
4.3	Output Generated	37
4.4	Customization	37
5	Using Y!LDA	39
6	Todo List	43
7	Namespace Index	45
7.1	Namespace List	45
8	Class Index	47
8.1	Class Hierarchy	47
9	Class Index	49
9.1	Class List	49
10	File Index	51
10.1	File List	51
11	Namespace Documentation	55
11.1	LDAUtil Namespace Reference	55
11.2	sampler Namespace Reference	56
11.2.1	Function Documentation	56
11.2.1.1	sample	56
12	Class Documentation	57
12.1	Checkpointter Class Reference	57
12.1.1	Detailed Description	58
12.1.2	Member Function Documentation	58

12.1.2.1	checkpoint	58
12.1.2.2	load_metadata	58
12.1.2.3	save_metadata	58
12.2	Client Class Reference	59
12.2.1	Detailed Description	59
12.2.2	Member Function Documentation	59
12.2.2.1	begin_putNget	59
12.2.2.2	get	59
12.2.2.3	put	60
12.2.2.4	remove	60
12.2.2.5	set	60
12.2.2.6	wait_for_all	60
12.2.2.7	wait_till_done	60
12.3	Context Class Reference	61
12.3.1	Detailed Description	61
12.3.2	Member Function Documentation	61
12.3.2.1	get_bool	61
12.3.2.2	get_double	62
12.3.2.3	get_instance	62
12.3.2.4	get_int	62
12.3.2.5	get_string	62
12.3.2.6	put_string	62
12.4	Cookie Struct Reference	63
12.4.1	Member Data Documentation	63
12.4.1.1	entity	63
12.4.1.2	msg_id	63
12.5	Data_Formatter Class Reference	64
12.5.1	Detailed Description	64
12.5.2	Member Function Documentation	64
12.5.2.1	format	64
12.5.2.2	get_dictionary	65

12.5.2.3	get_num_docs	65
12.5.2.4	get_total_num_words	65
12.6	DM_Client Class Reference	66
12.6.1	Detailed Description	66
12.6.2	Constructor & Destructor Documentation	67
12.6.2.1	DM_Client	67
12.6.2.2	~DM_Client	67
12.6.3	Member Function Documentation	67
12.6.3.1	begin_putNget	67
12.6.3.2	get	67
12.6.3.3	get_num_servers	67
12.6.3.4	put	67
12.6.3.5	remove	67
12.6.3.6	set	68
12.6.3.7	wait_for_all	68
12.6.3.8	wait_till_done	68
12.7	DM_Server Class Reference	69
12.7.1	Detailed Description	69
12.7.2	Constructor & Destructor Documentation	70
12.7.2.1	DM_Server	70
12.7.2.2	~DM_Server	70
12.7.3	Member Function Documentation	70
12.7.3.1	get	70
12.7.3.2	put	70
12.7.3.3	putNget_async	70
12.7.3.4	remove	71
12.7.3.5	set	71
12.7.3.6	waitForAllClients_async	71
12.8	LDAUtil::DM_Server_Names Class Reference	72
12.8.1	Member Function Documentation	72
12.8.1.1	get_servant_name	72

12.8.1.2	get_server_endpoint	72
12.9	DocumentReader Class Reference	73
12.9.1	Detailed Description	73
12.9.2	Constructor & Destructor Documentation	73
12.9.2.1	DocumentReader	73
12.9.2.2	~DocumentReader	73
12.9.3	Member Function Documentation	73
12.9.3.1	read	73
12.10	DocumentWriter Class Reference	74
12.10.1	Detailed Description	74
12.10.2	Constructor & Destructor Documentation	74
12.10.2.1	DocumentWriter	74
12.10.2.2	~DocumentWriter	74
12.10.3	Member Function Documentation	74
12.10.3.1	write	74
12.11	Execution_Strategy Class Reference	75
12.11.1	Detailed Description	75
12.11.2	Member Function Documentation	75
12.11.2.1	execute	75
12.12	Filter_Eval Class Reference	76
12.12.1	Detailed Description	76
12.12.2	Constructor & Destructor Documentation	76
12.12.2.1	Filter_Eval	76
12.12.2.2	~Filter_Eval	76
12.12.3	Member Function Documentation	76
12.12.3.1	get_eval	76
12.12.3.2	operator()	76
12.13	Filter_Optimizer Class Reference	77
12.13.1	Detailed Description	77
12.13.2	Constructor & Destructor Documentation	77
12.13.2.1	Filter_Optimizer	77

12.13.2.2 ~Filter_Optimizer	77
12.13.3 Member Function Documentation	77
12.13.3.1 operator()	77
12.14 Filter_Reader Class Reference	78
12.14.1 Detailed Description	78
12.14.2 Constructor & Destructor Documentation	78
12.14.2.1 Filter_Reader	78
12.14.2.2 ~Filter_Reader	78
12.14.3 Member Function Documentation	78
12.14.3.1 operator()	78
12.15 Filter_Sampler Class Reference	79
12.15.1 Detailed Description	79
12.15.2 Constructor & Destructor Documentation	79
12.15.2.1 Filter_Sampler	79
12.15.2.2 ~Filter_Sampler	79
12.15.3 Member Function Documentation	79
12.15.3.1 operator()	79
12.16 Filter_Tester Class Reference	80
12.16.1 Detailed Description	80
12.16.2 Constructor & Destructor Documentation	80
12.16.2.1 Filter_Tester	80
12.16.2.2 ~Filter_Tester	80
12.16.3 Member Function Documentation	80
12.16.3.1 operator()	80
12.17 Filter_Updater Class Reference	81
12.17.1 Constructor & Destructor Documentation	81
12.17.1.1 Filter_Updater	81
12.17.1.2 ~Filter_Updater	81
12.17.2 Member Function Documentation	81
12.17.2.1 operator()	81
12.18 Filter_Writer Class Reference	82

12.18.1 Detailed Description	82
12.18.2 Constructor & Destructor Documentation	82
12.18.2.1 Filter_Writer	82
12.18.2.2 ~Filter_Writer	82
12.18.3 Member Function Documentation	82
12.18.3.1 operator()	82
12.19 GenericTopKList< T, GreaterThan > Class Template Reference	83
12.19.1 Detailed Description	83
12.19.2 Constructor & Destructor Documentation	84
12.19.2.1 GenericTopKList	84
12.19.2.2 ~GenericTopKList	84
12.19.3 Member Function Documentation	84
12.19.3.1 clear	84
12.19.3.2 empty	84
12.19.3.3 pop	84
12.19.3.4 print	84
12.19.3.5 push	84
12.19.3.6 top	84
12.20 Hadoop_Checkpointer Class Reference	85
12.20.1 Constructor & Destructor Documentation	85
12.20.1.1 Hadoop_Checkpointer	85
12.20.1.2 ~Hadoop_Checkpointer	85
12.20.2 Member Function Documentation	85
12.20.2.1 checkpoint	85
12.21 Hashmap_Array< Key, Value > Class Template Reference	86
12.21.1 Detailed Description	86
12.21.2 Member Typedef Documentation	87
12.21.2.1 act_map	87
12.21.2.2 act_map_iter	87
12.21.3 Constructor & Destructor Documentation	87
12.21.3.1 Hashmap_Array	87

12.21.3.2 ~HashMap_Array	87
12.21.4 Member Function Documentation	87
12.21.4.1 begin	87
12.21.4.2 count	87
12.21.4.3 end	87
12.21.4.4 erase	87
12.21.4.5 get_lock	87
12.21.4.6 get_structure_lock	87
12.21.4.7 operator[]	87
12.21.4.8 size	87
12.22InvalidOldTopicExc Class Reference	88
12.22.1 Detailed Description	88
12.22.2 Constructor & Destructor Documentation	88
12.22.2.1 InvalidOldTopicExc	88
12.22.3 Member Function Documentation	88
12.22.3.1 what	88
12.23HashMap_Array< Key, Value >::iterator Class Reference	89
12.23.1 Detailed Description	89
12.23.2 Constructor & Destructor Documentation	89
12.23.2.1 iterator	89
12.23.3 Member Function Documentation	89
12.23.3.1 operator!=	89
12.23.3.2 operator++	89
12.23.4 Member Data Documentation	89
12.23.4.1 current_iter	89
12.24LDAUtil::Itoa Class Reference	91
12.24.1 Member Function Documentation	91
12.24.1.1 get_string	91
12.25Local_Checkpointer Class Reference	92
12.25.1 Constructor & Destructor Documentation	92
12.25.1.1 Local_Checkpointer	92

12.25.1.2 ~Local_Checkpointer	92
12.25.2 Member Function Documentation	92
12.25.2.1 checkpoint	92
12.25.2.2 load_metadata	93
12.25.2.3 save_metadata	93
12.26Model Class Reference	94
12.26.1 Detailed Description	94
12.26.2 Member Function Documentation	94
12.26.2.1 get_eval	94
12.26.2.2 save	95
12.26.2.3 write_statistics	95
12.26.3 Member Data Documentation	95
12.26.3.1 UNIGRAM	95
12.27Model_Builder Class Reference	96
12.27.1 Detailed Description	96
12.27.2 Member Function Documentation	96
12.27.2.1 create_execution_strategy	96
12.27.2.2 create_model_refiner	97
12.27.2.3 create_output	97
12.27.2.4 create_pipeline	97
12.27.2.5 get_model	97
12.28Model_Director Class Reference	98
12.28.1 Detailed Description	98
12.28.2 Constructor & Destructor Documentation	98
12.28.2.1 Model_Director	98
12.28.2.2 ~Model_Director	98
12.28.3 Member Function Documentation	98
12.28.3.1 build_model	98
12.29Model_Refiner Class Reference	99
12.29.1 Detailed Description	99
12.29.2 Member Function Documentation	100

12.29.2.1 allocate_document_buffer	100
12.29.2.2 deallocate_document_buffer	100
12.29.2.3 eval	100
12.29.2.4 get_nth_document	100
12.29.2.5 iteration_done	100
12.29.2.6 optimize	100
12.29.2.7 read	100
12.29.2.8 sample	101
12.29.2.9 test	101
12.29.2.10 update	101
12.29.2.11 write	101
12.30 Parameter Struct Reference	102
12.30.1 Detailed Description	102
12.30.2 Constructor & Destructor Documentation	103
12.30.2.1 Parameter	103
12.30.2.2 Parameter	103
12.30.2.3 ~Parameter	103
12.30.3 Member Function Documentation	103
12.30.3.1 dump	103
12.30.3.2 initialize_from_dump	103
12.30.3.3 initialize_from_values	103
12.30.3.4 operator=	103
12.30.4 Member Data Documentation	103
12.30.4.1 length	103
12.30.4.2 sum	103
12.30.4.3 values	103
12.31 Pipeline Class Reference	104
12.31.1 Detailed Description	104
12.31.2 Member Function Documentation	104
12.31.2.1 add_eval	104
12.31.2.2 add_optimizer	105

12.31.2.3 add_reader	105
12.31.2.4 add_sampler	105
12.31.2.5 add_tester	105
12.31.2.6 add_updater	105
12.31.2.7 add_writer	105
12.31.2.8 clear	105
12.31.2.9 destroy	105
12.31.2.10 get_eval	105
12.31.2.11 get_refiner	105
12.31.2.12 init	106
12.31.2.13 run	106
12.32 PNGCallback Class Reference	107
12.32.1 Detailed Description	107
12.32.2 Constructor & Destructor Documentation	108
12.32.2.1 PNGCallback	108
12.32.2.2 ~PNGCallback	108
12.32.3 Member Function Documentation	108
12.32.3.1 finished	108
12.32.3.2 num_synchs	108
12.32.3.3 set_done	108
12.32.3.4 wait_till_done	108
12.33 PNGJob Class Reference	109
12.33.1 Detailed Description	109
12.33.2 Constructor & Destructor Documentation	109
12.33.2.1 PNGJob	109
12.33.3 Member Data Documentation	109
12.33.3.1 delta	109
12.33.3.2 png_cb	109
12.33.3.3 word	110
12.34 Server_Helper Class Reference	111
12.34.1 Detailed Description	111

12.34.2 Member Function Documentation	111
12.34.2.1 combine	111
12.35simple_map Class Reference	112
12.35.1 Constructor & Destructor Documentation	112
12.35.1.1 simple_map	112
12.35.1.2 ~simple_map	112
12.35.2 Member Function Documentation	112
12.35.2.1 clear	112
12.35.2.2 get	112
12.35.2.3 hash	112
12.35.2.4 print	112
12.35.2.5 put	112
12.36LDAUtil::StringTokenizer Class Reference	113
12.36.1 Detailed Description	113
12.36.2 Member Function Documentation	113
12.36.2.1 tokenize	113
12.37LDAUtil::StringTrimmer Class Reference	114
12.37.1 Member Function Documentation	114
12.37.1.1 trim	114
12.38Synchronized_Training_Execution_Strategy Class Reference	115
12.38.1 Detailed Description	115
12.38.2 Constructor & Destructor Documentation	116
12.38.2.1 Synchronized_Training_Execution_Strategy	116
12.38.2.2 ~Synchronized_Training_Execution_Strategy	116
12.38.3 Member Function Documentation	116
12.38.3.1 execute	116
12.39Synchronizer Class Reference	117
12.39.1 Detailed Description	117
12.39.2 Constructor & Destructor Documentation	117
12.39.2.1 Synchronizer	117
12.39.2.2 ~Synchronizer	117

12.39.3 Member Function Documentation	117
12.39.3.1 is_all_iters_done	117
12.39.3.2 set_all_iters_done	118
12.39.3.3 synchronize	118
12.40 Synchronizer_Helper Class Reference	119
12.40.1 Detailed Description	119
12.40.2 Member Function Documentation	119
12.40.2.1 end_putNget	119
12.40.2.2 has_to_synchronize	119
12.40.2.3 initialize	120
12.40.2.4 reset_to_synchronize	120
12.40.2.5 synchronize	120
12.41 TBB_Pipeline Class Reference	121
12.41.1 Detailed Description	122
12.41.2 Constructor & Destructor Documentation	122
12.41.2.1 TBB_Pipeline	122
12.41.2.2 ~TBB_Pipeline	122
12.41.3 Member Function Documentation	122
12.41.3.1 add_eval	122
12.41.3.2 add_optimizer	122
12.41.3.3 add_reader	122
12.41.3.4 add_sampler	122
12.41.3.5 add_tester	122
12.41.3.6 add_updater	122
12.41.3.7 add_writer	123
12.41.3.8 clear	123
12.41.3.9 destroy	123
12.41.3.10 get_eval	123
12.41.3.11 get_refiner	123
12.41.3.12 init	123
12.41.3.13 run	123

12.41.4 Member Data Documentation	124
12.41.4.1 _eval	124
12.41.4.2 _init	124
12.41.4.3 _optimizer	124
12.41.4.4 _pipeline	124
12.41.4.5 _reader	124
12.41.4.6 _refiner	124
12.41.4.7 _sampler	124
12.41.4.8 _tester	124
12.41.4.9 _updater	124
12.41.4.10_writer	124
12.42 Testing_Execution_Strategy Class Reference	125
12.42.1 Detailed Description	125
12.42.2 Constructor & Destructor Documentation	125
12.42.2.1 Testing_Execution_Strategy	125
12.42.2.2 ~Testing_Execution_Strategy	125
12.42.3 Member Function Documentation	125
12.42.3.1 execute	125
12.43 TopicCounts Struct Reference	126
12.43.1 Constructor & Destructor Documentation	127
12.43.1.1 TopicCounts	127
12.43.1.2 TopicCounts	127
12.43.1.3 TopicCounts	127
12.43.1.4 TopicCounts	127
12.43.1.5 ~TopicCounts	127
12.43.1.6 TopicCounts	127
12.43.2 Member Function Documentation	127
12.43.2.1 addNewTop	127
12.43.2.2 addNewTopAftChk	128
12.43.2.3 assign	128
12.43.2.4 compact	128

12.43.2.5	convertTo	128
12.43.2.6	convertTo	128
12.43.2.7	convertTo	128
12.43.2.8	convertTo_d	128
12.43.2.9	decrement	128
12.43.2.10	equal	129
12.43.2.11	findAndDecrement	129
12.43.2.12	findAndIncrement	129
12.43.2.13	findOldnNew	129
12.43.2.14	get_counts	129
12.43.2.15	get_frequency	129
12.43.2.16	increment	129
12.43.2.17	init	129
12.43.2.18	init	129
12.43.2.19	operator+=	129
12.43.2.20	operator-=	130
12.43.2.21	print	130
12.43.2.22	removeOldTop	130
12.43.2.23	replace	130
12.43.2.24	setLength	130
12.43.2.25	upd_count	130
12.43.3	Member Data Documentation	131
12.43.3.1	frequency	131
12.43.3.2	items	131
12.43.3.3	length	131
12.43.3.4	origLength	131
12.43.3.5	QUIT	131
12.43.3.6	vec_items	131
12.44	TopKList Class Reference	132
12.44.1	Member Typedef Documentation	132
12.44.1.1	iterator	132

12.44.2 Constructor & Destructor Documentation	132
12.44.2.1 TopKList	132
12.44.2.2 ~TopKList	132
12.44.3 Member Function Documentation	132
12.44.3.1 clear	132
12.44.3.2 get_beg	132
12.44.3.3 get_end	133
12.44.3.4 get_max	133
12.44.3.5 insert_word	133
12.44.3.6 is_sorted	133
12.44.3.7 print	133
12.45 Training_Execution_Strategy Class Reference	134
12.45.1 Detailed Description	134
12.45.2 Constructor & Destructor Documentation	134
12.45.2.1 Training_Execution_Strategy	134
12.45.2.2 ~Training_Execution_Strategy	134
12.45.3 Member Function Documentation	134
12.45.3.1 execute	134
12.46 TypeTopicCounts Class Reference	136
12.46.1 Constructor & Destructor Documentation	137
12.46.1.1 TypeTopicCounts	137
12.46.1.2 TypeTopicCounts	137
12.46.1.3 ~TypeTopicCounts	137
12.46.2 Member Function Documentation	137
12.46.2.1 clear_stats	137
12.46.2.2 destroy	138
12.46.2.3 dump	138
12.46.2.4 equal	138
12.46.2.5 estimate_alphas	138
12.46.2.6 estimate_fit	138
12.46.2.7 estimate_fit	138

12.46.2.8 estimate_memoryn_warn	138
12.46.2.9 get_counts	138
12.46.2.10 get_counts	138
12.46.2.11 get_lock	138
12.46.2.12 get_num_topics	139
12.46.2.13 get_num_words	139
12.46.2.14 get_topic_stats	139
12.46.2.15 init	139
12.46.2.16 initialize	139
12.46.2.17 initialize	139
12.46.2.18 initialize_from_docs	139
12.46.2.19 initialize_from_dump	139
12.46.2.20 initialize_from_string	140
12.46.2.21 initialize_from_ttc	140
12.46.2.22 print	140
12.46.2.23 print	140
12.46.2.24 replace	140
12.46.2.25 upd_count	140
12.46.2.26 upd_count	140
12.46.2.27 verify_header	141
12.46.3 Friends And Related Function Documentation	141
12.46.3.1 Memcached_Synchronizer	141
12.46.4 Member Data Documentation	141
12.46.4.1 num_topics	141
12.46.4.2 tokens_per_topic	141
12.46.4.3 top_topics	141
12.46.4.4 topic_stats	141
12.47 Unigram_Model Class Reference	142
12.47.1 Constructor & Destructor Documentation	143
12.47.1.1 Unigram_Model	143
12.47.1.2 ~Unigram_Model	143

12.47.2 Member Function Documentation	143
12.47.2.1 get_eval	143
12.47.2.2 get_parameter	143
12.47.2.3 get_ttc	143
12.47.2.4 save	143
12.47.2.5 set_parameter	143
12.47.2.6 write_statistics	143
12.47.3 Member Data Documentation	143
12.47.3.1 ALPHA	143
12.47.3.2 BETA	143
12.48 Unigram_Model_Server_Helper Class Reference	144
12.48.1 Constructor & Destructor Documentation	144
12.48.1.1 Unigram_Model_Server_Helper	144
12.48.1.2 ~Unigram_Model_Server_Helper	144
12.48.2 Member Function Documentation	144
12.48.2.1 combine	144
12.49 Unigram_Model_Streammer Class Reference	145
12.49.1 Constructor & Destructor Documentation	146
12.49.1.1 Unigram_Model_Streammer	146
12.49.1.2 ~Unigram_Model_Streammer	146
12.49.2 Member Function Documentation	146
12.49.2.1 insert_word_to_dict	146
12.49.2.2 iteration_done	146
12.49.2.3 read	146
12.49.2.4 write	146
12.50 Unigram_Model_Streaming_Builder Class Reference	147
12.50.1 Constructor & Destructor Documentation	148
12.50.1.1 Unigram_Model_Streaming_Builder	148
12.50.1.2 ~Unigram_Model_Streaming_Builder	148
12.50.2 Member Function Documentation	148
12.50.2.1 create_execution_strategy	148

12.50.2.2	create_model_refiner	148
12.50.2.3	create_output	148
12.50.2.4	create_pipeline	148
12.50.2.5	get_dict	148
12.50.2.6	get_model	148
12.50.2.7	init_dict	149
12.50.3	Member Data Documentation	149
12.50.3.1	_dict	149
12.50.3.2	_global_dict	149
12.50.3.3	_model	149
12.50.3.4	_pipeline	149
12.50.3.5	_refiner	149
12.50.3.6	_strategy	149
12.51	Unigram_Model_Synchronized_Training_Builder Class Reference	150
12.51.1	Constructor & Destructor Documentation	150
12.51.1.1	Unigram_Model_Synchronized_Training_Builder	150
12.51.1.2	~Unigram_Model_Synchronized_Training_Builder	150
12.51.2	Member Function Documentation	150
12.51.2.1	create_execution_strategy	150
12.51.3	Member Data Documentation	151
12.51.3.1	_sync_helper	151
12.52	Unigram_Model_Synchronizer_Helper Class Reference	152
12.52.1	Constructor & Destructor Documentation	152
12.52.1.1	Unigram_Model_Synchronizer_Helper	152
12.52.1.2	~Unigram_Model_Synchronizer_Helper	152
12.52.2	Member Function Documentation	152
12.52.2.1	end_putNget	152
12.52.2.2	has_to_synchronize	153
12.52.2.3	initialize	153
12.52.2.4	reset_to_synchronize	153
12.52.2.5	synchronize	153

12.53 Unigram_Model_Tester Class Reference	154
12.53.1 Constructor & Destructor Documentation	155
12.53.1.1 Unigram_Model_Tester	155
12.53.1.2 ~Unigram_Model_Tester	155
12.53.2 Member Function Documentation	155
12.53.2.1 allocate_document_buffer	155
12.53.2.2 deallocate_document_buffer	155
12.53.2.3 eval	155
12.53.2.4 get_nth_document	155
12.53.2.5 iteration_done	155
12.53.2.6 optimize	156
12.53.2.7 read	156
12.53.2.8 sample	156
12.53.2.9 test	156
12.53.2.10 update	156
12.53.2.11 write	156
12.53.3 Member Data Documentation	157
12.53.3.1 _alpha	157
12.53.3.2 _beta	157
12.53.3.3 _num_topics	157
12.53.3.4 _num_words	157
12.53.3.5 _tdoc_writer	157
12.53.3.6 _ttc	157
12.53.3.7 _wdoc_rdr	157
12.53.3.8 doc_index	157
12.53.3.9 ignore_old_topic	157
12.54 Unigram_Model_Testing_Builder Class Reference	158
12.54.1 Constructor & Destructor Documentation	158
12.54.1.1 Unigram_Model_Testing_Builder	158
12.54.1.2 ~Unigram_Model_Testing_Builder	158
12.54.2 Member Function Documentation	158

12.54.2.1	create_execution_strategy	158
12.54.2.2	create_model_refiner	158
12.55	Unigram_Model_Trainer Class Reference	160
12.55.1	Detailed Description	160
12.55.2	Constructor & Destructor Documentation	161
12.55.2.1	Unigram_Model_Trainer	161
12.55.2.2	~Unigram_Model_Trainer	161
12.55.3	Member Function Documentation	161
12.55.3.1	allocate_document_buffer	161
12.55.3.2	deallocate_document_buffer	161
12.55.3.3	eval	161
12.55.3.4	get_nth_document	161
12.55.3.5	iteration_done	161
12.55.3.6	optimize	162
12.55.3.7	read	162
12.55.3.8	sample	162
12.55.3.9	test	162
12.55.3.10	update	162
12.55.3.11	write	162
12.55.4	Member Data Documentation	163
12.55.4.1	doc_index	163
12.56	Unigram_Model_Training_Builder Class Reference	164
12.56.1	Constructor & Destructor Documentation	165
12.56.1.1	Unigram_Model_Training_Builder	165
12.56.1.2	~Unigram_Model_Training_Builder	165
12.56.2	Member Function Documentation	165
12.56.2.1	create_execution_strategy	165
12.56.2.2	create_model_refiner	165
12.56.2.3	create_output	165
12.56.2.4	create_pipeline	165
12.56.2.5	get_dict	165

12.56.2.6	<code>get_model</code>	165
12.56.2.7	<code>init_dict</code>	166
12.56.2.8	<code>initialize_topics</code>	166
12.56.3	Member Data Documentation	166
12.56.3.1	<code>_checkpointer</code>	166
12.56.3.2	<code>_dict</code>	166
12.56.3.3	<code>_model</code>	166
12.56.3.4	<code>_pipeline</code>	166
12.56.3.5	<code>_refiner</code>	166
12.56.3.6	<code>_strategy</code>	166
12.57	<code>Unigram_Test_Data_Formatter</code> Class Reference	167
12.57.1	Constructor & Destructor Documentation	167
12.57.1.1	<code>Unigram_Test_Data_Formatter</code>	167
12.57.1.2	<code>~Unigram_Test_Data_Formatter</code>	167
12.57.2	Member Function Documentation	167
12.57.2.1	<code>insert_word_to_dict</code>	167
12.58	<code>Unigram_Train_Data_Formatter</code> Class Reference	168
12.58.1	Constructor & Destructor Documentation	169
12.58.1.1	<code>Unigram_Train_Data_Formatter</code>	169
12.58.1.2	<code>~Unigram_Train_Data_Formatter</code>	169
12.58.2	Member Function Documentation	169
12.58.2.1	<code>format</code>	169
12.58.2.2	<code>get_dictionary</code>	169
12.58.2.3	<code>get_num_docs</code>	169
12.58.2.4	<code>get_total_num_words</code>	169
12.58.2.5	<code>insert_word_to_dict</code>	170
12.58.2.6	<code>read_from_inp</code>	170
12.58.3	Member Data Documentation	170
12.58.3.1	<code>_dict</code>	170
12.58.3.2	<code>_doc_writer</code>	170
12.58.3.3	<code>_in</code>	170

12.58.3.4	_num_docs	170
12.58.3.5	_num_words_in_all_docs	170
12.58.3.6	_stopWords	170
12.59	WordIndexDictionary Class Reference	171
12.59.1	Detailed Description	171
12.59.2	Constructor & Destructor Documentation	171
12.59.2.1	WordIndexDictionary	171
12.59.2.2	~WordIndexDictionary	172
12.59.3	Member Function Documentation	172
12.59.3.1	dump	172
12.59.3.2	get_freq	172
12.59.3.3	get_index	172
12.59.3.4	get_num_words	172
12.59.3.5	get_prev_index	172
12.59.3.6	get_word	172
12.59.3.7	initialize_from_dict	172
12.59.3.8	initialize_from_dump	172
12.59.3.9	initialize_from_dumps	172
12.59.3.10	insert_word	172
12.59.3.11	lmatch_word_index	173
12.59.3.12	print	173
12.59.3.13	size	173
12.59.4	Member Data Documentation	173
12.59.4.1	frequencies	173
13	File Documentation	175
13.1	src/architecture.h File Reference	175
13.2	src/commons/Client.h File Reference	176
13.3	src/commons/comparator.cpp File Reference	177
13.3.1	Function Documentation	177
13.3.1.1	cnt_cmp	177

13.3.1.2	<code>cnt_cmp_ttc</code>	177
13.3.1.3	<code>freq_cmp</code>	177
13.3.1.4	<code>prob_cmp</code>	177
13.4	<code>src/commons/comparator.h</code> File Reference	178
13.4.1	Function Documentation	178
13.4.1.1	<code>cnt_cmp</code>	178
13.4.1.2	<code>cnt_cmp_ttc</code>	178
13.4.1.3	<code>freq_cmp</code>	178
13.4.1.4	<code>prob_cmp</code>	178
13.5	<code>src/commons/constants.h</code> File Reference	179
13.6	<code>src/commons/Context.cpp</code> File Reference	180
13.7	<code>src/commons/Context.h</code> File Reference	181
13.8	<code>src/commons/document.pb.cc</code> File Reference	182
13.8.1	Define Documentation	182
13.8.1.1	<code>DO_</code>	182
13.8.1.2	<code>DO_</code>	182
13.8.1.3	<code>DO_</code>	182
13.8.1.4	<code>DO_</code>	182
13.8.1.5	<code>DO_</code>	182
13.8.1.6	<code>INTERNAL_SUPPRESS_PROTOBUF_FIELD_DEPRECATION</code>	182
13.9	<code>src/commons/document.pb.h</code> File Reference	183
13.10	<code>src/commons/DocumentReader.cpp</code> File Reference	184
13.11	<code>src/commons/DocumentReader.h</code> File Reference	185
13.12	<code>src/commons/DocumentWriter.cpp</code> File Reference	186
13.13	<code>src/commons/DocumentWriter.h</code> File Reference	187
13.14	<code>src/commons/Formatter/Controller.cpp</code> File Reference	188
13.14.1	Function Documentation	188
13.14.1.1	<code>get_data_formatter</code>	188
13.14.1.2	<code>main</code>	188
13.14.1.3	<code>release_data_formatter</code>	188

13.15src/commons/TopicLearner/Controller.cpp File Reference	189
13.15.1 Function Documentation	189
13.15.1.1 main	189
13.16src/commons/Formatter/Data_Formatter.h File Reference	190
13.17src/commons/Formatter/FormatData_flags_define.h File Reference . .	191
13.17.1 Function Documentation	191
13.17.1.1 DEFINE_int32	191
13.17.1.2 DEFINE_string	191
13.17.1.3 DEFINE_string	191
13.17.1.4 DEFINE_string	191
13.18src/commons/LDAUtil.cpp File Reference	192
13.19src/commons/LDAUtil.h File Reference	193
13.20src/commons/Server/DistributedMap.cpp File Reference	194
13.21src/commons/Server/DistributedMap.h File Reference	195
13.22src/commons/Server/DM_Server.cpp File Reference	196
13.22.1 Function Documentation	197
13.22.1.1 main	197
13.23src/commons/Server/DM_Server.h File Reference	198
13.23.1 Typedef Documentation	199
13.23.1.1 PNGJobPtr	199
13.23.2 Variable Documentation	199
13.23.2.1 CHUNK	199
13.23.2.2 NUM_CONSUMERS	199
13.23.2.3 QUE_FULL	199
13.24src/commons/Server/Hashmap_Array.h File Reference	200
13.25src/commons/Server/Server_Helper.h File Reference	201
13.26src/commons/TopicLearner/Checkpoint.h File Reference	202
13.27src/commons/TopicLearner/Dirichlet.cpp File Reference	203
13.27.1 Function Documentation	203
13.27.1.1 digamma	203
13.27.1.2 log_gamma	203

13.28src/commons/TopicLearner/Dirichlet.h File Reference	204
13.28.1 Function Documentation	204
13.28.1.1 digamma	204
13.28.1.2 log_gamma	204
13.29src/commons/TopicLearner/DM_Client.cpp File Reference	205
13.30src/commons/TopicLearner/DM_Client.h File Reference	206
13.30.1 Typedef Documentation	206
13.30.1.1 CookiePtr	206
13.30.1.2 PNGCallbackPtr	206
13.31src/commons/TopicLearner/Execution_Strategy.h File Reference . . .	207
13.32src/commons/TopicLearner/Filter_Eval.cpp File Reference	208
13.33src/commons/TopicLearner/Filter_Eval.h File Reference	209
13.34src/commons/TopicLearner/Filter_Optimizer.cpp File Reference . . .	210
13.35src/commons/TopicLearner/Filter_Optimizer.h File Reference	211
13.36src/commons/TopicLearner/Filter_Reader.cpp File Reference	212
13.37src/commons/TopicLearner/Filter_Reader.h File Reference	213
13.38src/commons/TopicLearner/Filter_Sampler.cpp File Reference	214
13.39src/commons/TopicLearner/Filter_Sampler.h File Reference	215
13.40src/commons/TopicLearner/Filter_Tester.cpp File Reference	216
13.41src/commons/TopicLearner/Filter_Tester.h File Reference	217
13.42src/commons/TopicLearner/Filter_Updater.cpp File Reference	218
13.43src/commons/TopicLearner/Filter_Updater.h File Reference	219
13.44src/commons/TopicLearner/Filter_Writer.cpp File Reference	220
13.45src/commons/TopicLearner/Filter_Writer.h File Reference	221
13.46src/commons/TopicLearner/GenericTopKList.h File Reference	222
13.47src/commons/TopicLearner/Main_flags_define.h File Reference . . .	223
13.47.1 Function Documentation	226
13.47.1.1 DEFINE_bool	226
13.47.1.2 DEFINE_bool	226
13.47.1.3 DEFINE_bool	226
13.47.1.4 DEFINE_bool	226

13.47.1.5 DEFINE_double	226
13.47.1.6 DEFINE_double	226
13.47.1.7 DEFINE_int32	226
13.47.1.8 DEFINE_int32	226
13.47.1.9 DEFINE_int32	226
13.47.1.10 DEFINE_int32	226
13.47.1.11 DEFINE_int32	226
13.47.1.12 DEFINE_int32	226
13.47.1.13 DEFINE_int32	226
13.47.1.14 DEFINE_int32	226
13.47.1.15 DEFINE_int32	226
13.47.1.16 DEFINE_int32	226
13.47.1.17 DEFINE_int32	226
13.47.1.18 DEFINE_int32	226
13.47.1.19 DEFINE_string	226
13.47.1.20 DEFINE_string	226
13.47.1.21 DEFINE_string	226
13.47.1.22 DEFINE_string	226
13.47.1.23 DEFINE_string	226
13.48 src/commons/TopicLearner/Model.h File Reference	227
13.49 src/commons/TopicLearner/Model_Builder.h File Reference	228
13.50 src/commons/TopicLearner/Model_Director.cpp File Reference	229
13.51 src/commons/TopicLearner/Model_Director.h File Reference	230
13.52 src/commons/TopicLearner/Model_Refiner.h File Reference	231
13.53 src/commons/TopicLearner/Parameter.cpp File Reference	232
13.54 src/commons/TopicLearner/Parameter.h File Reference	233
13.54.1 Typedef Documentation	233
13.54.1.1 param	233
13.55 src/commons/TopicLearner/Pipeline.h File Reference	234
13.56 src/commons/TopicLearner/Synchronized_Training_Execution_-Strategy.cpp File Reference	235

13.56.1 Function Documentation	235
13.56.1.1 synchronize	235
13.57src/commons/TopicLearner/Synchronized_Training_Execution_- Strategy.h File Reference	236
13.58src/commons/TopicLearner/Synchronizer.cpp File Reference	237
13.59src/commons/TopicLearner/Synchronizer.h File Reference	238
13.60src/commons/TopicLearner/Synchronizer_Helper.h File Reference	239
13.61src/commons/TopicLearner/TBB_Pipeline.cpp File Reference	240
13.62src/commons/TopicLearner/TBB_Pipeline.h File Reference	241
13.63src/commons/TopicLearner/Testing_Execution_Strategy.cpp File Ref- erence	242
13.64src/commons/TopicLearner/Testing_Execution_Strategy.h File Refer- ence	243
13.65src/commons/TopicLearner/Training_Execution_Strategy.cpp File Reference	244
13.66src/commons/TopicLearner/Training_Execution_Strategy.h File Ref- erence	245
13.67src/commons/types.h File Reference	246
13.67.1 Define Documentation	247
13.67.1.1 PRINT_TIME	247
13.67.1.2 TIME	247
13.67.2 Typedef Documentation	247
13.67.2.1 base_generator_type	247
13.67.2.2 bigppair	247
13.67.2.3 mapped_vec	247
13.67.2.4 size_int	247
13.67.2.5 tppair	247
13.67.2.6 wppair	247
13.68src/commons/WordIndexDictionary.cpp File Reference	248
13.69src/commons/WordIndexDictionary.h File Reference	249
13.70src/mainpage.h File Reference	250
13.71src/multi_mcahine_usage.h File Reference	251
13.72src/single_machine_usage.h File Reference	252

13.73src/Unigram_Model/Formatter/Unigram_Test_Data_Formatter.cpp File Reference	253
13.74src/Unigram_Model/Formatter/Unigram_Test_Data_Formatter.h File Reference	254
13.75src/Unigram_Model/Formatter/Unigram_Train_Data_Formatter.cpp File Reference	255
13.76src/Unigram_Model/Formatter/Unigram_Train_Data_Formatter.h File Reference	256
13.77src/Unigram_Model/Merge/Merge_Dictionaries.cpp File Reference .	257
13.77.1 Function Documentation	257
13.77.1.1 DEFINE_int32	257
13.77.1.2 DEFINE_string	257
13.77.1.3 DEFINE_string	257
13.77.1.4 main	257
13.78src/Unigram_Model/Merge/Merge_Topic_Counts.cpp File Reference	258
13.78.1 Function Documentation	259
13.78.1.1 DEFINE_int32	259
13.78.1.2 DEFINE_int32	259
13.78.1.3 DEFINE_string	259
13.78.1.4 DEFINE_string	259
13.78.1.5 DEFINE_string	259
13.78.1.6 main	259
13.79src/Unigram_Model/Server/Unigram_Model_Server_Helper.cpp File Reference	260
13.80src/Unigram_Model/Server/Unigram_Model_Server_Helper.h File Reference	261
13.81src/Unigram_Model/TopicLearner/eff_small_map.cpp File Reference	262
13.82src/Unigram_Model/TopicLearner/eff_small_map.h File Reference . .	263
13.83src/Unigram_Model/TopicLearner/Hadoop_Checkpointer.cpp File Reference	264
13.84src/Unigram_Model/TopicLearner/Hadoop_Checkpointer.h File Ref- erence	265
13.85src/Unigram_Model/TopicLearner/Local_Checkpointer.cpp File Ref- erence	266

13.86src/Unigram_Model/TopicLearner/Local_Checkpointer.h File Reference	267
13.87src/Unigram_Model/TopicLearner/sampler.cpp File Reference	268
13.88src/Unigram_Model/TopicLearner/sampler.h File Reference	269
13.89src/Unigram_Model/TopicLearner/TopicCounts.cpp File Reference . .	270
13.90src/Unigram_Model/TopicLearner/TopicCounts.h File Reference . . .	271
13.90.1 Typedef Documentation	271
13.90.1.1 topicCounts	271
13.91src/Unigram_Model/TopicLearner/TopKList.cpp File Reference . . .	272
13.92src/Unigram_Model/TopicLearner/TopKList.h File Reference	273
13.93src/Unigram_Model/TopicLearner/TypeTopicCounts.cpp File Reference	274
13.93.1 Function Documentation	274
13.93.1.1 cnt_cmp	274
13.93.1.2 cnt_cmp_ttc	274
13.94src/Unigram_Model/TopicLearner/TypeTopicCounts.h File Reference	275
13.95src/Unigram_Model/TopicLearner/Unigram_Model.cpp File Reference	276
13.96src/Unigram_Model/TopicLearner/Unigram_Model.h File Reference .	277
13.97src/Unigram_Model/TopicLearner/Unigram_Model_Streamers.cpp File Reference	278
13.98src/Unigram_Model/TopicLearner/Unigram_Model_Streamers.h File Reference	279
13.99src/Unigram_Model/TopicLearner/Unigram_Model_Streaming_ Builder.cpp File Reference	280
13.100src/Unigram_Model/TopicLearner/Unigram_Model_Streaming_ Builder.h File Reference	281
13.101src/Unigram_Model/TopicLearner/Unigram_Model_Synchronized_ Training_Builder.cpp File Reference	282
13.102src/Unigram_Model/TopicLearner/Unigram_Model_Synchronized_ Training_Builder.h File Reference	283
13.103src/Unigram_Model/TopicLearner/Unigram_Model_Synchronizer_ Helper.cpp File Reference	284
13.104src/Unigram_Model/TopicLearner/Unigram_Model_Synchronizer_ Helper.h File Reference	285
13.105src/Unigram_Model/TopicLearner/Unigram_Model_Tester.cpp File Reference	286

13.106rc/Unigram_Model/TopicLearner/Unigram_Model_Tester.h File Reference	287
13.107rc/Unigram_Model/TopicLearner/Unigram_Model_Testing_Builder.cpp File Reference	288
13.108rc/Unigram_Model/TopicLearner/Unigram_Model_Testing_Builder.h File Reference	289
13.109rc/Unigram_Model/TopicLearner/Unigram_Model_Trainer.cpp File Reference	290
13.110rc/Unigram_Model/TopicLearner/Unigram_Model_Trainer.h File Reference	291
13.111rc/Unigram_Model/TopicLearner/Unigram_Model_Training_Builder.cpp File Reference	292
13.112rc/Unigram_Model/TopicLearner/Unigram_Model_Training_Builder.h File Reference	293
13.113rc/usage.h File Reference	294

Chapter 1

Y!LDA Topic Modelling Framework

What is Topic Modelling?

It is a Machine Learning technique to categorize data. If we have to group the home pages of Singapore Airline, National University of Singapore & Chijmes which is a restaurant in Singapore, we can group them as all belonging to Singapore. Now if we have more pages to group lets say, United Airlines, Australian National University and a restaurant in Berkeley, then we can group the combined set of pages in multiple ways: by country, by type of business and so on. Choosing one of these different ways is hard because each one of them has multiple roles to play depending on the context. In a document that talks about nations strengths then the grouping by nationality is good and in some document which talks about universties its apt to use the grouping by type of business. So the only alternative is to assign or tag each page with all the categories it belongs to. So we tag the United Airlines page as an airliner company in US and so on.

So whats the big difference between grouping objects or clustering & Topic Models? The following example clarifies the distinction: Consider objects of different colors. Clustering them is to find that there are 3 prototypical colors RGB & each object can be grouped by what its primary color is. That is we group object by prototypes. On the other hand with topic models we try to find the the composition of RGB in the color of each object, that is we say that this color is composed of 80% R, 9% G & 11% B. So topic models are definitely richer in the sense that any color can be decomposed into the prototypical colors but not all colors can be unambiguously grouped

What is Y!LDA topic modelling framework?

Though conceptually it sounds very good, to make it work on 100s of millions of pages, with 1000s of topics to infer & no editorial data is a very hard problem. The state of the art can only handle sizes that are 10 to 100 times smaller. We would also like the solution to scale with the number of computers so that we can add more machines and be able to solve bigger problems.

One way of solving the problem of Topic Modelling is called [Latent Dirichlet Allocation](#). This is a statistical model which specifies a probabilistic procedure to generate data. It defines a topic as a probability distribution over words. Essentially think of topics as having a vocabulary of their own with preference over words specified as a probability distribution.

We have implemented a framework to solve the topic modelling problem using LDA which can work at very large scale. Considerable effort has also been spent on creating an architecture for the framework which is flexible enough to allow the reuse of infrastructure for the implementation fancier models and extension. One of the main aims here is that scaling a new model should take minimal effort. For more details please take a look at [An Architecture for Parallel Topic Models](#)

What does the framework provide?

It provides a fast C++ implementation of the inferencing algorithm which can use both multi-core parallelism and multi-machine parallelism using a hadoop cluster. It can infer about a thousand topics on a million document corpus while running for a thousand iterations on an eight core machine in one day.

What are the requirements?

Hardware Requirements:

Small Corpus of the order of thousands of documents: Dual Core machines with 4-8 GB of RAM might be sufficient. Of course you can run the code on larger document sets but you will have to wait longer or cut down on the number of iterations.

Large Corpus of the order of millions of documents: 50 to 100 multi-Core (quad cores, dual quad cores, etc) machines with 8 to 16 GB of RAM can give good performance.

Software Requirements:

The code has been mainly tested on the linux platform. If you want to install on other platforms, since the source and libraries along with source are provided, you can try compiling them and let us know if it works.

Dependencies:

The code has dependencies on a number of libraries. To facilitate distribution we have shipped the code with the sources for the associated libraries. The following is the list:

1. **Ice-3.4.1.tar.gz**
An efficient inter process communication framework which is used for the distributed storage of (topic, word) tables.
2. **cppunit-1.12.1.tar.gz**
C++ unit testing framework. We use this for unit tests.
3. **mcpp-2.7.2.tar.gz**
C++ preprocessor
4. **boostinclude.tar.gz**
Boost libraries (various datatypes)
5. **gflags-1.2.tar.gz**
Google's flag processing library (used for commandline options)
6. **protobuf-2.2.0a.tar.gz**
Protocol buffers (used for serializing data to disk and as internal key data structure). Google's serialization library
7. **bzip2-1.0.5.tar.gz**
Data compression
8. **glog-0.3.0.tar.gz**
Logfile generation (Google's log library).
9. **tbb22_20090809oss.tar.gz**
Intel Threading Building Blocks. Multithreaded processing library. Much easier to use than pthreads. We use the pipeline class.

All the libraries except Ice should install without problems. With Ice, there are a lot dependencies involved and our automated build script might not work in your set-up. If so please install Ice manually.

How do you install?

1. Run make from the directory where you have checked out the source.
2. This will first install all the required libraries locally in the same directory. It will then compile the source code to generate the binaries described next.
3. Please check for a proper install before if you see compilation or linkage errors. There may be issues with installation of Ice. Care has been taken that the install happens on most well set-up machines but if it fails, we recommend installing Ice manually and copying the include files and the libraries to the include & lib directories in the Y!LDA install path. After this the compilation should go through fine.

What are the binaries that get installed and what do you use them for?

1. `formatter` : Used to format the raw text corpus into a binary format on which `learntopics` can be run. This is a preprocessing step that allows one to run `learntopics` many times on the same corpus. It also decreases the on disk size of the raw text corpus.
2. `learntopics`: Used to learn/infer the topics from the corpus and represent the documents in the corpus as mixture of these topics.
3. [DM_Server](#): This is the server that implements a distributed hash table that is used to store the global counts table while running the multi-machine version of the code.
4. `Merge_Dictionaries`: This is used to build the global dictionary by merging the local dictionaries
5. `Merge_Topic_Counts`: This is used to dump the global counts table to disk.

What are the scripts available and what do you do with them?

Scripts are available for the use of build system and the multi-machine setup with Hadoop. The build system uses the `bin/create*.sh` scripts to find out files & directories. The multi-machine setup has the `runLDA.sh`, `Formatter.sh` & `LDA.sh` scripts that launch the hadoop-streaming job for doing the distributed inference. Another script that helps you to organize your corpus on hdfs is `splitter.sh`

How do I use Y! LDA?

[Using Y!LDA](#)

Where to find the developer documentation?

[Y!LDA Architecture](#) & the code documentation made available through Doxygen

Chapter 2

Y!LDA Architecture

2.1 Introduction

Please refer the Main Page for an introduction

2.2 Goals

The approach to do topic modelling is to have a graphical model representing the generative assumptions the user has about the corpus. A graphical model is a probabilistic model representing the joint distribution of the random variables involved with a graph denoting the conditional independence assumptions amongst them. Solving the model is to infer the parameters of the model by processing the actual data. Doing this inference is the hardest part of the approach.

2.2.1 Adding new Models

There are a lot of variations on the basic LDA model and with each variation the inferencing logic changes. The parameters, the sufficient statistics that need to be maintained everything will be slightly different. One of the main goals of this framework is to make the job of adding new models simpler.

2.2.2 Common Infrastructure

One task that is mostly common across multiple models is the infrastructure needed to store documents, load them, create a pipeline that can optimally utilize multi-core parallelism. In the framework we aim to standardize on proven infrastructure that is known to provide efficient implementations so that the model writer just worries about adding only the parts that are relevant for doing the inference

2.2.3 Scalability

Another main aspect of this framework is to substantially increase the scale of the state of the art by utilizing parallelism both multi-core and multi-machine.

2.3 Main components of the System

Y!LDA uses the Gibbs sampling approach popularized by [Collapsed Gibbs Sampling](#). There are four main components in this approach:

1. **Model:**

This encapsulates the parameters of the model and the sufficient statistics that are necessary for inference

2. **Model_Refiner:**

This encapsulates the logic needed for refining the initial model which involves streaming the documents from disk, sampling new topic assignments, updating the model, performing diagnostics and optimization and writing the documents back to disk

3. **Pipeline:**

As can be seen above, the refiner does a sequence of operations on every document of the corpus. Some of them have to run serially but some others can be run parallelly. To enable exploiting multi-core parallelism, the **Pipeline** is defined to be composed of a set of operations called filters which can either be declared to be run serially or parallelly. The **Pipeline** comes with a scheduler that schedules the threads available on the machine to run these filters in an optimal fashion

4. **Execution_Strategy:**

This encapsulates the strategy that decides what filters a pipeline is composed of, how many times the documents are passed through the pipeline.

2.3.1 Builder Pattern

The Builder pattern fits very well for this approach. We implement a **Model_Builder** that builds the last three components depending on what model is needed and what mode the model is supposed to operate in.

The **Model_Builder** creates an initial **Model** and creates the required **Model_Refiner** by passing the Model(or the necessary components of the Model). It then creates a **Pipeline** and an **Execution_Strategy** as per the mode of operation.

The Director is pretty straightforward. It directs the given **Model_Builder** to create the necessary components and executes the defined **Execution_Strategy**. This refines the initial **Model** created by the builder into one that reflects parameters tuned to the corpus on which the **Model** was refined on. Then the **Model** is stored on disk for testing.

2.4 Distributed Set Up

To cater to the Scalability goals, as detailed in **An Architecture for Parallel Topic Models**, the framework implements a Distributed Memory based multi-machine setup that exploits multi-machine parallelism to the fullest. The main idea being that the inferencing happens locally while the state variables are kept up-to-date with a global copy that stored using a Distributed HashTable. To come up with an efficient distributed set up is a difficult thing and we definitely do not want

people reinvent the wheel here. So the framework tries to abstract the mechanism of distribution, the implementation of an efficient distributed HashTable and the mechanism needed for Synchronization.

2.4.1 Distributed_Map

The framework implements a Distributed_Map interface using Ice as a very efficient middleware. It essentially provides both a Server and [Client](#) implementation.

1. [DM_Server](#):

The server essentially hosts a chunk of the distributed hash table and supports the usual map operations. It also supports three special operations:

- Put: Which accumulates the values instead of replacing
- waitForAll: Which is a barrier implementation using AMD
- PutNGet: which is an asynchronous call that accumulates the passed value into the existing one and returns the final value back to the caller through a call back mechanism

2. [DM_Client](#):

A client that supports a single hash table view of the distributed system. The client transparently supports a rate limited, sliding-window based Asynchronous Method Invocation for the PutNGet which is a very useful operation to have for effective Synchronization. Refer to the VLDB paper for more information.

For most models, one need not worry about modifying the above. These only need to be used most of the times without bothering much about their implementation.

2.4.2 Synchronizer

The framework provides a default implementation of the Synchronization strategy detailed in [2]. The [Synchronizer](#) is run as a separate background thread apart from the main threads that do the inferencing. The actual task of synchronization is left to the implementation of a [Synchronizer_Helper](#) class. The Synchronizer only creates slots for synchronization and asks the helper to synchronize in those slots. It also takes care of running the Synchronization only till the inferencing is done.

However, there is a strong assumption that the synchronization proceeds in a linear fashion. That is the structures being synchronized are linear and can be synchronized one after the other. This is implicit in the Synchronizer's creation of slots.

2.4.3 Synchronizer_Helper

Every model has to only provide the [Synchronizer_Helper](#) implementation which spills the logic for synchronizing the model's relevant structures, maintains copies of them where needed and provides the callback function for the AMI putNGet.

2.5 Default Implementations provided

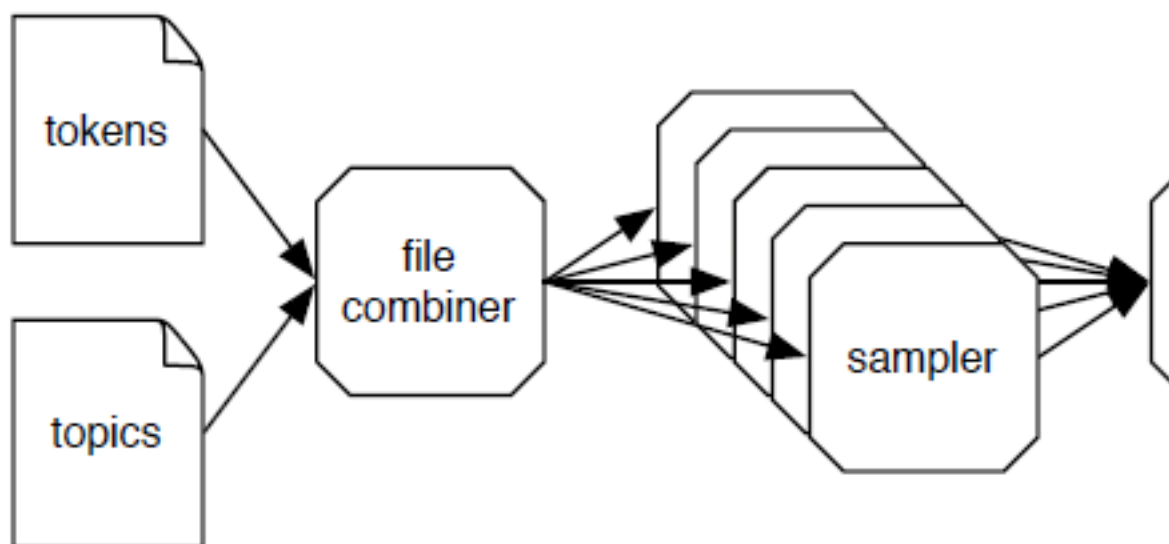
The framework provides default implementations for the [Pipeline](#) interface and the [Execution_Strategy](#) interface.

1. [TBB_Pipeline](#):

This implementation uses Intel's Threading Building Blocks for providing the [Pipeline](#) interface.

2. [Training_Execution_Strategy](#):

The default implementation of [Execution_Strategy](#) for LDA training. Assembles the following pipeline for data flow:



3. [Synchronized_Training_Execution_Strategy](#):

The default implementation of [Execution_Strategy](#) that extends [Training_Execution_Strategy](#) and adds Synchronization capability

4. [Testing_Execution_Strategy](#):

The default implementation of [Execution_Strategy](#) for LDA testing.

2.6 Unigram Model

The framework also provides the [Unigram_Model](#) implementations of the various common interfaces. This is the basic LDA model with the bag of words assumption. Please take a look at how the various interfaces are implemented. The main implementation needed is for [Model](#) & [Model_Refiner](#). Additionally, it implements efficient sparse data structures to store the sufficient statistics.

2.7 Adding a new Model

Please use the [Unigram_Model](#) implementation as an example to implement new models

2.8 Checkpoints

The framework also provide checkpointing functionality for the multi-machine setup in order to provide failure recovery. This is implemented by an external object that knows how to do three things: a. Serialize metadata to disk b. load previously serialized metadata on request c. Serialize the datastructures to disk

An appropriate checkpointer is passed as an argument while creating an [Execution_Strategy](#) The strategy uses checkpointers to checkpoint at regular intervals. At startup, it also checks if any checkpoints are available and if so, it starts up from that checkpoint.

Different checkpointers are needed for different setups. For ex., the framework uses the Local [Checkpoint](#) when running in single machine mode which only involves writing the iteration number as metadata. All other data needed for restart is already being serialized. However, for the multi-machine setup, a different mechanism is needed and a Hadoop [Checkpoint](#) is implemented.

This is an ongoing effort and we will add more stuff both to the code and documentation. We definitely need your help & contribution in making this better.

Here is an initial set of TODOs:

Todo

- Add unit tests to make the code more robust
- Add more code documentation for the [Unigram_Model](#) components
- Implement fancier models in later versions
- Implement extensions to the LDA model in later versions

These are in no particular order and we might re-prioritize later. Please mail me if you are interested in contributing

We shall use the git pull request (fork + pull model) for collaborative development.

Chapter 3

Multi-Machine Setup

Please take a look at [Single Machine Usage](#) for information on running individual commands. Here we give you ways to run those individual commands on multiple machines. So, we are not repeating the details on the individual commands.

1. Using Hadoop

(a) Organize your corpus on HDFS:

- i. Run `splitter.sh "queue" "orig-corpus" "organized-corpus" "chunks"`

The splitter program is a simple map-reduce streaming script that splits your original corpus into chunks gzip files. This enables Y!LDA to process one chunk on one machine. We advise you to use one full machine to run one instance of Y!LDA as the memory requirements may be large depending on your corpus size. A very large corpus of the order of 10 to 20 Million moderately sized documents can need anywhere from 5 to 6 GB of memory. You can reduce this by using more machines though. For ex., if you have a corpus of 1.5 Million documents you might want to split it across 8 machines using:

```
splitter.sh "queue" "orig-corpus"
"organized-corpus" 8
```

Run Y!LDA on the organized corpus:

- (b)
 - i. Assuming you have a homogenous setup, install Y!LDA on one machine.
 - ii. Run make jar to create LDALibs.jar file with all the required libraries and binaries
 - iii. Copy LDALibs.jar to HDFS
 - iv. Figure out the max memory allowed per map task for your cluster and use the same in the script via the maxmem parameter. This can be done by looking at any job conf (job.xml) and searching the value of "mapred.cluster.max.map.memory.mb" property.
 - v. Run `runLDA.sh 1 [train|test] "queue" "organized-corpus" "output-dir" "max-mem" "topics" "iters" "full_hdfs_path_of_LDALibs.jar" ["training-output"]`
 - vi. This starts a map-only script on each machine. The script starts the [DM_Server](#) on all the machines. Then Y!LDA is run on each machine. The input is one chunk of the corpus.
 - vii. For testing, use the test flag and provide directory storing the training output.

(c) Output generated

- i. Creates <chunks> folders in <output-dir> one for each client.
- ii. Each of these directories hold the same output as the single machine case but from different clients.

- iii. `<output-dir>/<client-id>/learntopics.WARNING` contains the output written to stderr by client `<client-id>`
- iv. `<output-dir>/<client-id>/lda.docToTop.txt` contains the topic proportions assigned to the documents in the portion of the corpus allotted to client `<client-id>`
- v. `<output-dir>/<client-id>/lda.topToWor.txt` contains the salient words learned for each topic. This remains almost same across clients. So you can pick one of these as the salient words per topic for the full corpus.
- vi. `<output-dir>/<client-id>/lda.ttc.dump` contains the actual model. Even this like the salient words is almost same across clients and any one can be used as the model for the full corpus.
- vii. `<output-dir>/global` contains the dump of the global dictionary and the partitioned global topic counts table. These are generated in the training phase and are critical for the test option to work.

Viewing progress

- (d) i. The stderr output of the code will be redirected into hadoop logs. So you can check the task logs from the tracking URL displayed in the output of `runLDA.sh` to see what is happening

(e) Failure Recovery

We provide a check-pointing mechanism to handle recovery from failures. The current scheme works in local mode and for distributed mode using Hadoop. The reason for this being that the distributed check-pointing uses the hdfs to store the check-points. The following is the process:

- i. The formatter task is run on the inputs and the formatted input is stored in a temporary location.
- ii. The `learntopics` task is run using the temporary location as an input and the specified output as the output directory. Care is taken to start the same number of mappers for both the formatter and `learntopics` tasks. The input is a dummy directory structure with dummy directories equal to the number of mappers.
- iii. Each `learntopics` task copies its portion of the formatted input by `dfs copy_to_local` the folder corresponding to its `mapred_task_partition`.
- iv. Runs `learntopics` with the temporary directory containing the formatted input as a check-point directory. So all information needed to start `learntopics` from the previous check-pointed iteration is available locally and any progress made is written back to the temporary input directory.

This mechanism is utilized by the scripts to detect failure cases and attempt to re-run the task again from the previous checkpoint. As `learntopics` is designed to check if check-point metadata is available in the working directory and use it to start-off from there a separate restart option is obviated. As a by product one gets the facility of doing incremental runs, that is, to run say 100 iterations, check the output and run the next 100 iterations if

needed. The scripts detect this condition and ask you if you want to start-off from where you left or restart from the beginning.

The scripts are designed in such a fashion that these happen transparently to the user. This is information for developers and for cases where the recovery mechanism could not handle the failure in the specified number of attempts. Check the stderr logs to see what the reason for failure is. Most times it is due to wrong usage which results in unrecoverable aborts. If you think its because of a flaky cluster, then try increasing the number of attempts. If nothing works and you think there is a bug in the code please let us know.

1. Using SSH - Assume you have 'm' machines

- (a) If you have a homogenous set up, install Y!LDA on one machine, run make jar and copy LDALibs.jar to all the other machines in the set up. Else install Y!LDA on all machines.
- (b) Split the corpus into 'm' parts and distribute them to the 'm' machines
- (c) Run formatter on each split of the corpus on every machine.
- (d) Run the Distributed_Map Server on each machine as a background process using nohup:

```
nohup ./DM_Server <model> <server_id> <num_clients> <host:port>
--Ice.ThreadPool.Server.SizeMax=9 &
```

- i. model: an integer that represents the model. Set it to 1 for [Unigram-Model](#)
- ii. server_id: a number that denotes the index of this server in the list of servers that is provided to 'learntopics'. If server1 has h:p, 10.1.1.1:10000 & is assigned id 0, server2 has h:p, 10.1.1.2:10000 & is assigned id 1, the list of servers that is provided to 'learntopics' has to be 10.1.1.1:10000, 10.1.1.2:10000 and not the other way around.
- iii. num_clients: a number that denotes the number of clients that will access the Distributed Map. This is usually equal to 'm'. This is used to provide a barrier implementation
- iv. host:port- the port and ip address on which the server must listen on

Run Y!LDA on the corpus:

- (e) i. `learntopics --topics=<topics> --iter=<iter> --servers=<list-of-servers> --chkptdir="/tmp" --chkptinterval=10000`
 - A. <list-of-servers>: The comma separated list of ip:port numbers of the servers involved in the set-up. The index of the ip:port numbers should be as per the server_id parameter used in starting the server

- B. `chkptdir` & `chkptinterval`: These are currently used only with the Hadoop set-up. Set `chkptdir` to something dummy. In order that the checkpointing code does not execute, set the `chkptinterval` to a very large value or some number greater than the number of iterations
- Create Global Dictionary - Run the following on server with id 0. Assuming `learntopics` was run in the folder `/tmp/corpus`
- ii. A. `mkdir -p /tmp/corpus/global_dict; cd /tmp/corpus/global_dict;`
 - B. `scp server_i:/tmp/corpus/lda.dict.dump lda.dict.dump.i` where the variable 'i' is the same as the `server_id`.
 - C. Merge Dictionaries
`Merge_Dictionaries --dictionaries=m`
`--dumpprefix=lda.dict.dump`
 - D. `mkdir -p ../global; mkdir -p ../global/topic_counts; cp lda.dict.dump ../global/;`
 - iii. Create a sharded Global Word-Topic Counts dump - Run on every machine in the set-up
 - A. `mkdir -p /tmp/corpus/global_top_cnts; cd /tmp/corpus/global_top_cnts;`
 - B. `scp server_0:/tmp/corpus/global/lda.dict.dump lda.dict.dump.global`
 - C. `Merge_Topic_Counts --topics=<topics>`
`--clientid=<server-id>`
`--servers=<list-of-servers>`
`--globaldictionary="lda.dict.dump.global"`
 - D. `scp lda.ttc.dump server_0:/tmp/corpus/global/topic_counts/lda.ttc.dump.$server-id`
 - Copy the parameters dump file to global dump - Run on server_0
 - iv. A. `cd /tmp/corpus; cp lda.par.dump global/topic_counts/lda.par.dump`
 - v. This completes training and the model is available on `server_0:/tmp/corpus/global`
 - (f) Running Y!LDA in test mode: Run on `server_0`. Assuming test corpus is in `/tmp/test_corpus`
 - i. `cd /tmp/test_corpus;`
 - ii. `cp -r ../corpus/global .`
 - iii. `learntopics -teststream --dumpprefix=global/topic_counts/lda --numdumps=m --dictionary=global/lda.dict.dump --maxmemory=2048 --topics=<topics>`

- iv. cat all your documents, in the same format that 'formatter' expects, to the above command's stdin

Chapter 4

Single Machine Setup

First step is to set LD_LIBRARY_PATH. This can be done by sourcing the script setLibVars.sh

Assume Y!LDA is installed in \$LDA_HOME.

```
cd $LDA_HOME
```

```
source ./setLibVars.sh
```

Basic Usage:

1. [Learning a Model](#)
2. [Using the Model](#)
3. [Output Generated](#)
4. [Customization](#)

4.1 Learning a Model

The process of learning a model has four steps:

1. [Tokenization and Formatting](#)
2. [Learning the topic mixtures](#)
3. [Viewing the word mixtures for each topic](#)
4. [Viewing the topic assignments](#)

4.1.1 Tokenization and Formatting

The tokenizer converts text into tokens that undergone some basic normalizations. Most likely you want to write your own. A Java version is provided for convenience more than anything else. This is a simple java class that tokenizes the file by splitting the stream around non-character[^a-zA-Z] boundaries into word tokens. So, it ignores numbers & punctuation currently. The raw text corpus is assumed to have the following format. Its a single file containing documents on every line. Each document has the format:

```
doc-id<space>aux-id<space>word1[<space>word2]*
```

The tokenizer just writes the tokens appended to the **doc-id** & **aux-id** to stdout.

The 'formatter' converts the text into GoogleProtocolBuffer messages. This is done to ensure that we have a very small file footprint on disk. It takes the tokenized corpus

supplied as input and formats it into the internal format needed for `learntopics`. The primary output is the documents in the internal format. The program by default removes stop words supplied statically in the [src/commons/constants.h](#) file. It then creates the dictionary from the remaining words.

The following is an illustration of this step. It uses the training set that is available with the package (`ut_out/ydir_1k.txt`)

```
$ cd $LDA_HOME/ut_out
$ cp ../Tokenizer.java .
$ javac Tokenizer.java
$ ls -al ydir_1k.txt
-rw-r--r-- 1 shravanm shravanm 1818848 2011-04-17 17:34
ydir_1k.txt
$ cat ydir_1k.txt | java Tokenizer | ../formatter
W0417 17:35:44.967370 19605 Controller.cpp:83]
-----
W0417 17:35:44.967788 19605 Controller.cpp:84] Log
files are being stored at /home/shravanm/workspace/LDA_-
Refactored/ut_out/unigram/formatter.*
W0417 17:35:44.967808 19605 Controller.cpp:85]
-----
W0417 17:35:44.968111 19605 Controller.cpp:91] Assuming
that corpus is being piped through stdin. Reading from
stdin...
W0417 17:35:45.652430 19605 Controller.cpp:107]
Formatting Complete. Formatted document stored in
lda.wor. You have used lda as the output prefix.
Make sure you use the same as the input prefix for
learntopics
W0417 17:35:45.652497 19605 Controller.cpp:113]
Dumping dictionary for later use by learntopics into
lda.dict.dump
W0417 17:35:45.674201 19605 Controller.cpp:117] Finished
dictionary dump
W0417 17:35:45.674236 19605 Controller.cpp:118]
Formatting done
W0417 17:35:45.674253 19605 Controller.cpp:122] Total
number of unique words found: 17208
```

```

W0417 17:35:45.674270 19605 Controller.cpp:123] Total num
of docs found: 900

W0417 17:35:45.674288 19605 Controller.cpp:124] Total num
of tokens found: 182525

$ ls -l lda*

-rw-r--r-- 1 shravanm shravanm 204655 2011-04-17 17:35
lda.dict.dump

-rw-r--r-- 1 shravanm shravanm 397983 2011-04-17 17:35
lda.wor

```

4.1.2 Learning the topic mixtures

'learntopics' learns a topic model from a corpus which has been formatted using 'formatter'. The input to this are the files generated by 'formatter'. So you need to run this in the same directory that contains these files. The output from this step contains two types of files: binary files that are used by other modes of operation like batch and stream mode testing and text files which are human readable and give a sense of what has happened. The generated output will be explained in detail in the next sections. But the primary outputs are the topic assignments to the documents in the corpus and the word mixtures that represent each topic that has been learnt.

Continuing with the illustration of step 1 we do the following to learn the model: Assuming that PWD=\$LDA_HOME/ut_out

```

$ ../learntopics --topics=100 --iter=500

Log file created at: 2011/04/17 17:44:04

Running on machine: offerenjoy

Log line format: [IWEF]mmdd hh:mm:ss.uuuuuu threadid
file:line] msg

W0417 17:44:04.082424 19628 Controller.cpp:68]
-----

W0417 17:44:04.082855 19628 Controller.cpp:69] Log
files are being stored at /home/shravanm/workspace/LDA_-
Refactored/ut_out/unigram/learntopics

W0417 17:44:04.082871 19628 Controller.cpp:70]
-----

W0417 17:44:04.083418 19628 Controller.cpp:81] You have
chosen single machine training mode

W0417 17:44:04.083976 19628 Unigram_Model_Training_-
Builder.cpp:43] Initializing Dictionary from

```

```
lda.dict.dump
W0417 17:44:04.137420 19628 Unigram_Model_Training_-
Builder.cpp:45] Dictionary Initialized
W0417 17:44:04.174685 19628 Unigram_Model_Trainer.cpp:24]
Initializing Word-Topic counts table from docs lda.wor,
lda.top using 17208 words & 100 topics.
W0417 17:44:04.250243 19628 Unigram_Model_Trainer.cpp:26]
Initialized Word-Topic counts table
W0417 17:44:04.250313 19628 Unigram_Model_Trainer.cpp:29]
Initializing Alpha vector from Alpha_bar = 50
W0417 17:44:04.250356 19628 Unigram_Model_Trainer.cpp:31]
Alpha vector initialized
W0417 17:44:04.250375 19628 Unigram_Model_Trainer.cpp:34]
Initializing Beta Parameter from specified Beta = 0.01
W0417 17:44:04.250395 19628 Unigram_Model_Trainer.cpp:37]
Beta param initialized
W0417 17:44:04.251711 19628 Training_Execution_-
Strategy.cpp:35] Starting Parallel training Pipeline
W0417 17:44:04.590452 19628 Training_Execution_-
Strategy.cpp:53] Iteration 1 done. Took 0.00564301 mins
W0417 17:44:04.700546 19628 Unigram_Model.cpp:92] Average
num of topics assigned per word = 4.17091
W0417 17:44:04.700599 19628 Training_Execution_-
Strategy.cpp:58] >>>>>>>> Log-Likelihood (model,
doc, total): -1.47974e+06 , -815572 , -2.29531e+06
W0417 17:44:04.703927 19628 Unigram_Model_-
Trainer.cpp:478] Restarting IO
W0417 17:44:04.992218 19628 Training_Execution_-
Strategy.cpp:53] Iteration 2 done. Took 0.00477359 mins
W0417 17:44:04.995560 19628 Unigram_Model_-
Trainer.cpp:478] Restarting IO
W0417 17:44:05.261360 19628 Training_Execution_-
Strategy.cpp:53] Iteration 3 done. Took 0.00439862 mins
W0417 17:44:05.264669 19628 Unigram_Model_-
Trainer.cpp:478] Restarting IO
W0417 17:44:05.516330 19628 Training_Execution_-
Strategy.cpp:53] Iteration 4 done. Took 0.00416395 mins
```

```
...  
...  
W0417 17:45:59.021239 19628 Unigram_Model_-  
Trainer.cpp:478] Restarting IO  
W0417 17:45:59.242962 19628 Training_Execution_-  
Strategy.cpp:53] Iteration 498 done. Took 0.00366346  
mins  
W0417 17:45:59.246363 19628 Unigram_Model_-  
Trainer.cpp:478] Restarting IO  
W0417 17:45:59.469266 19628 Training_Execution_-  
Strategy.cpp:53] Iteration 499 done. Took 0.00368553  
mins  
W0417 17:45:59.472687 19628 Unigram_Model_-  
Trainer.cpp:478] Restarting IO  
W0417 17:45:59.513748 19628 Training_Execution_-  
Strategy.cpp:53] Iteration 500 done. Took 0.000653982  
mins  
W0417 17:45:59.616693 19628 Unigram_Model.cpp:92] Average  
num of topics assigned per word = 1.85268  
W0417 17:45:59.616768 19628 Training_Execution_-  
Strategy.cpp:58] >>>>>>>>> Log-Likelihood (model,  
doc, total): -1.04167e+06 , -425400 , -1.46707e+06  
W0417 17:45:59.620058 19628 Unigram_Model_-  
Trainer.cpp:478] Restarting IO  
W0417 17:45:59.622022 19628 Training_Execution_-  
Strategy.cpp:66] >>>>>>>>> Check Pointing at  
iteration: 500  
W0417 17:45:59.625076 19628 Training_Execution_-  
Strategy.cpp:72] Parallel training Pipeline done  
W0417 17:45:59.625128 19628 Controller.cpp:128] Model has  
been learnt  
W0417 17:45:59.627476 19628 Unigram_Model.cpp:105] Saving  
model for test pipeline in lda.ttc.dump and lda.par.dump  
W0417 17:45:59.695773 19628 Unigram_Model.cpp:108] Model  
saved  
W0417 17:45:59.695843 19628 Unigram_Model.cpp:117]  
Writing top words identified per topic into  
lda.topToWor.txt
```

```
W0417 17:45:59.701156 19628 Unigram_Model.cpp:139] Word
statistics per topic written

W0417 17:45:59.701257 19628 Unigram_Model_Training_-
Builder.cpp:126] Saving document to topic-proportions
in lda.docToTop.txt

W0417 17:45:59.701279 19628 Unigram_Model_Training_-
Builder.cpp:127] Saving word to topic assignment in
lda.worToTop.txt

W0417 17:45:59.911658 19628 Unigram_Model_Training_-
Builder.cpp:193] Document to topic-proportions saved in
lda.docToTop.txt

W0417 17:45:59.911726 19628 Unigram_Model_Training_-
Builder.cpp:194] Word to topic assignment saved in
lda.worToTop.txt

$ ls -al lda*

-rw-r--r-- 1 shravanm shravanm 4 2011-04-17 17:45
lda.chk

-rw-r--r-- 1 shravanm shravanm 204655 2011-04-17 17:35
lda.dict.dump

-rw-r--r-- 1 shravanm shravanm 168223 2011-04-17 17:45
lda.docToTop.txt

-rw-r--r-- 1 shravanm shravanm 816 2011-04-17 17:45
lda.par.dump

-rw-r--r-- 1 shravanm shravanm 230304 2011-04-17 17:45
lda.top

-rw-r--r-- 1 shravanm shravanm 38415 2011-04-17 17:45
lda.topToWor.txt

-rw-r--r-- 1 shravanm shravanm 267631 2011-04-17 17:45
lda.ttc.dump

-rw-r--r-- 1 shravanm shravanm 397983 2011-04-17 17:35
lda.wor

-rw-r--r-- 1 shravanm shravanm 2213071 2011-04-17 17:45
lda.worToTop.txt
```

4.1.3 Viewing the word mixtures for each topic

```
$ cat lda.topToWor.txt

Topic 0: (center,0.171509) (diving,0.110265)
```

(scuba,0.0857668) (equipment,0.0686183) (mark,0.0637188)
 (olympic,0.0465703) (rescue,0.0441205) (aquatic,0.039221)
 (ymca,0.039221) (family,0.0367712) (ruiz,0.0343214)
 (dive,0.0318716) (sports,0.0294219) (divers,0.0294219)
 (safety,0.0294219) (minnesota,0.0294219)
 (orlando,0.0294219) (advanced,0.0294219)
 (training,0.0269721) (international,0.0245223)

Topic 1: (beach,0.138171) (resort,0.0760789)
 (florida,0.0714219) (world,0.062108) (center,0.0543465)
 (experience,0.0527942) (vacation,0.0481372)
 (fl,0.0465849) (located,0.0450326) (offers,0.0450326)
 (holiday,0.0403757) (south,0.0403757) (rates,0.0403757)
 (beautiful,0.0388233) (location,0.0341664)
 (sea,0.0341664) (activities,0.0341664)
 (resorts,0.0326141) (accommodations,0.0326141)
 (perfect,0.0326141)

...

Topic 25: (surf,0.381466) (surfing,0.0727078)
 (surfboards,0.0573127) (surfboard,0.0547468)
 (board,0.0530363) (beach,0.0367858) (shop,0.0367858)
 (malibu,0.0350753) (longboard,0.0325094)
 (boards,0.0325094) (gear,0.0299435) (bags,0.022246)
 (travel,0.022246) (racks,0.0213907) (wetsuits,0.0205354)
 (clothing,0.0205354) (california,0.0196801)
 (lessons,0.0188248) (ocean,0.016259) (surfers,0.0154037)

...

Topic 35: (credit,0.101053) (card,0.0893533)
 (money,0.0680813) (orders,0.0670177) (make,0.060636)
 (mail,0.0574452) (cards,0.0563816) (valley,0.055318)
 (paypal,0.0521272) (accept,0.0510636) (secure,0.0468092)
 (purchase,0.0468092) (time,0.0372368) (apple,0.0351095)
 (checks,0.0351095) (email,0.0351095) (address,0.0287279)
 (due,0.0276643) (special,0.0255371) (people,0.0234099)

...

Topic 43: (racquet,0.12544) (racquets,0.107418)
 (wilson,0.101651) (head,0.094442) (tennis,0.0908377)
 (prince,0.0821871) (shoes,0.0648861) (babolat,0.0367719)
 (pro,0.0367719) (bags,0.030284) (string,0.0266796)
 (men,0.0259588) (tour,0.0237961) (mp,0.0223544)
 (rackets,0.0223544) (ncode,0.0223544) (grips,0.0223544)
 (women,0.0223544) (dunlop,0.0216335) (nike,0.0194709)

...

```

Topic 81: (gear,0.151097) (water,0.0828022)
(swim,0.0786631) (floatation,0.0621068)
(html,0.0600373) (pro,0.0579677) (exercise,0.0538286)
(mask,0.0496896) (swimming,0.04762) (aqua,0.0414114)
(watergearwg,0.0372724) (aquajoggeraj,0.0372724)
(kids,0.0352028) (goggles,0.0331333)
(zoggszoggs,0.0310637) (fins,0.0310637) (hydro,0.0289942)
(snorkel,0.0269247) (snorkeling,0.0269247)
(caps,0.0269247)

...

Topic 96: (horse,0.190855) (horses,0.126676)
(riding,0.101343) (training,0.0996538) (farm,0.0785425)
(equestrian,0.0667202) (dressage,0.0472978)
(equine,0.0405421) (boarding,0.0380088)
(show,0.0337865) (stables,0.0270309) (lessons,0.0194308)
(michigan,0.0185864) (tack,0.0185864) (riders,0.0185864)
(ranch,0.016053) (farms,0.0152086) (pony,0.0152086)
(ponies,0.0143641) (sale,0.0135197)

```

4.1.4 Viewing the topic assignments

```

$ cat lda.worToTop.txt

www.teddybears.com/ recreation/toys (teddy,36)
(bears,36) (enjoy,66) (teddy,36) (bears,36) (enjoy,66)
(featuring,61) (teddy,36) (bears,36) (teddy,36)
(bear,36) (related,77) (information,66) (everyday,38)
(fun,44) (enjoy,66) (learn,2) (enter,28) (love,61)
(lord,51) (god,51) (heart,48) (soul,77) (strength,80)
(commandments,2) (give,13) (today,28) (hearts,61)
(impress,61) (house,80) (gates,60) (deuteronomy,36)
(site,66) (sponsored,77) (brown,48) (brehm,36) (bears,36)
(teddy,36) (bear,36) (artists,63) (teddy,36) (bears,36)
(teddy,36) (bear,36) (classifieds,61) (teddy,36)
(bear,36) (clubs,77) (teddy,36) (bear,36) (events,77)
(teddy,36) (bear,36) (retailers,66) (teddy,36) (bear,36)
(magazines,61) (teddy,36) (bear,36) (books,66) (teddy,36)
(bear,36) (history,28) (web,66) (page,66) (design,16)
(graphic,61) (elements,45) (embedded,61) (html,81)
(coding,48) (created,16) (copyrighted,22) (kelly,6)
(brown,48) (brehm,36) (rights,16) (reserved,16)

www.bearsbythesea.com/ recreation/toys (teddy,36)
(bear,36) (store,20) (pismo,18) (beach,25)
(california,25) (specialize,56) (muffy,14) (store,20)

```

```
(complete,20) (collections,87) (checkout,91) (web,66)
(site,66) (muffy,14) (muffy,14) (interested,66)
(information,56) (price,3) (guides,77) (forums,28)
(newsletters,78) (follow,56) (information,66) (link,78)
(interested,66) (purchasing,4) (items,4) (follow,56)
(online,20) (store,20) (link,78) (online,20) (store,20)
(information,66)

www.the-toybox.com recreation/toys (party,0)
(supplies,38) (wiggles,56) (licensed,19) (characters,83)
(jay,0) (jay,0) (jet,83) (cabbage,49) (patch,49)
(play,61) (tents,59) (activity,91) (master,0) (roll,57)
(building,90) (toy,83) (building,90) (racing,50)
(beanie,19) (babies,13) (personalized,19) (toys,83)
(requires,85) (minimum,12) (order,19) (enter,91)
(coupon,90) (code,90) (good,61) (coupons,90) (mail,56)
(order,19) (info,57) (order,13) (options,78) (return,4)
(policy,9) (shipping,13) (rates,1) (email,56) (wiggles,0)
(party,19) (supplies,38) (toys,83) (accessories,13)
(jay,0) (jay,0) (jet,83) (plane,91) (party,19)
(supplies,90) (toys,83) (accessories,13) (toys,83)
(party,19) (supplies,90) (licensed,19) (characters,83)
(toys,83) (sell,56) (free,13) (internet,85) (premier,78)
(shopping,83) (network,61) (discount,13) (shopping,4)
(internet,19) (click,78) (visit,66)
```

4.2 Using the Model

Once the model has been trained, you can provide the binary files as a parameter to 'learntopics' to infer topic mixtures on new documents. The following illustration uses the test set that is available with the package (ut_test/ydir_1k.tst.txt).

As explained in the overview, there are two modes in which the above learnt model can be used:

1. [Batch Mode](#)
2. [Streaming Mode](#)

4.2.1 Batch Mode

Assuming that PWD=\$LDA_HOME/ut_test.

First Tokenize and format the test data.

```
$ cp ../ut_out/Tokenizer.class .
```



```
$ cat ydir_1k.tst.txt | java Tokenizer | ../formatter
--dumpfile=../ut_out/unigram/lda.dict.dump
W0417 22:16:41.834889 20929 Controller.cpp:83]
-----
W0417 22:16:41.835304 20929 Controller.cpp:84] Log
files are being stored at /home/shravanm/workspace/LDA_
Refactored/ut_test/formatter.*
W0417 22:16:41.835325 20929 Controller.cpp:85]
-----
W0417 22:16:41.835626 20929 Controller.cpp:91] Assuming
that corpus is being piped through stdin. Reading from
stdin...
W0417 22:16:41.835649 20929 Controller.cpp:97] Will use
the dictionary dump ../ut_out/unigram/lda.dict.dump to
load the global dictionary.
W0417 22:16:41.836802 20929 Unigram_Test_Data_
Formatter.cpp:14] Initializing Dictionary from ../ut_
out/unigram/lda.dict.dump
W0417 22:16:41.900626 20929 Unigram_Test_Data_
Formatter.cpp:16] Num of unique words: 17208
W0417 22:16:42.145593 20929 Controller.cpp:107]
Formatting Complete. Formatted document stored in
lda.wor. You have used lda as the output prefix.
Make sure you use the same as the input prefix for
learntopics
W0417 22:16:42.145661 20929 Controller.cpp:115] Induced
local dictionary being dumped to lda.dict.dump
W0417 22:16:42.150445 20929 Controller.cpp:117] Finished
dictionary dump
W0417 22:16:42.150481 20929 Controller.cpp:118]
Formatting done
W0417 22:16:42.150507 20929 Controller.cpp:122] Total
number of unique words found: 3506
W0417 22:16:42.150533 20929 Controller.cpp:123] Total num
of docs found: 100
W0417 22:16:42.150559 20929 Controller.cpp:124] Total num
of tokens found: 16394
```

Note the --dumpfile flag. Here is where you provide the dictionary of words that the

model knows about. Only those words in the test data set are recognized and the rest are ignored.

```
$ ../learntopics -test --dumpprefix=../ut_out/unigram/lda
--topics=100
```

```
W0417 22:22:00.932081 20947 Controller.cpp:68]
```

```
-----
W0417 22:22:00.932518 20947 Controller.cpp:69] Log
files are being stored at /home/shravanm/workspace/LDA_-
Refactored/ut_test/learntopics.*
```

```
W0417 22:22:00.932539 20947 Controller.cpp:70]
```

```
-----
W0417 22:22:00.933087 20947 Controller.cpp:92] You have
chosen single machine testing mode
```

```
W0417 22:22:00.933671 20947 Unigram_Model_Training_-
Builder.cpp:43] Initializing Dictionary from
lda.dict.dump
```

```
W0417 22:22:00.945226 20947 Unigram_Model_Training_-
Builder.cpp:45] Dictionary Initialized
```

```
W0417 22:22:00.998955 20947 Unigram_Model_Tester.cpp:33]
Initializing Word-Topic counts table from dump ../ut_-
out/unigram/lda.ttc.dump using 3506 words & 100 topics.
```

```
W0417 22:22:01.029986 20947 Unigram_Model_Tester.cpp:51]
Initialized Word-Topic counts table
```

```
W0417 22:22:01.030045 20947 Unigram_Model_Tester.cpp:55]
Initializing Alpha vector from dumpfile ../ut_-
out/unigram/lda.par.dump
```

```
W0417 22:22:01.030118 20947 Unigram_Model_Tester.cpp:57]
Alpha vector initialized
```

```
W0417 22:22:01.030144 20947 Unigram_Model_Tester.cpp:60]
Initializing Beta Parameter from specified Beta = 0.01
```

```
W0417 22:22:01.030186 20947 Unigram_Model_Tester.cpp:63]
Beta param initialized
```

```
W0417 22:22:01.036779 20947 Testing_Execution_-
Strategy.cpp:24] Starting Parallel testing Pipeline
```

```
W0417 22:22:01.329864 20947 Testing_Execution_-
Strategy.cpp:36] Iteration 0 done. Took 0.00488273 mins
```

```
W0417 22:22:01.354588 20947 Unigram_Model.cpp:92] Average
num of topics assigned per word = 3.37222
```

```
W0417 22:22:01.354617 20947 Testing_Execution_-
Strategy.cpp:41] >>>>>>>>> Log-Likelihood (model,
doc, total): -1.93551e+06 , -57496.6 , -1.99301e+06
W0417 22:22:01.358059 20947 Testing_Execution_-
Strategy.cpp:49] Parallel testing Pipeline done
W0417 22:22:01.360312 20947 Unigram_Model.cpp:105] Saving
model for test pipeline in lda.ttc.dump and lda.par.dump
W0417 22:22:01.375068 20947 Unigram_Model.cpp:108] Model
saved
W0417 22:22:01.375098 20947 Unigram_Model.cpp:117]
Writing top words identified per topic into
lda.topToWor.txt
W0417 22:22:01.380234 20947 Unigram_Model.cpp:139] Word
statistics per topic written
W0417 22:22:01.380290 20947 Unigram_Model_Training_-
Builder.cpp:126] Saving document to topic-proportions
in lda.docToTop.txt
W0417 22:22:01.380311 20947 Unigram_Model_Training_-
Builder.cpp:127] Saving word to topic assignment in
lda.worToTop.txt
W0417 22:22:01.401484 20947 Unigram_Model_Training_-
Builder.cpp:193] Document to topic-proportions saved in
lda.docToTop.txt
W0417 22:22:01.401512 20947 Unigram_Model_Training_-
Builder.cpp:194] Word to topic assignment saved in
lda.worToTop.txt
```

Note the following flags:

1. test: This indicates the batch test mode
2. dumpprefix: This flag gives the prefix for the binary files storing the model in our case, `./ut_out/lda`. 'lda' is the default prefix used by the framework when none is specified. If you have specified a different prefix here is the place to use it. This will add different suffixes to fetch the various binary files to fetch the model saved and use them to infer the topic mixture for your new documents.
3. topics: This flag indicates the number of topics that your trained model contained. The same number should be used for your new documents.

```
$ ls -al lda*
```

```
-rw-r--r-- 1 shravanm shravanm 40059 2011-04-17 22:16
lda.dict.dump
```

```
-rw-r--r-- 1 shravanm shravanm 204655 2011-04-17 22:16
lda.dict.dump.global

-rw-r--r-- 1 shravanm shravanm 18947 2011-04-17 22:22
lda.docToTop.txt

-rw-r--r-- 1 shravanm shravanm 816 2011-04-17 22:22
lda.par.dump

-rw-r--r-- 1 shravanm shravanm 62380 2011-04-17 22:22
lda.top

-rw-r--r-- 1 shravanm shravanm 38389 2011-04-17 22:22
lda.topToWor.txt

-rw-r--r-- 1 shravanm shravanm 83799 2011-04-17 22:22
lda.ttc.dump

-rw-r--r-- 1 shravanm shravanm 36374 2011-04-17 22:16
lda.wor

-rw-r--r-- 1 shravanm shravanm 200546 2011-04-17 22:22
lda.worToTop.txt
```

This is pretty much the same output that you see when you learnt the model but for your new documents.

4.2.2 Streaming Mode

Again assuming that `PWD=$LDA_HOME/ut_test`

In this there is no need for formatting the documents but you need to provide the dictionary as an additional parameter. You also need to specify the maximum memory in MBs that you can allocate to store the model. You can tokenize and directly pipe the raw text documents through 'learntopics'.

```
$ java Tokenizer | ../learntopics -teststream
--dumpprefix=../ut_out/unigram/lda --topics=100
--dictionary=../ut_out/unigram/lda.dict.dump
--maxmem=128
```

```
W0417 23:06:24.342099 21004 Controller.cpp:68]
```

```
-----
W0417 23:06:24.342605 21004 Controller.cpp:69] Log
files are being stored at /home/shravanm/workspace/LDA_
Refactored/ut_test/learntopics.*
```

```
W0417 23:06:24.342627 21004 Controller.cpp:70]
```

```
-----
W0417 23:06:24.343189 21004 Controller.cpp:92] You have
```

chosen single machine testing mode

W0417 23:06:24.343723 21004 [Unigram_Model_Streaming_Builder.cpp](#):20] Initializing global dictionary from
../ut_out/unigram/lda.dict.dump

W0417 23:06:24.398543 21004 [Unigram_Model_Streaming_Builder.cpp](#):23] Dictionary initialized and has 17208

W0417 23:06:24.398643 21004 [Unigram_Model_Streaming_Builder.cpp](#):49] Estimating the words that will fit in 128 MB

W0417 23:06:24.486726 21004 [Unigram_Model_Streaming_Builder.cpp](#):52] 17208 will fit in 1.05881 MB of memory

W0417 23:06:24.486809 21004 [Unigram_Model_Streaming_Builder.cpp](#):53] Initializing Local Dictionary from
../ut_out/unigram/lda.dict.dump with 17208 words.

W0417 23:06:24.562101 21004 [Unigram_Model_Streaming_Builder.cpp](#):82] Local Dictionary Initialized. Size:
34416

W0417 23:06:24.565176 21004 [Unigram_Model_Streamer.cpp](#):27] Initializing Word-Topic counts table
from dump ../ut_out/unigram/lda.ttc.dump using 17208 words & 100 topics.

W0417 23:06:24.608408 21004 [Unigram_Model_Streamer.cpp](#):45] Initialized Word-Topic counts table

W0417 23:06:24.608469 21004 [Unigram_Model_Streamer.cpp](#):49] Initializing Alpha vector from dumpfile
../ut_out/unigram/lda.par.dump

W0417 23:06:24.608543 21004 [Unigram_Model_Streamer.cpp](#):51] Alpha vector initialized

W0417 23:06:24.608571 21004 [Unigram_Model_Streamer.cpp](#):54] Initializing Beta [Parameter](#) from
specified Beta = 0.01

W0417 23:06:24.608608 21004 [Unigram_Model_Streamer.cpp](#):57] Beta param initialized

W0417 23:06:24.615538 21004 [Testing_Execution_Strategy.cpp](#):24] Starting Parallel testing [Pipeline](#)

www.sauritchsurfboards.com/ recreation/sports/aquatic_sports watch out jeremy sherwin is here over the past six months you may have noticed this guy in every surf magazine published jeremy is finally getting his run

more.. copyright surfboards 2004 all rights reserved
 june 6 2004 new launches it s new and improved site you
 can now order custom surfboards online more improvements
 to come.. top selling models middot rocket fish middot
 speed egg middot classic middot squash

www.sauritchsurfboards.com/ recreation/sports/aquatic_
 sports (watch,83) (past,86) (months,77) (noticed,15)
 (guy,93) (surf,35) (magazine,86) (published,92)
 (finally,49) (run,21) (copyright,62) (surfboards,27)
 (rights,90) (reserved,59) (june,63) (launches,26)
 (improved,40) (site,26) (order,72) (custom,36)
 (surfboards,11) (online,68) (improvements,67) (top,29)
 (selling,82) (models,30) (middot,62) (rocket,23)
 (fish,67) (middot,35) (speed,29) (egg,2) (middot,22)
 (classic,58) (middot,69) (squash,67)

www.semente.pt recreation/sports/aquatic_sports por
 desde de 1999 para este site com o para browsers 4 ou
 superior de prefer ecirc ncia o internet explorer aqui
 gr aacute tis este site foi e pela think pink multimedia
 2000 surfboards all rights reserved

www.semente.pt recreation/sports/aquatic_sports (por,83)
 (de,86) (para,77) (site,15) (para,93) (browsers,35)
 (superior,86) (de,92) (prefer,49) (internet,21)
 (explorer,62) (aqui,27) (gr,90) (aacute,59) (site,63)
 (pink,26) (multimedia,40) (surfboards,26) (rights,72)
 (reserved,36)

W0417 23:06:38.769309 21004 [Testing_Execution_-
Strategy.cpp](#):36] Iteration 0 done. Took 0.235894 mins

W0417 23:06:38.875191 21004 [Unigram_Model.cpp](#):92] Average
 num of topics assigned per word = 1.85268

W0417 23:06:38.875252 21004 [Testing_Execution_-
Strategy.cpp](#):41] >>>>>>>> Log-Likelihood (model,
 doc, total): -2.52084e+06 , -194.288 , -2.52103e+06

W0417 23:06:38.875350 21004 [Testing_Execution_-
Strategy.cpp](#):49] Parallel testing [Pipeline](#) done

Note the flags used:

1. teststream: Indicates the streaming test mode
2. dictionary: The dictionary of words that the model knows about. Only those words in the test data set are recognized and the rest are ignored.
3. maxmemory: In MB. Denotes the amount of memory you allocate to store the

model.

The rest are same as batch mode.

Here there is no other output that you need to look for. The topic assignments are dumped back to stdout. The word mixtures for the topics are the same as that in the model.

4.3 Output Generated

1. `lda.wor`: This file generated by 'formatter' is the document in the protobuf format with words replaced by their indices
2. `lda.top`: This file generated by 'learntopics' contains the current topic assignments for all the documents in the protobuf format
3. `lda.dict.dump`: This file is a binary dump of the dictionary generated by the 'formatter' program. This will be implicitly used by the 'learntopics' program
4. `lda.ttc.dump`: This file generated by 'learntopics' is the binary dump of the word-topic counts table. This is the state at the end of all the iterations and essentially represents all the training that has happened through all the iterations. It is very important as this is a necessary input to the test pipeline.
5. `lda.par.dump`: This file generated by 'learntopics' is the binary dump of the parameters specified by the user which might be modified by an optimization step
6. `lda.chk`: This file generated by 'learntopics' is a checkpoint keeping the current iteration and some metadata
7. `lda*.txt`: These are for human consumption and there are three of these generated by 'learntopics':
 - (a) `lda.topToWor.txt` – The word mixtures for each topics
 - (b) `lda.worToTop.txt` – The topic assignments for every document on a per word basis
 - (c) `lda.docToTop.txt` – The topic proportions for every document

4.4 Customization

In its simplest form, all the parameters to 'formatter' and 'learntopics' have sensible default values which practically allows them to be used without any arguments. However, if you need to customize your setup then the following options are available:

- **I/O Customization:** By default all files output by the formatter, which are also the input to the learnTopcis program and all files output by the learntopics program use the prefix **lda**. This can be easily customized using the **outputprefix** & **inputprefix** flags respectively. But while this customization is used, one needs to be careful about keeping the **outputprefix** & **inputprefix** same.
- **Parameter Customization:** The weights of the Dirichlet Conjugates for both topics(alpha) & words(beta) can be changed by the **alpha** & **beta** flags.
- **Diagnostics & Optimization:** By default, the code performs alpha optimization every 25 iterations after burn in and also prints the log likelihood every 25 iterations. These are customizable using **optimizestats**, **printloglikelihood** & **burnin** flags.
- **Initialization Options:** By default, we do random initialization for the topic assignments in the first iteration. However, we can be a bit smarter about this. Instead of random assignments, we start out with no topic assignments and an empty word-topic counts tables. The sampling depends entirely on the smoothing mass for the first few documents and subsequent documents use the counts table built so far. This is similar in style to sequential monte carlo. You can use the `\online\` flag to signal this. We have seen that this leads to faster convergence.

Chapter 5

Using Y!LDA

The main purpose of the Y!LDA framework is to allow you to infer the set of constituents that can be used to represent your documents as mixtures of the inferred constituents. This is also called learning a model for your corpus. This takes a significant amount of time.

Once a model has been learnt, you can also use it to find out what topics a new document is made up by using LDA in test mode. In this mode, you provide the learnt model as a parameter and use that model to infer the topic mixture for new documents. This is very fast compared to learning the model. This can be done in two ways:

1. Batch mode: If you have a bunch of new documents for which you want to infer the topic mixture you use this mode.
2. Streaming mode: If you have documents which are to be streamed through the binary generating the topic mixture instantly, then this mode is for you.

The main difference between the two modes is that, in batch mode, the model is loaded for every batch of documents whereas in streaming mode, the model is loaded once and you can keep streaming the documents. In batch mode, you wait till all the documents in the batch are assigned topic mixtures whereas in streaming mode the topic mixture is assigned on a per document basis. In streaming mode, only a part of the model is loaded depending on how much memory you allocate for storing the model. However, we ensure that the model for the most important words (words that occur more frequently) are loaded before loading the model for lesser frequent words till the allocated memory is used up. The model loading time can be significant if it's big. Hence the case for a streaming mode for those of you who do not want to bear the cost of loading a big model every time you want to infer topic mixtures on new documents.

Each of these is described in detail with an illustration below in the next subsection.

The other dimension to consider is the size of the data. If your corpus has a couple of million documents and you are willing to wait for a couple of days then all you need to know is the single machine setup where all the data is on a single machine (usually multi-core) and you run LDA on that machine.

If your corpus is larger than that, you have no choice but to resort to the multi-machine setup. Here you split your data amongst multiple machines (possibly each machine being a multi-core one) and run LDA on all those machines. The Y!LDA framework currently provides a hadoop based distributed LDA mechanism that uses a custom implemented distributed hash table. The framework provides hadoop-streaming scripts that do the entire thing (splitting, training, testing) in a distributed fashion on the hadoop cluster backed by the HDFS as the distributed file system where your data is stored. If you do not have a hadoop cluster at your disposal, we currently do not provide any scripts to run things automatically. You will have to distribute the data manually, copy the binary version, if it's a homogenous setup, else install and run the framework on all the nodes containing data and finally execute binaries to merge the distributed parts of the model. We will only detail the steps to achieve them. Perhaps, one can contribute some scripts back so that this can be done automatically!

Single Machine Setup:

[Single Machine Usage](#)

Multi-Machine Setup:

[Multi Machine Usage](#)

Chapter 6

Todo List

Page [Y!LDA Architecture](#) Add unit tests to make the code more robust

Add more code documentation for the [Unigram_Model](#) components

Implement fancier models in later versions

Implement extensions to the LDA model in later versions

Chapter 7

Namespace Index

7.1 Namespace List

Here is a list of all namespaces with brief descriptions:

LDAUtil	55
sampler	56

Chapter 8

Class Index

8.1 Class Hierarchy

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

Checkpoint	57
Local_Checkpoint	92
Hadoop_Checkpoint	85
Client	59
DM_Client	66
Context	61
Cookie	63
Data_Formatter	64
Unigram_Train_Data_Formatter	168
Unigram_Model_Stream	145
Unigram_Test_Data_Formatter	167
GlobalTable::DistributedMap	
DM_Server	69
LDAUtil::DM_Server_Names	72
DocumentReader	73
DocumentWriter	74
Execution_Strategy	75
Testing_Execution_Strategy	125
Training_Execution_Strategy	134
Synchronized_Training_Execution_Strategy	115
Filter_Eval	76
Filter_Optimizer	77
Filter_Reader	78
Filter_Sampler	79

Filter_Tester	80
Filter_Updater	81
Filter_Writer	82
GenericTopKList< T, GreaterThan >	83
HashMap_Array< Key, Value >	86
InvalidOldTopicExc	88
HashMap_Array< Key, Value >::iterator	89
LDAUtil::Itoa	91
Model	94
Unigram_Model	142
Model_Builder	96
Unigram_Model_Streaming_Builder	147
Unigram_Model_Training_Builder	164
Unigram_Model_Synchronized_Training_Builder	150
Unigram_Model_Testing_Builder	158
Model_Director	98
Model_Refiner	99
Unigram_Model_Tester	154
Unigram_Model_Streammer	145
Unigram_Model_Trainer	160
Parameter	102
Pipeline	104
TBB_Pipeline	121
PNGCallback	107
PNGJob	109
Server_Helper	111
Unigram_Model_Server_Helper	144
simple_map	112
LDAUtil::StringTokenizer	113
LDAUtil::StringTrimmer	114
Synchronizer	117
Synchronizer_Helper	119
Unigram_Model_Synchronizer_Helper	152
TopicCounts	126
TopKList	132
TypeTopicCounts	136
WordIndexDictionary	171

Chapter 9

Class Index

9.1 Class List

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

Checkpointner (Used to implement failure recovery)	57
Client	59
Context (An object that maintains the context for the executing code)	61
Cookie	63
Data_Formatter (An interface for formatter objects)	64
DM_Client (The client used to access the Distributed Map)	66
DM_Server (The Server class that implements the DistributedMap Ice inter- face)	69
LDAUtil::DM_Server_Names	72
DocumentReader	73
DocumentWriter	74
Execution_Strategy (Interface for strategy objects)	75
Filter_Eval (A filter in the TBB pipeline)	76
Filter_Optimizer (A filter in the TBB pipeline)	77
Filter_Reader (A filter in the TBB pipeline)	78
Filter_Sampler (A filter in the TBB pipeline)	79
Filter_Tester (A filter in the TBB pipeline)	80
Filter_Updater	81
Filter_Writer (A filter in the TBB pipeline)	82
GenericTopKList< T, GreaterThan > (A list that maintains top K elements) .	83
Hadoop_Checkpointer	85
Hashmap_Array< Key, Value >	86
InvalidOldTopicExc	88
Hashmap_Array< Key, Value >::iterator	89
LDAUtil::Itoa	91

Local_Checkpointer	92
Model	94
Model_Builder	96
Model_Director (The Director class of the Builder pattern)	98
Model_Refiner	99
Parameter	102
Pipeline (An interface that all pipeline objects must implement)	104
PNGCallback (The call back object for the Put and Get AMI)	107
PNGJob	109
Server_Helper (Server Helper interface)	111
simple_map	112
LDAUtil::StringTokenizer	113
LDAUtil::StringTrimmer	114
Synchronized_Training_Execution_Strategy (Default implementation of the Execution_Strategy interface)	115
Synchronizer	117
Synchronizer_Helper (A helper class for the synchronizer)	119
TBB_Pipeline	121
Testing_Execution_Strategy	125
TopicCounts	126
TopKList	132
Training_Execution_Strategy	134
TypeTopicCounts	136
Unigram_Model	142
Unigram_Model_Server_Helper	144
Unigram_Model_Streammer	145
Unigram_Model_Streaming_Builder	147
Unigram_Model_Synchronized_Training_Builder	150
Unigram_Model_Synchronizer_Helper	152
Unigram_Model_Tester	154
Unigram_Model_Testing_Builder	158
Unigram_Model_Trainer	160
Unigram_Model_Training_Builder	164
Unigram_Test_Data_Formatter	167
Unigram_Train_Data_Formatter	168
WordIndexDictionary (A two way dictionary of words to indices)	171

Chapter 10

File Index

10.1 File List

Here is a list of all files with brief descriptions:

src/architecture.h	175
src/mainpage.h	250
src/multi_mcahine_usage.h	251
src/single_machine_usage.h	252
src/usage.h	294
src/commons/Client.h	176
src/commons/comparator.cpp	177
src/commons/comparator.h	178
src/commons/constants.h	179
src/commons/Context.cpp	180
src/commons/Context.h	181
src/commons/document.pb.cc	182
src/commons/document.pb.h	183
src/commons/DocumentReader.cpp	184
src/commons/DocumentReader.h	185
src/commons/DocumentWriter.cpp	186
src/commons/DocumentWriter.h	187
src/commons/LDAUtil.cpp	192
src/commons/LDAUtil.h	193
src/commons/types.h	246
src/commons/WordIndexDictionary.cpp	248
src/commons/WordIndexDictionary.h	249
src/commons/Formatter/Controller.cpp	188
src/commons/Formatter/Data_Formatter.h	190
src/commons/Formatter/FormatData_flags_define.h	191

src/commons/Server/DistributedMap.cpp	194
src/commons/Server/DistributedMap.h	195
src/commons/Server/DM_Server.cpp	196
src/commons/Server/DM_Server.h	198
src/commons/Server/Hashmap_Array.h	200
src/commons/Server/Server_Helper.h	201
src/commons/TopicLearner/Checkpointier.h	202
src/commons/TopicLearner/Controller.cpp	189
src/commons/TopicLearner/Dirichlet.cpp	203
src/commons/TopicLearner/Dirichlet.h	204
src/commons/TopicLearner/DM_Client.cpp	205
src/commons/TopicLearner/DM_Client.h	206
src/commons/TopicLearner/Execution_Strategy.h	207
src/commons/TopicLearner/Filter_Eval.cpp	208
src/commons/TopicLearner/Filter_Eval.h	209
src/commons/TopicLearner/Filter_Optimizer.cpp	210
src/commons/TopicLearner/Filter_Optimizer.h	211
src/commons/TopicLearner/Filter_Reader.cpp	212
src/commons/TopicLearner/Filter_Reader.h	213
src/commons/TopicLearner/Filter_Sampler.cpp	214
src/commons/TopicLearner/Filter_Sampler.h	215
src/commons/TopicLearner/Filter_Tester.cpp	216
src/commons/TopicLearner/Filter_Tester.h	217
src/commons/TopicLearner/Filter_Updater.cpp	218
src/commons/TopicLearner/Filter_Updater.h	219
src/commons/TopicLearner/Filter_Writer.cpp	220
src/commons/TopicLearner/Filter_Writer.h	221
src/commons/TopicLearner/GenericTopKList.h	222
src/commons/TopicLearner/Main_flags_define.h	223
src/commons/TopicLearner/Model.h	227
src/commons/TopicLearner/Model_Builder.h	228
src/commons/TopicLearner/Model_Director.cpp	229
src/commons/TopicLearner/Model_Director.h	230
src/commons/TopicLearner/Model_Refiner.h	231
src/commons/TopicLearner/Parameter.cpp	232
src/commons/TopicLearner/Parameter.h	233
src/commons/TopicLearner/Pipeline.h	234
src/commons/TopicLearner/Synchronized_Training_Execution_	
Strategy.cpp	235
src/commons/TopicLearner/Synchronized_Training_Execution_Strategy.h	236
src/commons/TopicLearner/Synchronizer.cpp	237
src/commons/TopicLearner/Synchronizer.h	238
src/commons/TopicLearner/Synchronizer_Helper.h	239
src/commons/TopicLearner/TBB_Pipeline.cpp	240
src/commons/TopicLearner/TBB_Pipeline.h	241
src/commons/TopicLearner/Testing_Execution_Strategy.cpp	242

src/commons/TopicLearner/Testing_Execution_Strategy.h	243
src/commons/TopicLearner/Training_Execution_Strategy.cpp	244
src/commons/TopicLearner/Training_Execution_Strategy.h	245
src/Unigram_Model/Formatter/Unigram_Test_Data_Formatter.cpp	253
src/Unigram_Model/Formatter/Unigram_Test_Data_Formatter.h	254
src/Unigram_Model/Formatter/Unigram_Train_Data_Formatter.cpp	255
src/Unigram_Model/Formatter/Unigram_Train_Data_Formatter.h	256
src/Unigram_Model/Merge/Merge_Dictionaries.cpp	257
src/Unigram_Model/Merge/Merge_Topic_Counts.cpp	258
src/Unigram_Model/Server/Unigram_Model_Server_Helper.cpp	260
src/Unigram_Model/Server/Unigram_Model_Server_Helper.h	261
src/Unigram_Model/TopicLearner/eff_small_map.cpp	262
src/Unigram_Model/TopicLearner/eff_small_map.h	263
src/Unigram_Model/TopicLearner/Hadoop_Checkpointer.cpp	264
src/Unigram_Model/TopicLearner/Hadoop_Checkpointer.h	265
src/Unigram_Model/TopicLearner/Local_Checkpointer.cpp	266
src/Unigram_Model/TopicLearner/Local_Checkpointer.h	267
src/Unigram_Model/TopicLearner/sampler.cpp	268
src/Unigram_Model/TopicLearner/sampler.h	269
src/Unigram_Model/TopicLearner/TopicCounts.cpp	270
src/Unigram_Model/TopicLearner/TopicCounts.h	271
src/Unigram_Model/TopicLearner/TopKList.cpp	272
src/Unigram_Model/TopicLearner/TopKList.h	273
src/Unigram_Model/TopicLearner/TypeTopicCounts.cpp	274
src/Unigram_Model/TopicLearner/TypeTopicCounts.h	275
src/Unigram_Model/TopicLearner/Unigram_Model.cpp	276
src/Unigram_Model/TopicLearner/Unigram_Model.h	277
src/Unigram_Model/TopicLearner/Unigram_Model_Stream.cpp	278
src/Unigram_Model/TopicLearner/Unigram_Model_Stream.h	279
src/Unigram_Model/TopicLearner/Unigram_Model_Streaming_Builder.cpp	280
src/Unigram_Model/TopicLearner/Unigram_Model_Streaming_Builder.h	281
src/Unigram_Model/TopicLearner/Unigram_Model_Synchronized_ Training_Builder.cpp	282
src/Unigram_Model/TopicLearner/Unigram_Model_Synchronized_ Training_Builder.h	283
src/Unigram_Model/TopicLearner/Unigram_Model_Synchronizer_ Helper.cpp	284
src/Unigram_Model/TopicLearner/Unigram_Model_Synchronizer_Helper.h	285
src/Unigram_Model/TopicLearner/Unigram_Model_Tester.cpp	286
src/Unigram_Model/TopicLearner/Unigram_Model_Tester.h	287
src/Unigram_Model/TopicLearner/Unigram_Model_Testing_Builder.cpp	288
src/Unigram_Model/TopicLearner/Unigram_Model_Testing_Builder.h	289
src/Unigram_Model/TopicLearner/Unigram_Model_Trainer.cpp	290
src/Unigram_Model/TopicLearner/Unigram_Model_Trainer.h	291
src/Unigram_Model/TopicLearner/Unigram_Model_Training_Builder.cpp	292
src/Unigram_Model/TopicLearner/Unigram_Model_Training_Builder.h	293

Chapter 11

Namespace Documentation

11.1 LDAUtil Namespace Reference

Classes

- class [StringTokenizer](#)
- class [StringTrimmer](#)
- class [Itoa](#)
- class [DM_Server_Names](#)

11.2 sampler Namespace Reference

Functions

- `topic_t sample (const topicCounts *currentTypeTopicCounts, topic_t old_topic, const atomic< topic_t > *tokens_per_topic, const topic_t *local_topic_counts, const topic_t *local_topic_index, const int &non_zero_topics, const double &smoothingOnlyMass, const double &topicBetaMass, const double *cachedCoefficients, const double &betaSum, const double *alpha, double *topic_term_scores, const topic_t &numTopics, variate_generator< base_generator_type &, boost::uniform_real<> > *unif01)`

11.2.1 Function Documentation

- 11.2.1.1** `topic_t sampler::sample (const topicCounts * currentTypeTopicCounts, topic_t old_topic, const atomic< topic_t > * tokens_per_topic, const topic_t * local_topic_counts, const topic_t * local_topic_index, const int & non_zero_topics, const double & smoothingOnlyMass, const double & topicBetaMass, const double * cachedCoefficients, const double & betaSum, const double * alpha, double * topic_term_scores, const topic_t & numTopics, variate_generator< base_generator_type &, boost::uniform_real<> > * unif01)`

Chapter 12

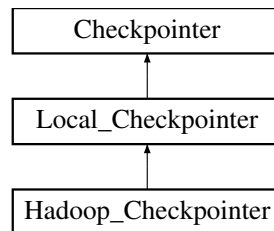
Class Documentation

12.1 Checkpointer Class Reference

Used to implement failure recovery.

```
#include <Checkpointer.h>
```

Inheritance diagram for Checkpointer:



Public Member Functions

- virtual void [save_metadata](#) (std::string &state)=0
Serialize the metadata.
- virtual std::string [load_metadata](#) ()=0
Load the metadata.
- virtual void [checkpoint](#) ()=0
Serialize other necessary data structures.

12.1.1 Detailed Description

Used to implement failure recovery. The checkpointer interface. A checkpointer object helps create a checkpoint and also to load from a checkpoint

A checkpointer has two things: 1. Metadata - (Ex.: The iteration at which this checkpoint was created) 2. The method to serialize the necessary data structures to disk

12.1.2 Member Function Documentation

12.1.2.1 `virtual void Checkpointer::checkpoint () [pure virtual]`

Serialize other necessary data structures.

Implemented in [Hadoop_Checkpointer](#), and [Local_Checkpointer](#).

12.1.2.2 `virtual std::string Checkpointer::load_metadata () [pure virtual]`

Load the metadata.

Implemented in [Local_Checkpointer](#).

12.1.2.3 `virtual void Checkpointer::save_metadata (std::string & state) [pure virtual]`

Serialize the metadata.

Implemented in [Local_Checkpointer](#).

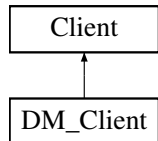
The documentation for this class was generated from the following file:

- [src/commons/TopicLearner/Checkpoint.h](#)

12.2 Client Class Reference

```
#include <Client.h>
```

Inheritance diagram for Client:



Public Member Functions

- virtual bool [get](#) (const string &key, string &val)=0
- virtual void [set](#) (const string &key, const string &val)=0
- virtual void [put](#) (const string &key, const string &delta)=0
- virtual void [begin_putNget](#) (const string &key, const string &val)=0
- virtual bool [remove](#) (const string &key, string &val)=0
- virtual void [wait_for_all](#) ()=0

Provides barrier functionality.

- virtual void [wait_till_done](#) ()=0

12.2.1 Detailed Description

The interface to be implemented by any client for the Distributed Map service

12.2.2 Member Function Documentation

12.2.2.1 virtual void Client::begin_putNget (const string &key, const string &val) [pure virtual]

Asynchronous put and get operation. It begins with a put. The call back in [Synchronizer_Helper](#) is called which creates the effect of a get

Implemented in [DM_Client](#).

12.2.2.2 virtual bool Client::get (const string &key, string &val) [pure virtual]

Gets the serialized val stored for the key in the Distributed Map. Returns true if key exists and false otherwise

Implemented in [DM_Client](#).

**12.2.2.3 virtual void Client::put (const string & *key*, const string & *delta*)
[pure virtual]**

Adds delta to the serialized value for the key in the Distributed Map. This has accumulator semantics. The accumulator logic is provided by the [Server_Helper](#).

Implemented in [DM_Client](#).

12.2.2.4 virtual bool Client::remove (const string & *key*, string & *val*) [pure virtual]

Remove the key from the Distributed Map returning the serialized value stored as val. Returns true if key exists and false otherwise

Implemented in [DM_Client](#).

12.2.2.5 virtual void Client::set (const string & *key*, const string & *val*) [pure virtual]

Sets val as the serialized value for the key in the Distributed Map. This has replace semantics

Implemented in [DM_Client](#).

12.2.2.6 virtual void Client::wait_for_all () [pure virtual]

Provides barrier functionality.

Implemented in [DM_Client](#).

12.2.2.7 virtual void Client::wait_till_done () [pure virtual]

Provides functionality to wait for any asynchronous communication to end

Implemented in [DM_Client](#).

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

- [src/commons/Client.h](#)

12.3 Context Class Reference

An object that maintains the context for the executing code.

```
#include <Context.h>
```

Public Member Functions

- int [get_int](#) (string key)
Get an integer valued property named key.
- string [get_string](#) (string key)
Get a string valued property named key.
- double [get_double](#) (string key)
Get a double valued property named key.
- bool [get_bool](#) (string key)
Get a bool valued property named key.
- void [put_string](#) (string key, string val)

Static Public Member Functions

- static [Context](#) & [get_instance](#) ()

12.3.1 Detailed Description

An object that maintains the context for the executing code. This is a Singleton object that maintains a context for the executing code. It contains a list of properties stored as a key-value map

All the flags defined are made available through this object. It can also be used as a mechanism for message passing. This reduces coupling between the code and gflags.

12.3.2 Member Function Documentation

12.3.2.1 bool Context::get_bool (string key)

Get a bool valued property named key.

12.3.2.2 double Context::get_double (string *key*)

Get a double valued property named key.

12.3.2.3 Context & Context::get_instance () [static]**12.3.2.4 int Context::get_int (string *key*)**

Get an integer valued property named key.

12.3.2.5 string Context::get_string (string *key*)

Get a string valued property named key.

12.3.2.6 void Context::put_string (string *key*, string *val*)

Put a string value for property key into the map

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

- src/commons/[Context.h](#)
- src/commons/[Context.cpp](#)

12.4 Cookie Struct Reference

```
#include <DM_Client.h>
```

Public Attributes

- string [entity](#)
- int [msg_id](#)

12.4.1 Member Data Documentation

12.4.1.1 string Cookie::entity

12.4.1.2 int Cookie::msg_id

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

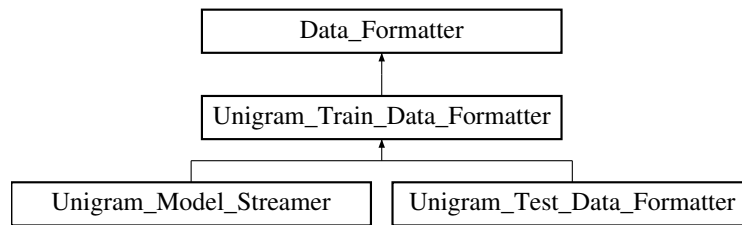
- [src/commons/TopicLearner/DM_Client.h](#)

12.5 Data_Formatter Class Reference

An interface for formatter objects.

```
#include <Data_Formatter.h>
```

Inheritance diagram for Data_Formatter:



Public Member Functions

- virtual void `format ()=0`
Perform the actual formatting.
- virtual `WordIndexDictionary & get_dictionary ()=0`
Return the dictionary being used by the formatter.
- virtual int `get_num_docs ()=0`
The number of documents formatted.
- virtual int `get_total_num_words ()=0`
The total number of words found.

12.5.1 Detailed Description

An interface for formatter objects. A formatter is an object that converts raw text corpus into binary so that its disk footprint is low and there is no parsing involved while reading it back

12.5.2 Member Function Documentation

12.5.2.1 virtual void Data_Formatter::format () [pure virtual]

Perform the actual formatting.

Implemented in [Unigram_Train_Data_Formatter](#).

12.5.2.2 virtual WordIndexDictionary& Data_Formatter::get_dictionary () [pure virtual]

Return the dictionary being used by the formatter.

Implemented in [Unigram_Train_Data_Formatter](#).

12.5.2.3 virtual int Data_Formatter::get_num_docs () [pure virtual]

The number of documents formatted.

Implemented in [Unigram_Train_Data_Formatter](#).

12.5.2.4 virtual int Data_Formatter::get_total_num_words () [pure virtual]

The total number of words found.

Implemented in [Unigram_Train_Data_Formatter](#).

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

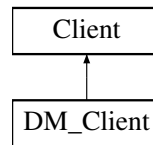
- [src/commons/Formatter/Data_Formatter.h](#)

12.6 DM_Client Class Reference

The client used to access the Distributed Map.

```
#include <DM_Client.h>
```

Inheritance diagram for DM_Client:



Public Member Functions

- [DM_Client](#) (int num_entities, const string &servers, [Synchronizer_Helper](#) &sync_helper)
- virtual [~DM_Client](#) ()
- void [put](#) (const string &entity, const string &delta)

The usual map operations.

- void [set](#) (const string &entity, const string &counts)
- bool [get](#) (const string &entity, string &counts)
- bool [remove](#) (const string &entity, string &counts)
- void [begin_putNget](#) (const string &entity, const string &delta)

Start the asynchronous putNget operation.

- void [wait_for_all](#) ()

Execute a barrier.

- void [wait_till_done](#) ()
- int [get_num_servers](#) ()

12.6.1 Detailed Description

The client used to access the Distributed Map. Wrapper class to delegate the Distributed Map operations to the appropriate Server

12.6.2 Constructor & Destructor Documentation

12.6.2.1 DM_Client::DM_Client (int *num_entities*, const string & *servers*, Synchronizer_Helper & *sync_helper*)

The servers in the Distributed Map setup The max number of entities that you are going to store in the map. This is just to make sure that things dont break when you are processing a very small number of entities particularly smaller than MAX_MSGS The helper class that provides the actual call back functionality

12.6.2.2 DM_Client::~~DM_Client () [virtual]

12.6.3 Member Function Documentation

12.6.3.1 void DM_Client::begin_putNget (const string & *entity*, const string & *delta*) [virtual]

Start the asynchronous putNget operation.

Implements [Client](#).

12.6.3.2 bool DM_Client::get (const string & *key*, string & *val*) [virtual]

Gets the serialized val stored for the key in the Distributed Map. Returns true if key exists and false otherwise

Implements [Client](#).

12.6.3.3 int DM_Client::get_num_servers ()

12.6.3.4 void DM_Client::put (const string & *entity*, const string & *delta*) [virtual]

The usual map operations.

Implements [Client](#).

12.6.3.5 bool DM_Client::remove (const string & *key*, string & *val*) [virtual]

Remove the key from the Distributed Map returning the serialized value stored as val. Returns true if key exists and false otherwise

Implements [Client](#).

**12.6.3.6 void DM_Client::set (const string & *key*, const string & *val*)
[virtual]**

Sets *val* as the serialized value for the *key* in the Distributed Map. This has replace semantics

Implements [Client](#).

12.6.3.7 void DM_Client::wait_for_all () [virtual]

Execute a barrier.

Implements [Client](#).

12.6.3.8 void DM_Client::wait_till_done () [virtual]

Provides functionality to wait for any asynchronous communication to end

Implements [Client](#).

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

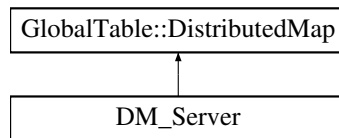
- [src/commons/TopicLearner/DM_Client.h](#)
- [src/commons/TopicLearner/DM_Client.cpp](#)

12.7 DM_Server Class Reference

The Server class that implements the DistributedMap Ice interface.

```
#include <DM_Server.h>
```

Inheritance diagram for DM_Server:



Public Member Functions

- [DM_Server](#) (int num_clients, [Server_Helper](#) &)
- virtual [~DM_Server](#) ()
- void [put](#) (const string &key, const string &val, const Ice::Current &)
- void [set](#) (const string &key, const string &val, const Ice::Current &)
- bool [get](#) (const string &key, string &val, const Ice::Current &)
- bool [remove](#) (const string &, string &, const Ice::Current &)
- void [putNget_async](#) (const AMD_DistributedMap_putNgetPtr &png_cb, const string &word, const string &delta, const Ice::Current &)
- void [waitForAllClients_async](#) (const AMD_DistributedMap_waitForAllClientsPtr &, const Ice::Current &)

Barrier. Helps clients synchronize.

12.7.1 Detailed Description

The Server class that implements the DistributedMap Ice interface. It essentially holds a part of the distributed map. Associates a 'key' to a particular 'value'

The implementation abstracts out the semantics. The only accepted key & value types are strings Other types of objects have to be serialized before they can be stored.

It provides the usual get, set & remove operations with the usual semantics.

It also provides accumulator semantics with the put operation. Values stored using a put operation are accumulated instead of being replaced. Note that the semantics of this accumulating is provided by a helper class that implements the [Server_Helper](#) interface

There is also a putNget operation which uses AMD to not block dispatch threads. The putNget is also an AMI. More info on AMI in client class. Please look up Ice

documentation for more details. This operation puts a value for the given key and returns the result of the accumulate operation.

It also provides a barrier implementation that lets clients synchronize. This is a very simple use of AMD The dispatch threads are released without busy waiting. Only a counter is incremented with every call. Once the counter reaches num_clients registered, a response is sent to all queued up call back objects.

12.7.2 Constructor & Destructor Documentation

12.7.2.1 `DM_Server::DM_Server (int num_clients, Server_Helper & server_helper)`

The num of clients that will be accessing the DistributedMap This parameter is used to provide a barrier

12.7.2.2 `DM_Server::~~DM_Server ()` **[virtual]**

12.7.3 Member Function Documentation

12.7.3.1 `bool DM_Server::get (const string & key, string & val, const Ice::Current &)`

The get operation. Sets val with the value of the key and returns true if the key exists. Else does not set val and returns false

12.7.3.2 `void DM_Server::put (const string & key, const string & val, const Ice::Current &)`

The put operation. Note the slight difference in semantics This has accumulator semantics. The value is accumulated and not replaced

12.7.3.3 `void DM_Server::putNget_async (const AMD_DistributedMap_-putNgetPtr & png_cb, const string & word, const string & delta, const Ice::Current &)`

The put and get operation. Does a put(word,delta) and returns the updated value png_-cb is the AMD call back object

12.7.3.4 bool DM_Server::remove (const string & *word*, string & *counts*, const Ice::Current &)

The remove operation. Removes the key-value pair from the table. If the key does not exist, returns false. Else returns true.

12.7.3.5 void DM_Server::set (const string & *key*, const string & *val*, const Ice::Current &)

The set operation. The value is replaced if the key already exists.

12.7.3.6 void DM_Server::waitForAllClients_async (const AMD_DistributedMap_waitForAllClientsPtr & *cb*, const Ice::Current &)

Barrier. Helps clients synchronize.

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

- [src/commons/Server/DM_Server.h](#)
- [src/commons/Server/DM_Server.cpp](#)

12.8 LDAUtil::DM_Server_Names Class Reference

```
#include <LDAUtil.h>
```

Static Public Member Functions

- static string [get_servant_name](#) (const int server_id)
- static string [get_server_endpoint](#) (const string &host_port)

12.8.1 Member Function Documentation

12.8.1.1 string LDAUtil::DM_Server_Names::get_servant_name (const int *server_id*) [**static**]

12.8.1.2 string LDAUtil::DM_Server_Names::get_server_endpoint (const string & *host_port*) [**static**]

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

- src/commons/[LDAUtil.h](#)
- src/commons/[LDAUtil.cpp](#)

12.9 DocumentReader Class Reference

```
#include <DocumentReader.h>
```

Public Member Functions

- [DocumentReader](#) (string w_fname_)
- virtual [~DocumentReader](#) ()
- int [read](#) (google::protobuf::Message *msg)

12.9.1 Detailed Description

Wrapper around protobuf messages for convenient reading of words, topics & (word,index) pairs from word, topic, dictionary dump files respectively. Assumes that each msg is in a binary file in record* format where record=(size of serialized msg,msg serialized as string)

12.9.2 Constructor & Destructor Documentation

12.9.2.1 DocumentReader::DocumentReader (string w_fname_)

Constructs a [DocumentReader](#) object to read msgs from w_fname and optionally topics from t_fname

12.9.2.2 DocumentReader::~~DocumentReader () [virtual]

12.9.3 Member Function Documentation

12.9.3.1 int DocumentReader::read (google::protobuf::Message * msg)

The default method to read a message from w_fname which is in (size of serialized msg,msg serialized as string)* format

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

- src/commons/[DocumentReader.h](#)
- src/commons/[DocumentReader.cpp](#)

12.10 DocumentWriter Class Reference

```
#include <DocumentWriter.h>
```

Public Member Functions

- [DocumentWriter](#) (string w_fname_)
- virtual [~DocumentWriter](#) ()
- bool [write](#) (const google::protobuf::Message &msg)

12.10.1 Detailed Description

Wrapper around protobuf msgs for convenient writing of words, topics & (word,index) pairs into word, topic, dictionary dump files respectively. Each msg is written into a binary file in record* format where record=(size of serialized msg,msg serialized as string)

12.10.2 Constructor & Destructor Documentation

12.10.2.1 DocumentWriter::DocumentWriter (string w_fname_)

Constructs a [DocumentWriter](#) object to write msgs into w_fname and optionally topics to t_fname. You can however choose to ignore the words file if you want to write only topics by setting w_fname_="" & using write_topics() method.

12.10.2.2 DocumentWriter::~~DocumentWriter () [virtual]

12.10.3 Member Function Documentation

12.10.3.1 bool DocumentWriter::write (const google::protobuf::Message &msg)

The default method to write a message to w_fname in (size of serialized msg,msg serialized as string)* format

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

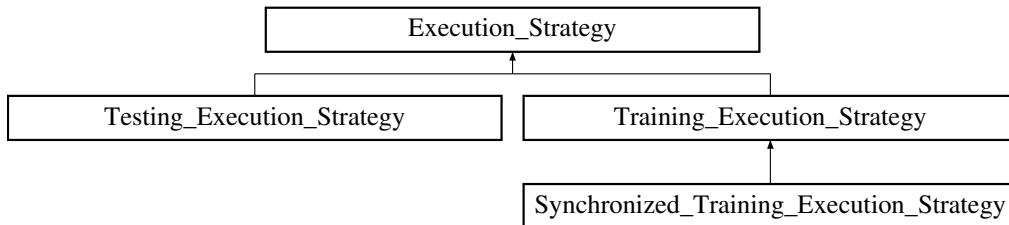
- src/commons/[DocumentWriter.h](#)
- src/commons/[DocumentWriter.cpp](#)

12.11 Execution_Strategy Class Reference

Interface for strategy objects.

```
#include <Execution_Strategy.h>
```

Inheritance diagram for Execution_Strategy:



Public Member Functions

- virtual void `execute` ()=0

12.11.1 Detailed Description

Interface for strategy objects. Implement this interface to define a strategy detailing what filters are added to the pipeline and how to run the pipeline

12.11.2 Member Function Documentation

12.11.2.1 virtual void Execution_Strategy::execute () [pure virtual]

Implemented in [Synchronized_Training_Execution_Strategy](#), [Testing_Execution_Strategy](#), and [Training_Execution_Strategy](#).

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

- `src/commons/TopicLearner/Execution_Strategy.h`

12.12 Filter_Eval Class Reference

A filter in the TBB pipeline.

```
#include <Filter_Eval.h>
```

Public Member Functions

- [Filter_Eval](#) ([Model_Refiner](#) &)
- virtual [~Filter_Eval](#) ()
- void * [operator\(\)](#) (void *)
- double [get_eval](#) ()

12.12.1 Detailed Description

A filter in the TBB pipeline. Delegates the task to be done to `refiner.eval`. The output from the eval step is accumulated and can be queried using [get_eval\(\)](#)

12.12.2 Constructor & Destructor Documentation

12.12.2.1 [Filter_Eval::Filter_Eval](#) ([Model_Refiner](#) & *refiner*)

12.12.2.2 [Filter_Eval::~~Filter_Eval](#) () [**virtual**]

12.12.3 Member Function Documentation

12.12.3.1 double [Filter_Eval::get_eval](#) ()

12.12.3.2 void * [Filter_Eval::operator\(\)](#) (void * *token*)

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

- [src/commons/TopicLearner/Filter_Eval.h](#)
- [src/commons/TopicLearner/Filter_Eval.cpp](#)

12.13 Filter_Optimizer Class Reference

A filter in the TBB pipeline.

```
#include <Filter_Optimizer.h>
```

Public Member Functions

- [Filter_Optimizer](#) ([Model_Refiner](#) &)
- virtual [~Filter_Optimizer](#) ()
- void * [operator\(\)](#) (void *)

12.13.1 Detailed Description

A filter in the TBB pipeline. Delegates the task to be done to `refiner.optimize()`

12.13.2 Constructor & Destructor Documentation

12.13.2.1 `Filter_Optimizer::Filter_Optimizer (Model_Refiner & refiner)`

12.13.2.2 `Filter_Optimizer::~~Filter_Optimizer ()` [**virtual**]

12.13.3 Member Function Documentation

12.13.3.1 `void * Filter_Optimizer::operator() (void * token)`

Each document is passed through the filter using this method If we have read in tau documents, we update the global alphas Else we compute gradients & accumulate them in the local alphas

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

- `src/commons/TopicLearner/Filter_Optimizer.h`
- `src/commons/TopicLearner/Filter_Optimizer.cpp`

12.14 Filter_Reader Class Reference

A filter in the TBB pipeline.

```
#include <Filter_Reader.h>
```

Public Member Functions

- [Filter_Reader](#) ([Model_Refiner](#) &)
- virtual [~Filter_Reader](#) ()
- void * [operator\(\)](#) (void *)

12.14.1 Detailed Description

A filter in the TBB pipeline. Delegates the task to be done to `refiner.read()`

12.14.2 Constructor & Destructor Documentation

12.14.2.1 `Filter_Reader::Filter_Reader (Model_Refiner & refiner)`

12.14.2.2 `Filter_Reader::~~Filter_Reader ()` [**virtual**]

12.14.3 Member Function Documentation

12.14.3.1 `void * Filter_Reader::operator() (void *)`

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

- [src/commons/TopicLearner/Filter_Reader.h](#)
- [src/commons/TopicLearner/Filter_Reader.cpp](#)

12.15 Filter_Sampler Class Reference

A filter in the TBB pipeline.

```
#include <Filter_Sampler.h>
```

Public Member Functions

- [Filter_Sampler](#) ([Model_Refiner](#) &)
- virtual [~Filter_Sampler](#) ()
- void * [operator\(\)](#) (void *)

12.15.1 Detailed Description

A filter in the TBB pipeline. Delegates the task to be done to [refiner.sample\(\)](#)

12.15.2 Constructor & Destructor Documentation

12.15.2.1 [Filter_Sampler::Filter_Sampler](#) ([Model_Refiner](#) & *refiner*)

12.15.2.2 [Filter_Sampler::~~Filter_Sampler](#) () [**virtual**]

12.15.3 Member Function Documentation

12.15.3.1 void * [Filter_Sampler::operator\(\)](#) (void * *token*)

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

- src/commons/TopicLearner/[Filter_Sampler.h](#)
- src/commons/TopicLearner/[Filter_Sampler.cpp](#)

12.16 Filter_Tester Class Reference

A filter in the TBB pipeline.

```
#include <Filter_Tester.h>
```

Public Member Functions

- [Filter_Tester](#) ([Model_Refiner](#) &)
- virtual [~Filter_Tester](#) ()
- void * [operator\(\)](#) (void *)

12.16.1 Detailed Description

A filter in the TBB pipeline. Delegates the task to be done to `refiner.test()`

12.16.2 Constructor & Destructor Documentation

12.16.2.1 `Filter_Tester::Filter_Tester (Model_Refiner & refiner)`

12.16.2.2 `Filter_Tester::~~Filter_Tester ()` [**virtual**]

12.16.3 Member Function Documentation

12.16.3.1 `void * Filter_Tester::operator() (void * token)`

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

- `src/commons/TopicLearner/Filter_Tester.h`
- `src/commons/TopicLearner/Filter_Tester.cpp`

12.17 Filter_Updater Class Reference

```
#include <Filter_Updater.h>
```

Public Member Functions

- [Filter_Updater](#) ([Model_Refiner](#) &)
- virtual [~Filter_Updater](#) ()
- void * [operator\(\)](#) (void *)

12.17.1 Constructor & Destructor Documentation

12.17.1.1 [Filter_Updater::Filter_Updater](#) ([Model_Refiner](#) & *refiner*)

12.17.1.2 [Filter_Updater::~~Filter_Updater](#) () [**virtual**]

12.17.2 Member Function Documentation

12.17.2.1 void * [Filter_Updater::operator\(\)](#) (void * *token*)

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

- [src/commons/TopicLearner/Filter_Updater.h](#)
- [src/commons/TopicLearner/Filter_Updater.cpp](#)

12.18 Filter_Writer Class Reference

A filter in the TBB pipeline.

```
#include <Filter_Writer.h>
```

Public Member Functions

- [Filter_Writer](#) ([Model_Refiner](#) &)
- virtual [~Filter_Writer](#) ()
- void * [operator\(\)](#) (void *)

12.18.1 Detailed Description

A filter in the TBB pipeline. Delegates the task to be done to `refiner.write()`

12.18.2 Constructor & Destructor Documentation

12.18.2.1 `Filter_Writer::Filter_Writer (Model_Refiner & refiner)`

12.18.2.2 `Filter_Writer::~~Filter_Writer ()` [**virtual**]

12.18.3 Member Function Documentation

12.18.3.1 `void * Filter_Writer::operator() (void * token)`

We receive the document to write to disk. This contains both words & their topic assignments. We use `write_topics()` to only write the topics which internally clears the words.

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

- `src/commons/TopicLearner/Filter_Writer.h`
- `src/commons/TopicLearner/Filter_Writer.cpp`

12.19 GenericTopKList< T, GreaterThan > Class Template Reference

A list that maintains top K elements.

```
#include <GenericTopKList.h>
```

Public Member Functions

- [GenericTopKList](#) (size_t K_=10)
- virtual [~GenericTopKList](#) ()
- void [push](#) (T elem)
- T [top](#) ()
- void [pop](#) ()
- bool [empty](#) ()
- void [print](#) (ostream &out)
- void [clear](#) ()

12.19.1 Detailed Description

```
template<class T, class GreaterThan> class GenericTopKList< T, GreaterThan  
>
```

A list that maintains top K elements.

12.19.2 Constructor & Destructor Documentation

12.19.2.1 `template<class T , class GreaterThan > GenericTopKList< T, GreaterThan >::GenericTopKList (size_t K_ = 10) [inline]`

12.19.2.2 `template<class T , class GreaterThan > virtual GenericTopKList< T, GreaterThan >::~~GenericTopKList () [inline, virtual]`

12.19.3 Member Function Documentation

12.19.3.1 `template<class T , class GreaterThan > void GenericTopKList< T, GreaterThan >::clear () [inline]`

12.19.3.2 `template<class T , class GreaterThan > bool GenericTopKList< T, GreaterThan >::empty () [inline]`

12.19.3.3 `template<class T , class GreaterThan > void GenericTopKList< T, GreaterThan >::pop () [inline]`

12.19.3.4 `template<class T , class GreaterThan > void GenericTopKList< T, GreaterThan >::print (ostream & out) [inline]`

12.19.3.5 `template<class T , class GreaterThan > void GenericTopKList< T, GreaterThan >::push (T elem) [inline]`

12.19.3.6 `template<class T , class GreaterThan > T GenericTopKList< T, GreaterThan >::top () [inline]`

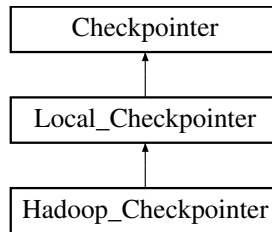
The documentation for this class was generated from the following file:

- `src/commons/TopicLearner/`[GenericTopKList.h](#)

12.20 Hadoop_Checkpointer Class Reference

```
#include <Hadoop_Checkpointer.h>
```

Inheritance diagram for Hadoop_Checkpointer:



Public Member Functions

- [Hadoop_Checkpointer \(\)](#)
- virtual [~Hadoop_Checkpointer \(\)](#)
- void [checkpoint \(\)](#)

Serialize other necessary data structures.

12.20.1 Constructor & Destructor Documentation

12.20.1.1 [Hadoop_Checkpointer::Hadoop_Checkpointer \(\)](#)

12.20.1.2 [Hadoop_Checkpointer::~~Hadoop_Checkpointer \(\)](#) **[virtual]**

12.20.2 Member Function Documentation

12.20.2.1 [void Hadoop_Checkpointer::checkpoint \(\)](#) **[virtual]**

Serialize other necessary data structures.

Reimplemented from [Local_Checkpointer](#).

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

- [src/Unigram_Model/TopicLearner/Hadoop_Checkpointer.h](#)
- [src/Unigram_Model/TopicLearner/Hadoop_Checkpointer.cpp](#)

12.21 Hashmap_Array< Key, Value > Class Template Reference

```
#include <Hashmap_Array.h>
```

Classes

- class [iterator](#)

Public Types

- typedef unordered_map< Key, Value > [act_map](#)
- typedef unordered_map< Key, Value >::[iterator](#) [act_map_iter](#)

Public Member Functions

- [Hashmap_Array](#) ()
- virtual [~Hashmap_Array](#) ()
- size_t [size](#) () const
- size_t [count](#) (Key key) const
- Value & [operator](#)[] (Key key)
- void [erase](#) (Key key)
- word_mutex_t * [get_lock](#) (Key key)
- word_mutex_t * [get_structure_lock](#) ()
- [iterator](#) [begin](#) ()
- [iterator](#) [end](#) ()

12.21.1 Detailed Description

template<class Key, class Value> class Hashmap_Array< Key, Value >

An array of hash maps. Reduces Contention on multi-threaded access.

12.21.2 Member Typedef Documentation

12.21.2.1 `template<class Key, class Value> typedef unordered_map<Key, Value> Hashmap_Array< Key, Value >::act_map`

12.21.2.2 `template<class Key, class Value> typedef unordered_map<Key, Value>::iterator Hashmap_Array< Key, Value >::act_map_iter`

12.21.3 Constructor & Destructor Documentation

12.21.3.1 `template<class Key, class Value> Hashmap_Array< Key, Value >::Hashmap_Array () [inline]`

12.21.3.2 `template<class Key, class Value> virtual Hashmap_Array< Key, Value >::~~Hashmap_Array () [inline, virtual]`

12.21.4 Member Function Documentation

12.21.4.1 `template<class Key, class Value> iterator Hashmap_Array< Key, Value >::begin () [inline]`

12.21.4.2 `template<class Key, class Value> size_t Hashmap_Array< Key, Value >::count (Key key) const [inline]`

12.21.4.3 `template<class Key, class Value> iterator Hashmap_Array< Key, Value >::end () [inline]`

12.21.4.4 `template<class Key, class Value> void Hashmap_Array< Key, Value >::erase (Key key) [inline]`

12.21.4.5 `template<class Key, class Value> word_mutex_t* Hashmap_Array< Key, Value >::get_lock (Key key) [inline]`

12.21.4.6 `template<class Key, class Value> word_mutex_t* Hashmap_Array< Key, Value >::get_structure_lock () [inline]`

12.21.4.7 `template<class Key, class Value> Value& Hashmap_Array< Key, Value >::operator[] (Key key) [inline]`

12.21.4.8 `template<class Key, class Value> size_t Hashmap_Array< Key, Value >::size () const [inline]`

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

- `src/commons/Server/Hashmap_Array.h`

12.22 InvalidOldTopicExc Class Reference

```
#include <types.h>
```

Public Member Functions

- [InvalidOldTopicExc](#) (word_t word_, topic_t old_topic_)
- virtual const char * [what](#) () const throw ()

12.22.1 Detailed Description

Exception thrown when an old topic is not found This is fatal and causes the program to stop

12.22.2 Constructor & Destructor Documentation

12.22.2.1 `InvalidOldTopicExc::InvalidOldTopicExc (word_t word_, topic_t old_topic_) [inline]`

12.22.3 Member Function Documentation

12.22.3.1 `virtual const char* InvalidOldTopicExc::what () const throw () [inline, virtual]`

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

- [src/commons/types.h](#)

12.23 Hashmap_Array< Key, Value >::iterator Class Reference

```
#include <Hashmap_Array.h>
```

Public Member Functions

- [iterator](#) (int ind, [act_map_iter](#) iter, [act_map_iter](#) end, [act_map](#) *bc)
- void [operator++](#) (int)
- bool [operator!=](#) ([iterator](#) &inp)

Public Attributes

- [act_map_iter](#) [current_iter](#)

12.23.1 Detailed Description

template<class Key, class Value> class Hashmap_Array< Key, Value >::iterator

The iterator is not thread safe.

12.23.2 Constructor & Destructor Documentation

12.23.2.1 **template<class Key, class Value> Hashmap_Array< Key, Value >::iterator::iterator** (int *ind*, [act_map_iter](#) *iter*, [act_map_iter](#) *end*, [act_map](#) * *bc*) [**inline**]

12.23.3 Member Function Documentation

12.23.3.1 **template<class Key, class Value> bool Hashmap_Array< Key, Value >::iterator::operator!=** ([iterator](#) & *inp*) [**inline**]

12.23.3.2 **template<class Key, class Value> void Hashmap_Array< Key, Value >::iterator::operator++** (int) [**inline**]

12.23.4 Member Data Documentation

12.23.4.1 **template<class Key, class Value> [act_map_iter](#) Hashmap_Array< Key, Value >::iterator::current_iter**

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

- [src/commons/Server/HashMap_Array.h](#)

12.24 LDAUtil::Itoa Class Reference

```
#include <LDAUtil.h>
```

Static Public Member Functions

- static string [get_string](#) (const int *i*)

12.24.1 Member Function Documentation

12.24.1.1 string LDAUtil::Itoa::get_string (const int *i*) [static]

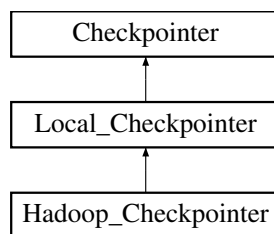
The documentation for this class was generated from the following files:

- src/commons/[LDAUtil.h](#)
- src/commons/[LDAUtil.cpp](#)

12.25 Local_Checkpointer Class Reference

```
#include <Local_Checkpointer.h>
```

Inheritance diagram for Local_Checkpointer:



Public Member Functions

- [Local_Checkpointer \(\)](#)
- virtual [~Local_Checkpointer \(\)](#)
- virtual void [save_metadata](#) (std::string &state)
Serialize the metadata.
- virtual std::string [load_metadata](#) ()
Load the metadata.
- virtual void [checkpoint](#) ()
Serialize other necessary data structures.

12.25.1 Constructor & Destructor Documentation

12.25.1.1 `Local_Checkpointer::Local_Checkpointer ()`

12.25.1.2 `Local_Checkpointer::~~Local_Checkpointer ()` [**virtual**]

12.25.2 Member Function Documentation

12.25.2.1 `void Local_Checkpointer::checkpoint ()` [**virtual**]

Serialize other necessary data structures.

Implements [Checkpointer](#).

Reimplemented in [Hadoop_Checkpointer](#).

12.25.2.2 `std::string Local_Checkpointer::load_metadata ()` **[virtual]**

Load the metadata.

Implements [Checkpointter](#).

12.25.2.3 `void Local_Checkpointer::save_metadata (std::string & state)`
[virtual]

Serialize the metadata.

Implements [Checkpointter](#).

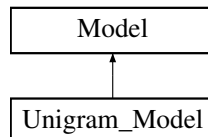
The documentation for this class was generated from the following files:

- [src/Unigram_Model/TopicLearner/Local_Checkpointer.h](#)
- [src/Unigram_Model/TopicLearner/Local_Checkpointer.cpp](#)

12.26 Model Class Reference

```
#include <Model.h>
```

Inheritance diagram for Model:



Public Member Functions

- virtual double [get_eval](#) ()=0
Model's contribution of log-likelihood.
- virtual bool [save](#) ()=0
Serialize to disk.
- virtual void [write_statistics](#) ([WordIndexDictionary](#) &)=0
Explain: word mixtures for the latent topics.

Static Public Attributes

- static const int [UNIGRAM](#) = 1

12.26.1 Detailed Description

A marker interface for the LDA graphical model and its extensions. A model should be able to compute its contribution to the log-likelihood, serialize to disk & also explain itself by writing the word mixtures that represent the latent topics to disk

12.26.2 Member Function Documentation

12.26.2.1 virtual double Model::get_eval () [pure virtual]

Model's contribution of log-likelihood.

Implemented in [Unigram_Model](#).

12.26.2.2 virtual bool Model::save () [pure virtual]

Serialize to disk.

Implemented in [Unigram_Model](#).

12.26.2.3 virtual void Model::write_statistics (WordIndexDictionary &) [pure virtual]

Explain: word mixtures for the latent topics.

Implemented in [Unigram_Model](#).

12.26.3 Member Data Documentation**12.26.3.1 const int Model::UNIGRAM = 1 [static]**

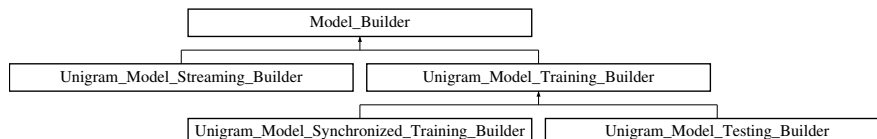
The documentation for this class was generated from the following file:

- [src/commons/TopicLearner/Model.h](#)

12.27 Model_Builder Class Reference

```
#include <Model_Builder.h>
```

Inheritance diagram for Model_Builder:



Public Member Functions

- virtual [Model_Refiner](#) & [create_model_refiner](#) ()=0
- virtual [Pipeline](#) & [create_pipeline](#) ([Model_Refiner](#) &)=0
- virtual [Execution_Strategy](#) & [create_execution_strategy](#) ([Pipeline](#) &)=0
- virtual void [create_output](#) ()=0
- virtual [Model](#) & [get_model](#) ()=0

12.27.1 Detailed Description

The builder class which builds the different components needed to create the model. We use the Builder pattern.

The components are: 1. [Model_Refiner](#): This main component that describes how the model should be refined as documents pass through the pipeline 2. [Pipeline](#): A pipeline of filters that perform the various refinements defined by the refiner 3. [Execution_Strategy](#): A strategy that defines how the pipeline of filters has to be executed

Usually, different modes demand different builders and similar or different components. So keep in mind that inheritance can be used here to maximize code reuse

12.27.2 Member Function Documentation

12.27.2.1 virtual [Execution_Strategy](#)& [Model_Builder](#)::[create_execution_strategy](#) ([Pipeline](#) &) `[pure virtual]`

Implemented in [Unigram_Model_Streaming_Builder](#), [Unigram_Model_Synchronized_Training_Builder](#), [Unigram_Model_Testing_Builder](#), and [Unigram_Model_Training_Builder](#).

**12.27.2.2 virtual Model_Refiner& Model_Builder::create_model_refiner ()
[pure virtual]**

Implemented in [Unigram_Model_Streaming_Builder](#), [Unigram_Model_Testing_Builder](#), and [Unigram_Model_Training_Builder](#).

12.27.2.3 virtual void Model_Builder::create_output () [pure virtual]

Implemented in [Unigram_Model_Streaming_Builder](#), and [Unigram_Model_Training_Builder](#).

**12.27.2.4 virtual Pipeline& Model_Builder::create_pipeline (Model_Refiner &)
[pure virtual]**

Implemented in [Unigram_Model_Streaming_Builder](#), and [Unigram_Model_Training_Builder](#).

12.27.2.5 virtual Model& Model_Builder::get_model () [pure virtual]

Implemented in [Unigram_Model_Streaming_Builder](#), and [Unigram_Model_Training_Builder](#).

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

- [src/commons/TopicLearner/Model_Builder.h](#)

12.28 Model_Director Class Reference

The Director class of the Builder pattern.

```
#include <Model_Director.h>
```

Public Member Functions

- [Model_Director \(\)](#)
- virtual [~Model_Director \(\)](#)
- [Model](#) & [build_model](#) ([Model_Builder](#) &builder)

12.28.1 Detailed Description

The Director class of the Builder pattern. Simple steps: Use the builder to 1. Create Refiner 2. Create a pipeline of filters to perform the various refinements defined by the Refiner 3. Create a strategy to execute the pipeline of filters 4. Execute the strategy 5. Return the model inside the builder that has been refined by the strategy

12.28.2 Constructor & Destructor Documentation

12.28.2.1 [Model_Director::Model_Director \(\)](#)

12.28.2.2 [Model_Director::~~Model_Director \(\)](#) [**virtual**]

12.28.3 Member Function Documentation

12.28.3.1 [Model](#) & [Model_Director::build_model](#) ([Model_Builder](#) & *builder*)

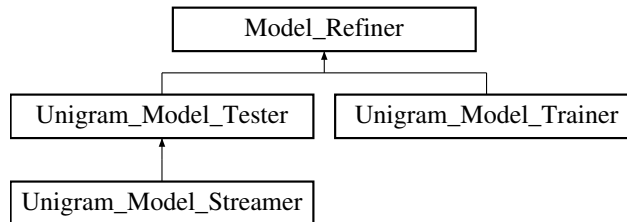
The documentation for this class was generated from the following files:

- src/commons/TopicLearner/[Model_Director.h](#)
- src/commons/TopicLearner/[Model_Director.cpp](#)

12.29 Model_Refiner Class Reference

```
#include <Model_Refiner.h>
```

Inheritance diagram for Model_Refiner:



Public Member Functions

- virtual google::protobuf::Message * [allocate_document_buffer](#) (size_t)=0
- virtual void [deallocate_document_buffer](#) (google::protobuf::Message *)=0
- virtual google::protobuf::Message * [get_nth_document](#) (google::protobuf::Message *docs, size_t n)=0
- virtual void * [read](#) (google::protobuf::Message &)=0
- virtual void * [sample](#) (void *)=0
- virtual void * [test](#) (void *)=0
- virtual void * [update](#) (void *)=0
- virtual void * [optimize](#) (void *)=0
- virtual void * [eval](#) (void *, double &)=0
- virtual void [write](#) (void *)=0
- virtual void [iteration_done](#) ()=0

12.29.1 Detailed Description

An interface that defines the necessary refinements for refining the LDA graphical model and its extensions. A refinement is defined as an operation done to improve the model like sampling. The definition is also extended to operations that enable a refinement like reading documents and evaluating log-likelihoods.

The provider of a model's implementation has to implement this interface suitably

12.29.2 Member Function Documentation

12.29.2.1 `virtual google::protobuf::Message* Model_Refiner::allocate_document_buffer (size_t) [pure virtual]`

Implemented in [Unigram_Model_Tester](#), and [Unigram_Model_Trainer](#).

12.29.2.2 `virtual void Model_Refiner::deallocate_document_buffer (google::protobuf::Message *) [pure virtual]`

Implemented in [Unigram_Model_Tester](#), and [Unigram_Model_Trainer](#).

12.29.2.3 `virtual void* Model_Refiner::eval (void *, double &) [pure virtual]`

Implemented in [Unigram_Model_Tester](#), and [Unigram_Model_Trainer](#).

12.29.2.4 `virtual google::protobuf::Message* Model_Refiner::get_nth_document (google::protobuf::Message * docs, size_t n) [pure virtual]`

Implemented in [Unigram_Model_Tester](#), and [Unigram_Model_Trainer](#).

12.29.2.5 `virtual void Model_Refiner::iteration_done () [pure virtual]`

Implemented in [Unigram_Model_Streamers](#), [Unigram_Model_Tester](#), and [Unigram_Model_Trainer](#).

12.29.2.6 `virtual void* Model_Refiner::optimize (void *) [pure virtual]`

Implemented in [Unigram_Model_Tester](#), and [Unigram_Model_Trainer](#).

12.29.2.7 `virtual void* Model_Refiner::read (google::protobuf::Message &) [pure virtual]`

Implemented in [Unigram_Model_Streamers](#), [Unigram_Model_Tester](#), and [Unigram_Model_Trainer](#).

12.29.2.8 virtual void* Model_Refiner::sample (void *) [pure virtual]

Implemented in [Unigram_Model_Tester](#), and [Unigram_Model_Trainer](#).

12.29.2.9 virtual void* Model_Refiner::test (void *) [pure virtual]

Implemented in [Unigram_Model_Tester](#), and [Unigram_Model_Trainer](#).

12.29.2.10 virtual void* Model_Refiner::update (void *) [pure virtual]

Implemented in [Unigram_Model_Tester](#), and [Unigram_Model_Trainer](#).

12.29.2.11 virtual void Model_Refiner::write (void *) [pure virtual]

Implemented in [Unigram_Model_Streamer](#), [Unigram_Model_Tester](#), and [Unigram_Model_Trainer](#).

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

- [src/commons/TopicLearner/Model_Refiner.h](#)

12.30 Parameter Struct Reference

```
#include <Parameter.h>
```

Public Member Functions

- [Parameter](#) ()
- [Parameter](#) (const [Parameter](#) ¶m)
- virtual [~Parameter](#) ()
- [Parameter](#) & [operator=](#) (const [Parameter](#) ¶m)
- virtual void [dump](#) (string fname)
- virtual void [initialize_from_dump](#) (string fname)
- virtual void [initialize_from_values](#) (int length_, const double *values_, float sum_)

Public Attributes

- int [length](#)
- double * [values](#)
- double [sum](#)

12.30.1 Detailed Description

A class to represent the various parameters like the dirichlet conjugate weights and such

Can be vector valued parameters

This class additionally stores the sum and provides functionality to dump to and initialize from disk

12.30.2 Constructor & Destructor Documentation

12.30.2.1 `Parameter::Parameter ()`

12.30.2.2 `Parameter::Parameter (const Parameter & param)`

12.30.2.3 `Parameter::~~Parameter ()` [**virtual**]

12.30.3 Member Function Documentation

12.30.3.1 `void Parameter::dump (string fname)` [**virtual**]

12.30.3.2 `void Parameter::initialize_from_dump (string fname)` [**virtual**]

12.30.3.3 `void Parameter::initialize_from_values (int length_, const double * values_, float sum_)` [**virtual**]

12.30.3.4 `Parameter & Parameter::operator= (const Parameter & param)`

12.30.4 Member Data Documentation

12.30.4.1 `int Parameter::length`

12.30.4.2 `double Parameter::sum`

12.30.4.3 `double* Parameter::values`

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

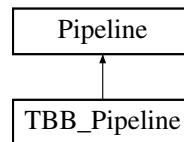
- [src/commons/TopicLearner/Parameter.h](#)
- [src/commons/TopicLearner/Parameter.cpp](#)

12.31 Pipeline Class Reference

An interface that all pipeline objects must implement.

```
#include <Pipeline.h>
```

Inheritance diagram for Pipeline:



Public Member Functions

- virtual void [init](#) ()=0
- virtual void [add_reader](#) ()=0
- virtual void [add_sampler](#) ()=0
- virtual void [add_updater](#) ()=0
- virtual void [add_optimizer](#) ()=0
- virtual void [add_eval](#) ()=0
- virtual void [add_writer](#) ()=0
- virtual void [add_tester](#) ()=0
- virtual void [clear](#) ()=0
- virtual void [destroy](#) ()=0
- virtual void [run](#) ()=0
- virtual [Model_Refiner](#) & [get_refiner](#) ()=0
- virtual double [get_eval](#) ()=0

12.31.1 Detailed Description

An interface that all pipeline objects must implement. A pipeline is a sequence of computation steps that can be performed on some data passing through it. Its similar to an assembly pipeline.

12.31.2 Member Function Documentation

12.31.2.1 virtual void Pipeline::add_eval () [pure virtual]

Implemented in [TBB_Pipeline](#).

12.31.2.2 virtual void Pipeline::add_optimizer () [pure virtual]

Implemented in [TBB_Pipeline](#).

12.31.2.3 virtual void Pipeline::add_reader () [pure virtual]

Implemented in [TBB_Pipeline](#).

12.31.2.4 virtual void Pipeline::add_sampler () [pure virtual]

Implemented in [TBB_Pipeline](#).

12.31.2.5 virtual void Pipeline::add_tester () [pure virtual]

Implemented in [TBB_Pipeline](#).

12.31.2.6 virtual void Pipeline::add_updater () [pure virtual]

Implemented in [TBB_Pipeline](#).

12.31.2.7 virtual void Pipeline::add_writer () [pure virtual]

Implemented in [TBB_Pipeline](#).

12.31.2.8 virtual void Pipeline::clear () [pure virtual]

Implemented in [TBB_Pipeline](#).

12.31.2.9 virtual void Pipeline::destroy () [pure virtual]

Implemented in [TBB_Pipeline](#).

12.31.2.10 virtual double Pipeline::get_eval () [pure virtual]

Implemented in [TBB_Pipeline](#).

12.31.2.11 virtual Model_Refiner& Pipeline::get_refiner () [pure virtual]

Implemented in [TBB_Pipeline](#).

12.31.2.12 virtual void Pipeline::init () [pure virtual]

Implemented in [TBB_Pipeline](#).

12.31.2.13 virtual void Pipeline::run () [pure virtual]

Implemented in [TBB_Pipeline](#).

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

- [src/commons/TopicLearner/Pipeline.h](#)

12.32 PNGCallback Class Reference

The call back object for the Put and Get AMI.

```
#include <DM_Client.h>
```

Public Member Functions

- [PNGCallback](#) ([Synchronizer_Helper](#) &sync_helper, int num_msgs)

The sync_helper and the max num of msgs to track.

- [~PNGCallback](#) ()
- bool [wait_till_done](#) (int msg_id)

Wait till the server responds to msg with id msg_id.

- void [set_done](#) (int msg_id, bool status)

set or clear the status flag for a particular msg_id

- int [num_synchs](#) ()

Provides the number of completed AMI callbacks.

- void [finished](#) (const Ice::AsyncResultPtr &r)

12.32.1 Detailed Description

The call back object for the Put and Get AMI. Ice calls the finished method once the AMI completes. The client is not tied to any particular call back. The actual call back is provided by a helper class, the sync_helper whose end method is called to complete the full AMI call.

This call back class also doubles up as a rate limiter implementing a sliding window mechanism for sending messages to the server.

Given a max number of msgs, it creates an array to track the status of the msgs that have been sent to the server

It provides a monitor based wait mechanism for the client which can wait till the server responds back to a particular msg. We take the liberty of making a fcfs assumption on the server side processing for keeping things simple.

12.32.2 Constructor & Destructor Documentation

12.32.2.1 PNGCallback::PNGCallback (Synchronizer_Helper & *sync_helper*, int *num_msgs*) [inline]

The *sync_helper* and the max num of msgs to track.

12.32.2.2 PNGCallback::~~PNGCallback () [inline]

12.32.3 Member Function Documentation

12.32.3.1 void PNGCallback::finished (const Ice::AsyncResultPtr & *r*) [inline]

This method is called by Ice run time when the server responds. We do some bookkeeping and delegate the task of actually completing the AMI by calling *sync_helper.end* method

12.32.3.2 int PNGCallback::num_synchs () [inline]

Provides the number of completed AMI callbacks.

12.32.3.3 void PNGCallback::set_done (int *msg_id*, bool *status*) [inline]

set or clear the status flag for a particular *msg_id*

12.32.3.4 bool PNGCallback::wait_till_done (int *msg_id*) [inline]

Wait till the server responds to msg with id *msg_id*.

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

- [src/commons/TopicLearner/DM_Client.h](#)

12.33 PNGJob Class Reference

```
#include <DM_Server.h>
```

Public Member Functions

- [PNGJob](#) (const AMD_DistributedMap_putNgetPtr &cb, const string &[word](#), const string &[delta](#))

Public Attributes

- AMD_DistributedMap_putNgetPtr [png_cb](#)
The call back object.
- string [word](#)
- string [delta](#)

12.33.1 Detailed Description

Container class for a put and get job. The putNget_async operation is an asynchronous operation with asynchronous method dispatch.

Hence a call should be queued up for later processing and this class stores all the details pertinent to that call.

12.33.2 Constructor & Destructor Documentation

12.33.2.1 [PNGJob::PNGJob](#) (const AMD_DistributedMap_putNgetPtr &*cb*, const string &*word*, const string &*delta*) [[inline](#)]

12.33.3 Member Data Documentation

12.33.3.1 string PNGJob::delta

The serialized form of data that needs to be accumulated into the entry pointed by word

12.33.3.2 AMD_DistributedMap_putNgetPtr PNGJob::png_cb

The call back object.

12.33.3.3 string PNGJob::word

The word on which this operation is called

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

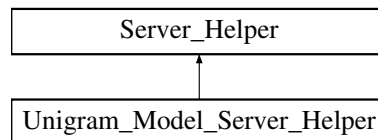
- src/commons/Server/[DM_Server.h](#)

12.34 Server_Helper Class Reference

Server Helper interface.

```
#include <Server_Helper.h>
```

Inheritance diagram for Server_Helper:



Public Member Functions

- virtual void [combine](#) (const string *entity*, const string &*old*, const string &*delta*, string &*combined*)=0

12.34.1 Detailed Description

Server Helper interface. Users of the Distributed Map need to provide semantics for the put operation using this helper class.

The put operation calls `helper.combine()` to perform the accumulate operation

All server helpers must implement this interface

12.34.2 Member Function Documentation

12.34.2.1 virtual void `Server_Helper::combine` (const string *entity*, const string &*old*, const string &*delta*, string &*combined*) [**pure virtual**]

Implemented in [Unigram_Model_Server_Helper](#).

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

- `src/commons/Server/Server_Helper.h`

12.35 simple_map Class Reference

```
#include <eff_small_map.h>
```

Public Member Functions

- [simple_map](#) (int $N=0x7F$)
- virtual [~simple_map](#) ()
- int [hash](#) (topic_t key)
- cnt_t [get](#) (topic_t key)
- void [put](#) (topic_t key, cnt_t val)
- std::string [print](#) ()
- void [clear](#) ()

12.35.1 Constructor & Destructor Documentation

12.35.1.1 [simple_map::simple_map](#) (int $N = 0x7F$)

12.35.1.2 [simple_map::~~simple_map](#) () [**virtual**]

12.35.2 Member Function Documentation

12.35.2.1 [void simple_map::clear](#) ()

12.35.2.2 [cnt_t simple_map::get](#) (topic_t *key*)

12.35.2.3 [int simple_map::hash](#) (topic_t *key*) [**inline**]

12.35.2.4 [std::string simple_map::print](#) ()

12.35.2.5 [void simple_map::put](#) (topic_t *key*, cnt_t *val*)

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

- src/Unigram_Model/TopicLearner/[eff_small_map.h](#)
- src/Unigram_Model/TopicLearner/[eff_small_map.cpp](#)

12.36 LDAUtil::StringTokenizer Class Reference

```
#include <LDAUtil.h>
```

Static Public Member Functions

- static void [tokenize](#) (const string &s, char delim, vector< string > &ret_vec)

12.36.1 Detailed Description

Utility class for tokenization and such other commonly used functions

12.36.2 Member Function Documentation

12.36.2.1 void LDAUtil::StringTokenizer::tokenize (const string & *s*, char *delim*, vector< string > & *ret_vec*) [**static**]

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

- src/commons/[LDAUtil.h](#)
- src/commons/[LDAUtil.cpp](#)

12.37 LDAUtil::StringTrimmer Class Reference

```
#include <LDAUtil.h>
```

Static Public Member Functions

- static void [trim](#) (string &s, char trim_char= ' ')

12.37.1 Member Function Documentation

12.37.1.1 void LDAUtil::StringTrimmer::trim (string & *s*, char *trim_char* = ' ') [**static**]

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

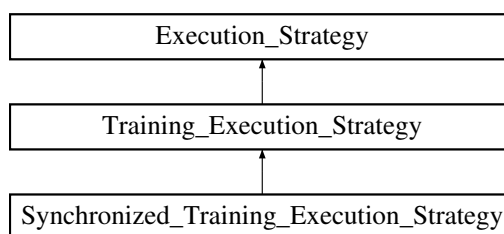
- src/commons/[LDAUtil.h](#)
- src/commons/[LDAUtil.cpp](#)

12.38 Synchronized_Training_Execution_Strategy Class Reference

Default implementation of the [Execution_Strategy](#) interface.

```
#include <Synchronized_Training_Execution_Strategy.h>
```

Inheritance diagram for Synchronized_Training_Execution_Strategy:



Public Member Functions

- [Synchronized_Training_Execution_Strategy](#) ([Pipeline](#) &, [Model](#) &, [Checkpointer](#) &, [Synchronizer_Helper](#) &)
- virtual [~Synchronized_Training_Execution_Strategy](#) ()
- void [execute](#) ()

12.38.1 Detailed Description

Default implementation of the [Execution_Strategy](#) interface. Runs the [Training_Execution_Strategy](#) at its core and also provides for synchronization of the model

12.38.2 Constructor & Destructor Documentation

12.38.2.1 `Synchronized_Training_Execution_Strategy::Synchronized_Training_Execution_Strategy (Pipeline & pipeline, Model & model, Checkpointer & checkpointer, Synchronizer_Helper & sync_helper)`

12.38.2.2 `Synchronized_Training_Execution_Strategy::~Synchronized_Training_Execution_Strategy ()` `[virtual]`

12.38.3 Member Function Documentation

12.38.3.1 `void Synchronized_Training_Execution_Strategy::execute ()` `[virtual]`

Define what filters need to added and when Runs the pipeline for 'iter' iterations

Reimplemented from [Training_Execution_Strategy](#).

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

- [src/commons/TopicLearner/Synchronized_Training_Execution_Strategy.h](#)
- [src/commons/TopicLearner/Synchronized_Training_Execution_Strategy.cpp](#)

12.39 Synchronizer Class Reference

```
#include <Synchronizer.h>
```

Public Member Functions

- [Synchronizer](#) ([Synchronizer_Helper](#) &)
- virtual [~Synchronizer](#) ()
- void [synchronize](#) ()
The synchronization strategy.
- void [set_all_iters_done](#) ()
- bool [is_all_iters_done](#) ()
Check if the [Execution_Strategy](#) has finished.

12.39.1 Detailed Description

The synchronization strategy that uses the [Synchronizer_Helper](#) to perform the concrete synchronization steps. It provides slots, so to say, for synchronization and calls the [Synchronizer_Helper](#) to perform its duties for that slot.

This greatly reduces the burden on the model writer

So whenever a model needs to be scaled by providing a distributed inferencing solution, all the model writer has to do is write a [Synchronizer_Helper](#).

This is a default synchronization strategy provided and users can implement their own strategy by extending this

The main aim is to run the inferencing locally while keeping the model in sync globally through the use of this [Synchronizer](#) and a Distributed Map

12.39.2 Constructor & Destructor Documentation

12.39.2.1 [Synchronizer::Synchronizer](#) ([Synchronizer_Helper](#) & *sync_helper*)

12.39.2.2 [Synchronizer::~~Synchronizer](#) () [**virtual**]

12.39.3 Member Function Documentation

12.39.3.1 bool [Synchronizer::is_all_iters_done](#) ()

Check if the [Execution_Strategy](#) has finished.

12.39.3.2 void Synchronizer::set_all_iters_done ()

Set that all iterations of the [Execution_Strategy](#) have been done

12.39.3.3 void Synchronizer::synchronize ()

The synchronization strategy.

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

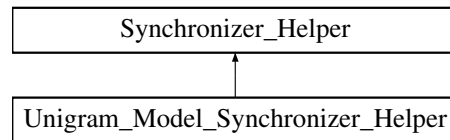
- [src/commons/TopicLearner/Synchronizer.h](#)
- [src/commons/TopicLearner/Synchronizer.cpp](#)

12.40 Synchronizer_Helper Class Reference

A helper class for the synchronizer.

```
#include <Synchronizer_Helper.h>
```

Inheritance diagram for Synchronizer_Helper:



Public Member Functions

- virtual void [initialize](#) ()=0
- virtual void [synchronize](#) ()=0
- virtual bool [has_to_synchronize](#) ()=0
- virtual void [reset_to_synchronize](#) ()=0
- virtual void [end_putNget](#) (const std::string &word, const std::string &counts)=0

12.40.1 Detailed Description

A helper class for the synchronizer. Each helper object implements synchronization algorithms for maintaining the data structures in sync

The [Synchronizer](#) class depends on this interface to enable synchronization of a multi-machine LDA set-up

12.40.2 Member Function Documentation

12.40.2.1 virtual void Synchronizer_Helper::end_putNget (const std::string &word, const std::string &counts) [pure virtual]

This is a callback from the [Client](#) when an `async_putNget` is used on the [Client](#). So when a [Client](#) is instantiated, you need to pass a reference of (`*this`)

Implemented in [Unigram_Model_Synchronizer_Helper](#).

12.40.2.2 virtual bool Synchronizer_Helper::has_to_synchronize () [pure virtual]

Returns true as long as all items to be synchronized are not synchronized

Implemented in [Unigram_Model_Synchronizer_Helper](#).

12.40.2.3 virtual void Synchronizer_Helper::initialize () [pure virtual]

Implemented in [Unigram_Model_Synchronizer_Helper](#).

12.40.2.4 virtual void Synchronizer_Helper::reset_to_synchronize () [pure virtual]

After this call, [has_to_synchronize\(\)](#) should return true

Implemented in [Unigram_Model_Synchronizer_Helper](#).

12.40.2.5 virtual void Synchronizer_Helper::synchronize () [pure virtual]

Implemented in [Unigram_Model_Synchronizer_Helper](#).

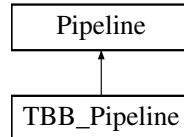
The documentation for this class was generated from the following file:

- [src/commons/TopicLearner/Synchronizer_Helper.h](#)

12.41 TBB_Pipeline Class Reference

```
#include <TBB_Pipeline.h>
```

Inheritance diagram for TBB_Pipeline:



Public Member Functions

- [TBB_Pipeline \(Model_Refiner &\)](#)
- [virtual ~TBB_Pipeline \(\)](#)
- [void init \(\)](#)
- [void add_reader \(\)](#)
- [void add_sampler \(\)](#)
- [void add_updater \(\)](#)
- [void add_optimizer \(\)](#)
- [void add_eval \(\)](#)
- [void add_writer \(\)](#)
- [void add_tester \(\)](#)
- [void clear \(\)](#)
- [void destroy \(\)](#)
- [void run \(\)](#)
- [Model_Refiner & get_refiner \(\)](#)
- [double get_eval \(\)](#)

Protected Attributes

- [task_scheduler_init _init](#)
- [tbb::pipeline * _pipeline](#)
- [Model_Refiner & _refiner](#)
- [filter * _reader](#)
- [filter * _sampler](#)
- [filter * _updater](#)
- [filter * _optimizer](#)
- [filter * _eval](#)
- [filter * _writer](#)
- [filter * _tester](#)

12.41.1 Detailed Description

An implementation of the [Pipeline](#) interface using Intel's Threading Building Blocks. TBB::filter is the basic unit of computation and TBB::Pipeline puts the filters together.

What filters exist in a pipeline and how the pipeline is executed is implemented via an Execution Strategy

Calling each of the add methods adds that particular filter to the pipeline

12.41.2 Constructor & Destructor Documentation

12.41.2.1 TBB_Pipeline::TBB_Pipeline (Model_Refiner & *refiner*)

12.41.2.2 TBB_Pipeline::~~TBB_Pipeline () [virtual]

12.41.3 Member Function Documentation

12.41.3.1 void TBB_Pipeline::add_eval () [virtual]

Implements [Pipeline](#).

12.41.3.2 void TBB_Pipeline::add_optimizer () [virtual]

Implements [Pipeline](#).

12.41.3.3 void TBB_Pipeline::add_reader () [virtual]

Implements [Pipeline](#).

12.41.3.4 void TBB_Pipeline::add_sampler () [virtual]

Implements [Pipeline](#).

12.41.3.5 void TBB_Pipeline::add_tester () [virtual]

Implements [Pipeline](#).

12.41.3.6 void TBB_Pipeline::add_updater () [virtual]

Implements [Pipeline](#).

12.41.3.7 void TBB_Pipeline::add_writer () [virtual]

Implements [Pipeline](#).

12.41.3.8 void TBB_Pipeline::clear () [virtual]

Implements [Pipeline](#).

12.41.3.9 void TBB_Pipeline::destroy () [virtual]

Implements [Pipeline](#).

12.41.3.10 double TBB_Pipeline::get_eval () [virtual]

Implements [Pipeline](#).

12.41.3.11 Model_Refiner & TBB_Pipeline::get_refiner () [virtual]

Implements [Pipeline](#).

12.41.3.12 void TBB_Pipeline::init () [virtual]

Implements [Pipeline](#).

12.41.3.13 void TBB_Pipeline::run () [virtual]

Implements [Pipeline](#).

12.41.4 Member Data Documentation

12.41.4.1 filter * TBB_Pipeline::_eval [protected]

12.41.4.2 task_scheduler_init TBB_Pipeline::_init [protected]

12.41.4.3 filter * TBB_Pipeline::_optimizer [protected]

12.41.4.4 tbb::pipeline* TBB_Pipeline::_pipeline [protected]

12.41.4.5 filter* TBB_Pipeline::_reader [protected]

12.41.4.6 Model_Refiner& TBB_Pipeline::_refiner [protected]

12.41.4.7 filter * TBB_Pipeline::_sampler [protected]

12.41.4.8 filter * TBB_Pipeline::_tester [protected]

12.41.4.9 filter * TBB_Pipeline::_updater [protected]

12.41.4.10 filter * TBB_Pipeline::_writer [protected]

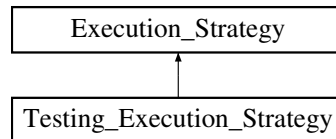
The documentation for this class was generated from the following files:

- [src/commons/TopicLearner/TBB_Pipeline.h](#)
- [src/commons/TopicLearner/TBB_Pipeline.cpp](#)

12.42 Testing_Execution_Strategy Class Reference

```
#include <Testing_Execution_Strategy.h>
```

Inheritance diagram for Testing_Execution_Strategy:



Public Member Functions

- [Testing_Execution_Strategy](#) ([Pipeline](#) &, [Model](#) &)
- virtual [~Testing_Execution_Strategy](#) ()
- void [execute](#) ()

12.42.1 Detailed Description

This is a default implementation for the [Execution_Strategy](#) interface for testing using a model.

12.42.2 Constructor & Destructor Documentation

12.42.2.1 `Testing_Execution_Strategy::Testing_Execution_Strategy (Pipeline & pipeline, Model & model)`

12.42.2.2 `Testing_Execution_Strategy::~~Testing_Execution_Strategy ()`
[**virtual**]

12.42.3 Member Function Documentation

12.42.3.1 `void Testing_Execution_Strategy::execute ()` [**virtual**]

Define what filters need to added and when Runs the assembled pipeline once

Implements [Execution_Strategy](#).

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

- `src/commons/TopicLearner/Testing_Execution_Strategy.h`
- `src/commons/TopicLearner/Testing_Execution_Strategy.cpp`

12.43 TopicCounts Struct Reference

```
#include <TopicCounts.h>
```

Public Member Functions

- [TopicCounts](#) ()
- [TopicCounts](#) (int [length](#))
- [TopicCounts](#) (cnt_topic_t *it, int len)
- [TopicCounts](#) (const std::string &counts)
- void [init](#) (cnt_topic_t *it, int len)
- void [init](#) (const std::string &counts)
- [~TopicCounts](#) ()
- void [assign](#) (int [length](#), bool setLen=true)
- void [setLength](#) (int [length_](#))
- void [findOldnNew](#) (topic_t oldTopic, topic_t newTopic, topic_t **oldTop, topic_t **newTop)
- int [get_frequency](#) ()
- cnt_t [get_counts](#) (topic_t topic)
- int [convertTo](#) ([mapped_vec](#) &map, int mult=1) const
- void [convertTo](#) ([simple_map](#) &map, int mult=1) const
- void [convertTo](#) (std::string &counts) const
- int [convertTo_d](#) ([mapped_vec](#) &map, double mult) const
- bool [findAndIncrement](#) (topic_t topic)
- bool [findAndDecrement](#) (topic_t topic)
- void [compact](#) ()
- void [addNewTop](#) (topic_t topic, cnt_t count=1)
- void [addNewTopAftChk](#) (topic_t topic, cnt_t count=1)
- void [upd_count](#) ([mapped_vec](#) &delta, tbb::atomic< topic_t > *t=NULL)
- void [operator+=](#) ([TopicCounts](#) &inp)
- void [operator-=](#) ([TopicCounts](#) &inp)
- void [removeOldTop](#) (topic_t ind, cnt_topic_t &ct)
- void [replace](#) ([TopicCounts](#) &tc)
- void [decrement](#) (topic_t ind, topic_t **newTop)
- void [increment](#) (topic_t ind)
- std::string [print](#) ()
- [TopicCounts](#) ([mapped_vec](#) &map)
- bool [equal](#) (const [TopicCounts](#) &expected)

Public Attributes

- cnt_topic_t * [items](#)
- topic_t [length](#)
- topic_t [origLength](#)
- std::vector< cnt_topic_t > [vec_items](#)
- int [frequency](#)
- bool [QUIT](#)

12.43.1 Constructor & Destructor Documentation

12.43.1.1 TopicCounts::TopicCounts ()

12.43.1.2 TopicCounts::TopicCounts (int *length*)

Typical constructor usage. Constructs a [TopicCounts](#) structure which can hold length elements. However, this is a dynamic structure and length can change in the process of usage by the use of setter methods This also allocates sufficient storage to store length elements

12.43.1.3 TopicCounts::TopicCounts (cnt_topic_t * *it*, int *len*)

Constructs [TopicCounts](#) from it & len

12.43.1.4 TopicCounts::TopicCounts (const std::string & *counts*)

12.43.1.5 TopicCounts::~~TopicCounts ()

12.43.1.6 TopicCounts::TopicCounts (mapped_vec & *map*)

Initialize the structure from the map This is only for testing purposes and should be used with caution.

12.43.2 Member Function Documentation

12.43.2.1 void TopicCounts::addNewTop (topic_t *topic*, cnt_t *count* = 1)

Adds a new topic to the array making sure to resize it if required. Doesn't check for uniqueness Also assumes that memory allocation has happened

12.43.2.2 void TopicCounts::addNewTopAftChk (topic_t *topic*, cnt_t *count* = 1)

Same as above but also checks if memory has been allocated. If not, tries to allocate memory. The check for allocated memory is not very robust as all the members are public and malicious users can set things such that it does not work.

12.43.2.3 void TopicCounts::assign (int *length*, bool *setLen* = true)

The function that allocates memory. origLength memory must be allocated since length can grow upto origLenth. It supports finer granularity access. You can disable the block alignment of lenth which is done by default using setLen if you have already made sure of this

12.43.2.4 void TopicCounts::compact ()

Checks if there is memory wastage and tries to reduce it by compacting memory. It does compaction only when there are more than INIT_TC_SIZE + SUBSEQ_ALLOCS entries and there is at least INIT_TC_ZISE memory being wasted. In such a case, it compact SUBSEQ_ALLOCS of memory. Usually SUBSEQ_ALLOCS should be set to half of INIT_TC_SIZE

***** All pointers to items are invalid once this runs successfully. Use extra caution when you use this or some method which uses this *****

12.43.2.5 void TopicCounts::convertTo (std::string & *counts*) const**12.43.2.6 void TopicCounts::convertTo (simple_map & *map*, int *mult* = 1) const****12.43.2.7 int TopicCounts::convertTo (mapped_vec & *map*, int *mult* = 1) const**

Stores a map representation of this vector into map. You can specify an optional multiplier to get a representation of (-1 * TC) or (2 * TC) where TC is the current structure. This multiplies count of each topic by the multiplier

12.43.2.8 int TopicCounts::convertTo_d (mapped_vec & *map*, double *mult*) const**12.43.2.9 void TopicCounts::decrement (topic_t *ind*, topic_t ** *newTop*)**

Used to decrement the count of the topic found at index ind by 1. Also takes care of repointing newTop if its position changes.

12.43.2.10 `bool TopicCounts::equal (const TopicCounts & expected)`

12.43.2.11 `bool TopicCounts::findAndDecrement (topic_t topic)`

12.43.2.12 `bool TopicCounts::findAndIncrement (topic_t topic)`

If you want to increment the count of a topic by 1 without knowing the index, you can use this method. But usually increment and decrement are joint operations and you would not want to iterate through the entire array twice. Its advantageous to use the findOldnNew function and increment & decrement methods directly as in `TypeTopicCounts::upd_count(word,oldTop,newTop)`

12.43.2.13 `void TopicCounts::findOldnNew (topic_t oldTopic, topic_t newTopic, topic_t ** oldTop, topic_t ** newTop)`

Typical ways of creating [TopicCounts](#): 1.TopicCounts(length) 2.TopicCounts();assign(length);init(items,length) 3.TopicCounts();assign(-1,false);init(items,length) Find oldTopic & newTopic positions in items and return them in oldTop & newTop. Note that cnt_topic_t is a packed structure. So accessing either the topic or count individually will modify your pointer arithmetic by two

12.43.2.14 `cnt_t TopicCounts::get_counts (topic_t topic)`

12.43.2.15 `int TopicCounts::get_frequency ()`

12.43.2.16 `void TopicCounts::increment (topic_t ind)`

Used to increment by 1 the count of an existing topic found at index ind. This assumes that the ind is in range and hence that the topic exists

12.43.2.17 `void TopicCounts::init (const std::string & counts)`

12.43.2.18 `void TopicCounts::init (cnt_topic_t * it, int len)`

This assumes that sufficient memory has already been allocated to hold len elems and just copies them to items

12.43.2.19 `void TopicCounts::operator+= (TopicCounts & inp)`

Convenience operators for syntactic sugar

12.43.2.20 void TopicCounts::operator-= (TopicCounts & *inp*)

Convenience operators for syntactic sugar

12.43.2.21 std::string TopicCounts::print ()**12.43.2.22 void TopicCounts::removeOldTop (topic_t *ind*, cnt_topic_t & *ct*)**

Should be called when you know that `ct.choose.cnt==1` or `ct.choose.cnt-1==0`. This will logically remove this entry from the array also taking care of repointing the new-Top to the correct position once `ct` is removed. After length is decremented it is checked if compaction is needed and if so, SUBSEQ_ALLOCS of memory is compacted. The removal is simply a swap with the last element since the current count is 1 and the items array is always sorted in descending order and only has non-zero entries. Also all pointers to items are going to be invalid if compaction runs successfully.

***** Be extra careful while using this *****

12.43.2.23 void TopicCounts::replace (TopicCounts & *tc*)

Replace the current counts with those in `tc`

12.43.2.24 void TopicCounts::setLength (int *length_*)

Block aligns the length and sets the `origLength` parameter to suit that. `origLength` is always `INIT_TC + i * SUBSEQ_ALLOCS` where `i=0,1,2,...`

12.43.2.25 void TopicCounts::upd_count (mapped_vec & *delta*, tbb::atomic<topic_t > * *t* = NULL)

Update items with the delta counts passed as a map Also update `n(t)` simultaneously using `t`. Note that `delta` is modified because elements are removed from it. If you don't want this, create a copy and send it along. The strategy is to update items with the counts in `delta` in place and sort it finally.

12.43.3 Member Data Documentation

12.43.3.1 `int TopicCounts::frequency`

12.43.3.2 `cnt_topic_t* TopicCounts::items`

12.43.3.3 `topic_t TopicCounts::length`

12.43.3.4 `topic_t TopicCounts::origLength`

12.43.3.5 `bool TopicCounts::QUIT`

12.43.3.6 `std::vector<cnt_topic_t> TopicCounts::vec_items`

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

- `src/Unigram_Model/TopicLearner/TopicCounts.h`
- `src/Unigram_Model/TopicLearner/TopicCounts.cpp`

12.44 TopKList Class Reference

```
#include <TopKList.h>
```

Public Types

- typedef cnt_topic_t * [iterator](#)

Public Member Functions

- [TopKList](#) (int K_)
- virtual [~TopKList](#) ()
- void [insert_word](#) (const cnt_word_t &cnt_word)
- bool [is_sorted](#) ()
- cnt_word_t [get_max](#) ()
- [iterator](#) [get_beg](#) ()
- [iterator](#) [get_end](#) ()
- void [print](#) ()
- void [clear](#) ()

12.44.1 Member Typedef Documentation

12.44.1.1 typedef cnt_topic_t* TopKList::iterator

12.44.2 Constructor & Destructor Documentation

12.44.2.1 TopKList::TopKList (int K_)

Constructs a [TopKList](#) supporting top K_ elements

12.44.2.2 TopKList::~~TopKList () [virtual]

12.44.3 Member Function Documentation

12.44.3.1 void TopKList::clear ()

Clears the list

12.44.3.2 TopKList::iterator TopKList::get_beg ()

Iterator methods. Return an iterator to the beginning of the list

12.44.3.3 TopKList::iterator TopKList::get_end ()

Iterator methods. Return an iterator to the end of the list

12.44.3.4 cnt_word_t TopKList::get_max ()

Returns the max element which is the first elem

12.44.3.5 void TopKList::insert_word (const cnt_word_t & *cnt_word*)**12.44.3.6 bool TopKList::is_sorted ()**

Used for test convenience. Just to check the array is always sorted and has at most K elems. Should return true at any point of time

12.44.3.7 void TopKList::print ()

Print the list to log(INFO)

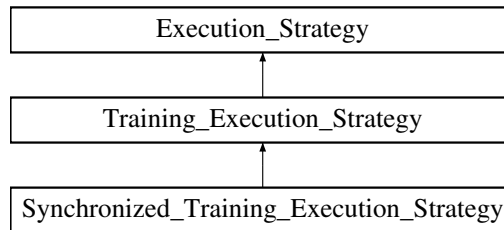
The documentation for this class was generated from the following files:

- src/Unigram_Model/TopicLearner/[TopKList.h](#)
- src/Unigram_Model/TopicLearner/[TopKList.cpp](#)

12.45 Training_Execution_Strategy Class Reference

```
#include <Training_Execution_Strategy.h>
```

Inheritance diagram for Training_Execution_Strategy:



Public Member Functions

- [Training_Execution_Strategy](#) ([Pipeline](#) &, [Model](#) &, [Checkpoint](#)er &)
- virtual [~Training_Execution_Strategy](#) ()
- void [execute](#) ()

12.45.1 Detailed Description

This is a default implementation for the [Execution_Strategy](#) interface for training a model.

This can use a passed in [Checkpoint](#)er to do failure recovery

A pipeline needs to be passed.

12.45.2 Constructor & Destructor Documentation

12.45.2.1 [Training_Execution_Strategy::Training_Execution_Strategy](#)
([Pipeline](#) & *pipeline*, [Model](#) & *model*, [Checkpoint](#)er & *checkpoint*er)

12.45.2.2 [Training_Execution_Strategy::~~Training_Execution_Strategy](#) ()
[**virtual**]

12.45.3 Member Function Documentation

12.45.3.1 void [Training_Execution_Strategy::execute](#) () [**virtual**]

Define what filters need to added and when Runs the pipeline for 'iter' iterations

Implements [Execution_Strategy](#).

Reimplemented in [Synchronized_Training_Execution_Strategy](#).

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

- [src/commons/TopicLearner/Training_Execution_Strategy.h](#)
- [src/commons/TopicLearner/Training_Execution_Strategy.cpp](#)

12.46 TypeTopicCounts Class Reference

```
#include <TypeTopicCounts.h>
```

Public Member Functions

- [TypeTopicCounts](#) ()
- [TypeTopicCounts](#) (word_t num_words, topic_t num_topics)
- virtual [~TypeTopicCounts](#) ()
- void [initialize_from_docs](#) (string wfname, string tfname)
- int [verify_header](#) (DocumentReader &doc_rdr)
- void [initialize_from_string](#) (word_t word, string &counts)
- bool [initialize_from_dump](#) (string fname, [WordIndexDictionary](#) *local_dict=NULL, [WordIndexDictionary](#) *global_dict=NULL, size_t offset=0)
- void [initialize_from_ttc](#) ([TypeTopicCounts](#) *ttc)
- void [estimate_alphas](#) (double *alphas, double &alpha_sum)
- void [dump](#) (string fname)
- topic_t [get_counts](#) (word_t word, [topicCounts](#) *tc)
- topic_t [get_counts](#) (atomic< topic_t > *tc)
- word_t [get_num_words](#) ()
- topic_t [get_num_topics](#) ()
- word_mutex_t * [get_lock](#) (word_t word)
- pair< [TopKList](#) **, [TopKList](#) * > [get_topic_stats](#) ()
- void [replace](#) (word_t word, [topicCounts](#) &tc)
- void [upd_count](#) (word_t word, topic_t old_topic, topic_t new_topic, bool ignore_old_topic=false)
- void [upd_count](#) (word_t word, [mapped_vec](#) delta, string dbg="")
- bool [equal](#) (const [TypeTopicCounts](#) &expected)
- string [print](#) (word_t word)
- void [print](#) ()
- void [initialize](#) ([topicCounts](#) *wtc, atomic< topic_t > *tc, word_t word=0)
- void [initialize](#) ([topicCounts](#) **wtc, atomic< topic_t > *tc)

Static Public Member Functions

- static pair< int, float > [estimate_fit](#) (string fname, [WordIndexDictionary](#) *dict)
- static pair< int, float > [estimate_fit](#) (string fname, float used_memory, int &incoming_words)

Protected Member Functions

- void [estimate_memoryn_warn](#) (long num_elems)
- void [clear_stats](#) ()
- void [init](#) (topic_t num_topics_)
- void [destroy](#) ()

Protected Attributes

- atomic< topic_t > * [tokens_per_topic](#)
- topic_t [num_topics](#)
- TopKList ** [topic_stats](#)
- TopKList * [top_topics](#)

Friends

- class [Memcached_Synchronizer](#)

12.46.1 Constructor & Destructor Documentation

12.46.1.1 TypeTopicCounts::TypeTopicCounts ()

Construct an empty Word_Topic Counts table

12.46.1.2 TypeTopicCounts::TypeTopicCounts (word_t num_words_, topic_t num_topics_)

Constructs a Word Topic Counts table that will have num_words_ unique words and each word can be assigned a maximum of num_topics_ topics

12.46.1.3 TypeTopicCounts::~TypeTopicCounts () [virtual]

12.46.2 Member Function Documentation

12.46.2.1 void TypeTopicCounts::clear_stats () [protected]

Clears all the topic statistics

12.46.2.2 void TypeTopicCounts::destroy () [protected]

12.46.2.3 void TypeTopicCounts::dump (string *fname*)

Use the [DocumentWriter](#) to dump the topicCounts and the n(t) into a dump file (fname).
Writes num_words topicCounts vectors and then n(t)

12.46.2.4 bool TypeTopicCounts::equal (const TypeTopicCounts & *expected*)

12.46.2.5 void TypeTopicCounts::estimate_alphas (double * *alphas*, double & *alpha_sum*)

12.46.2.6 pair< int, float > TypeTopicCounts::estimate_fit (string *fname*, float *used_memory*, int & *incoming_words*) [static]

12.46.2.7 pair< int, float > TypeTopicCounts::estimate_fit (string *fname*, WordIndexDictionary * *dict*) [static]

12.46.2.8 void TypeTopicCounts::estimate_memoryn_warn (long *num_elems*) [protected]

Used to estimate the amount of memory being used while initializing this structure in order to warn if it exceeds WARN_MEMORY_SIZE and fail if it exceeds MAX_MEMORY_USAGE

12.46.2.9 topic_t TypeTopicCounts::get_counts (atomic< topic_t > * *tc*)

The memory for tc is assumed to be allocated. This method just copies the n(t) into tc

12.46.2.10 topic_t TypeTopicCounts::get_counts (word_t *word*, topicCounts * *tc*)

The memory for topicCounts is assumed to have been allocated This method then just copied the topicCounts vector from the table into tc

12.46.2.11 word_mutex_t * TypeTopicCounts::get_lock (word_t *word*)

If you want to lock the structure externally, you can get the lock that controls access to the word by this method Method: word_mutex_t::scoped_lock lock(*get_lock(word), false);

12.46.2.12 `topic_t TypeTopicCounts::get_num_topics ()`

The number of topics being learnt

12.46.2.13 `word_t TypeTopicCounts::get_num_words ()`

The number of unique words for which the topic counts are maintained

12.46.2.14 `pair< TopKList **, TopKList * > TypeTopicCounts::get_topic_stats ()`

Get the TopicStats per topic & the hot/top NUM_TOP_TOPICS topics. Used to print the topic stats for the top topics

12.46.2.15 `void TypeTopicCounts::init (topic_t num_topics_) [protected]`**12.46.2.16** `void TypeTopicCounts::initialize (topicCounts ** wtc, atomic< topic_t > * tc)`

Initialize using and array of topicCounts. Used in testing. Refer TypeTopicCountsTest.cpp

12.46.2.17 `void TypeTopicCounts::initialize (topicCounts * wtc, atomic< topic_t > * tc, word_t word = 0)`

Initialize the structure using explicit topicCounts for word. Used in testing. Refer TypeTopicCountsTest.cpp

12.46.2.18 `void TypeTopicCounts::initialize_from_docs (string wfname, string tfname)`

Reads documents using [DocumentReader](#). For each document and for each word encountered, it updates the topic counts for that word based on the topic assignment.

12.46.2.19 `bool TypeTopicCounts::initialize_from_dump (string fname, WordIndexDictionary * local_dict = NULL, WordIndexDictionary * global_dict = NULL, size_t offset = 0)`

Reads the serialize dump (fname) in the protobufere format into memory. The dump is generated using [dump\(\)](#) method. The [DocumentReader](#) is used to read from the pro-

tobuf format dump file. There will be num_words entries in the dump for topicCounts and the last entry is the n(t). So first we read num_words topicCounts & then n(t)

12.46.2.20 `void TypeTopicCounts::initialize_from_string (word_t word, string & counts)`

12.46.2.21 `void TypeTopicCounts::initialize_from_ttc (TypeTopicCounts * ttc)`

12.46.2.22 `void TypeTopicCounts::print ()`

Dump the structure to log(INFO)

12.46.2.23 `string TypeTopicCounts::print (word_t word)`

12.46.2.24 `void TypeTopicCounts::replace (word_t word, topicCounts & tc)`

Replace the counts for this word with those in tc

12.46.2.25 `void TypeTopicCounts::upd_count (word_t word, mapped_vec delta, string dbg = "")`

Update the counts for word using the delta map. The map contains the topic to count deltas. A lock is obtained on the word and is delegated to the relevant topicCounts Used by the background synchronizers

12.46.2.26 `void TypeTopicCounts::upd_count (word_t word, topic_t old_topic, topic_t new_topic, bool ignore_old_topic = false)`

Main update used by the updater filter. Decrements the old_topic counts by 1 and increments the new_topic counts by 1 for word. Acquire a lock. Find the old and new topic locations and update them using the topicCounts vector's methods. However the decrement and increment methods take an index and do fast updates. So there is some pointer arithmetic going on here to find the index into the vector using the position pointers. Also note that the basic structure used has the topic and count packed in a 64 bit integer. So whenever we want to refer to the topic or count individually we need to multiply or divide by 2

12.46.2.27 `int TypeTopicCounts::verify_header (DocumentReader & doc_rdr)`

12.46.3 Friends And Related Function Documentation

12.46.3.1 `friend class Memcached_Synchronizer` [`friend`]

12.46.4 Member Data Documentation

12.46.4.1 `topic_t TypeTopicCounts::num_topics` [`protected`]

12.46.4.2 `atomic<topic_t>* TypeTopicCounts::tokens_per_topic`
[`protected`]

12.46.4.3 `TopKList * TypeTopicCounts::top_topics` [`protected`]

12.46.4.4 `TopKList** TypeTopicCounts::topic_stats` [`protected`]

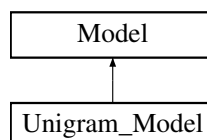
The documentation for this class was generated from the following files:

- `src/Unigram_Model/TopicLearner/TypeTopicCounts.h`
- `src/Unigram_Model/TopicLearner/TypeTopicCounts.cpp`

12.47 Unigram_Model Class Reference

```
#include <Unigram_Model.h>
```

Inheritance diagram for Unigram_Model:



Public Member Functions

- [Unigram_Model](#) (int, int)
- virtual [~Unigram_Model](#) ()
- [Parameter](#) & [get_parameter](#) (int)
- void [set_parameter](#) (int, [Parameter](#) &)
- [TypeTopicCounts](#) & [get_ttc](#) ()
- double [get_eval](#) ()

Model's contribution of log-likelihood.

- bool [save](#) ()
- void [write_statistics](#) ([WordIndexDictionary](#) &)

Explain: word mixtures for the latent topics.

Static Public Attributes

- static const int [ALPHA](#) = 1
- static const int [BETA](#) = 2

12.47.1 Constructor & Destructor Documentation

12.47.1.1 `Unigram_Model::Unigram_Model (int num_words, int num_topics)`

12.47.1.2 `Unigram_Model::~~Unigram_Model ()` `[virtual]`

12.47.2 Member Function Documentation

12.47.2.1 `double Unigram_Model::get_eval ()` `[virtual]`

Model's contribution of log-likelihood.

Implements [Model](#).

12.47.2.2 `Parameter & Unigram_Model::get_parameter (int which)`

12.47.2.3 `TypeTopicCounts & Unigram_Model::get_ttc ()`

12.47.2.4 `bool Unigram_Model::save ()` `[virtual]`

Serialize to disk.

Implements [Model](#).

12.47.2.5 `void Unigram_Model::set_parameter (int which, Parameter & param)`

12.47.2.6 `void Unigram_Model::write_statistics (WordIndexDictionary &)` `[virtual]`

Explain: word mixtures for the latent topics.

Implements [Model](#).

12.47.3 Member Data Documentation

12.47.3.1 `const int Unigram_Model::ALPHA = 1` `[static]`

12.47.3.2 `const int Unigram_Model::BETA = 2` `[static]`

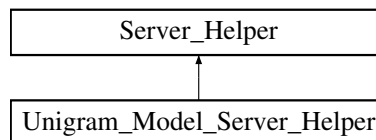
The documentation for this class was generated from the following files:

- `src/Unigram_Model/TopicLearner/Unigram_Model.h`
- `src/Unigram_Model/TopicLearner/Unigram_Model.cpp`

12.48 Unigram_Model_Server_Helper Class Reference

```
#include <Unigram_Model_Server_Helper.h>
```

Inheritance diagram for Unigram_Model_Server_Helper:



Public Member Functions

- [Unigram_Model_Server_Helper \(\)](#)
- virtual [~Unigram_Model_Server_Helper \(\)](#)
- void [combine](#) (const string *entity*, const string &*old*, const string &*delta*, string &*combined*)

12.48.1 Constructor & Destructor Documentation

12.48.1.1 `Unigram_Model_Server_Helper::Unigram_Model_Server_Helper ()`

12.48.1.2 `Unigram_Model_Server_Helper::~~Unigram_Model_Server_Helper () [virtual]`

12.48.2 Member Function Documentation

12.48.2.1 `void Unigram_Model_Server_Helper::combine (const string entity, const string & old, const string & delta, string & combined) [virtual]`

Implements [Server_Helper](#).

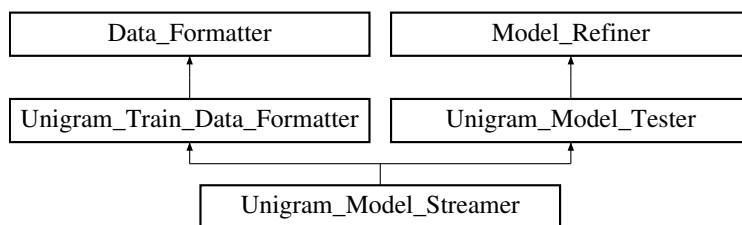
The documentation for this class was generated from the following files:

- `src/Unigram_Model/Server/Unigram_Model_Server_Helper.h`
- `src/Unigram_Model/Server/Unigram_Model_Server_Helper.cpp`

12.49 Unigram_Model_Streamers Class Reference

```
#include <Unigram_Model_Streamers.h>
```

Inheritance diagram for Unigram_Model_Streamers:



Public Member Functions

- [Unigram_Model_Streamers](#) ([TypeTopicCounts](#) &, [Parameter](#) &, [Parameter](#) &, [WordIndexDictionary](#) &, [WordIndexDictionary](#) &)
- virtual [~Unigram_Model_Streamers](#) ()
- void * [read](#) (google::protobuf::Message &)
- void [write](#) (void *)
- void [iteration_done](#) ()

Protected Member Functions

- int [insert_word_to_dict](#) (std::string word)

12.49.1 Constructor & Destructor Documentation

12.49.1.1 `Unigram_Model_Streamers::Unigram_Model_Streamers (TypeTopicCounts & ttc, Parameter & alpha, Parameter & beta, WordIndexDictionary & local_dict, WordIndexDictionary & global_dict)`

12.49.1.2 `Unigram_Model_Streamers::~~Unigram_Model_Streamers ()`
[**virtual**]

12.49.2 Member Function Documentation

12.49.2.1 `int Unigram_Model_Streamers::insert_word_to_dict (std::string word)` [**protected**, **virtual**]

Reimplemented from [Unigram_Train_Data_Formatter](#).

12.49.2.2 `void Unigram_Model_Streamers::iteration_done ()` [**virtual**]

Reimplemented from [Unigram_Model_Tester](#).

12.49.2.3 `void * Unigram_Model_Streamers::read (google::protobuf::Message & doc)` [**virtual**]

Reimplemented from [Unigram_Model_Tester](#).

12.49.2.4 `void Unigram_Model_Streamers::write (void * token)` [**virtual**]

Reimplemented from [Unigram_Model_Tester](#).

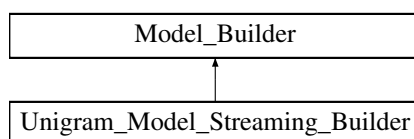
The documentation for this class was generated from the following files:

- [src/Unigram_Model/TopicLearner/Unigram_Model_Streamers.h](#)
- [src/Unigram_Model/TopicLearner/Unigram_Model_Streamers.cpp](#)

12.50 Unigram_Model_Streaming_Builder Class Reference

```
#include <Unigram_Model_Streaming_Builder.h>
```

Inheritance diagram for Unigram_Model_Streaming_Builder:



Public Member Functions

- [Unigram_Model_Streaming_Builder \(\)](#)
- virtual [~Unigram_Model_Streaming_Builder \(\)](#)
- virtual [Model_Refiner & create_model_refiner \(\)](#)
- virtual [Pipeline & create_pipeline \(Model_Refiner &\)](#)
- virtual [Execution_Strategy & create_execution_strategy \(Pipeline &\)](#)
- void [create_output \(\)](#)
- [Model & get_model \(\)](#)

Protected Member Functions

- void [init_dict \(\)](#)
- [WordIndexDictionary & get_dict \(\)](#)

Protected Attributes

- [Unigram_Model * _model](#)
- [WordIndexDictionary * _dict](#)
- [WordIndexDictionary * _global_dict](#)
- [Model_Refiner * _refiner](#)
- [Pipeline * _pipeline](#)
- [Execution_Strategy * _strategy](#)

12.50.1 Constructor & Destructor Documentation

12.50.1.1 `Unigram_Model_Streaming_Builder::Unigram_Model_Streaming_Builder ()`

12.50.1.2 `Unigram_Model_Streaming_Builder::~~Unigram_Model_Streaming_Builder ()` `[virtual]`

12.50.2 Member Function Documentation

12.50.2.1 `Execution_Strategy & Unigram_Model_Streaming_Builder::create_execution_strategy (Pipeline & pipeline)` `[virtual]`

Implements [Model_Builder](#).

12.50.2.2 `Model_Refiner & Unigram_Model_Streaming_Builder::create_model_refiner ()` `[virtual]`

Implements [Model_Builder](#).

12.50.2.3 `void Unigram_Model_Streaming_Builder::create_output ()` `[virtual]`

Implements [Model_Builder](#).

12.50.2.4 `Pipeline & Unigram_Model_Streaming_Builder::create_pipeline (Model_Refiner & refiner)` `[virtual]`

Implements [Model_Builder](#).

12.50.2.5 `WordIndexDictionary & Unigram_Model_Streaming_Builder::get_dict ()` `[protected]`

12.50.2.6 `Model & Unigram_Model_Streaming_Builder::get_model ()` `[virtual]`

Implements [Model_Builder](#).

12.50.2.7 void Unigram_Model_Streaming_Builder::init_dict ()
[protected]

12.50.3 Member Data Documentation

12.50.3.1 WordIndexDictionary* Unigram_Model_Streaming_Builder::_dict
[protected]

12.50.3.2 WordIndexDictionary* Unigram_Model_Streaming_Builder::_global_dict [protected]

12.50.3.3 Unigram_Model* Unigram_Model_Streaming_Builder::_model
[protected]

12.50.3.4 Pipeline* Unigram_Model_Streaming_Builder::_pipeline
[protected]

12.50.3.5 Model_Refiner* Unigram_Model_Streaming_Builder::_refiner
[protected]

12.50.3.6 Execution_Strategy* Unigram_Model_Streaming_Builder::_strategy
[protected]

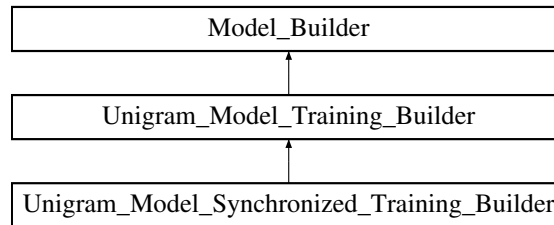
The documentation for this class was generated from the following files:

- src/Unigram_Model/TopicLearner/[Unigram_Model_Streaming_Builder.h](#)
- src/Unigram_Model/TopicLearner/[Unigram_Model_Streaming_Builder.cpp](#)

12.51 Unigram_Model_Synchronized_Training_ - Builder Class Reference

```
#include <Unigram_Model_Synchronized_Training_Builder.h>
```

Inheritance diagram for Unigram_Model_Synchronized_Training_Builder:



Public Member Functions

- [Unigram_Model_Synchronized_Training_Builder \(\)](#)
- virtual [~Unigram_Model_Synchronized_Training_Builder \(\)](#)
- virtual [Execution_Strategy & create_execution_strategy \(Pipeline &\)](#)

Protected Attributes

- [Synchronizer_Helper * _sync_helper](#)

12.51.1 Constructor & Destructor Documentation

12.51.1.1 [Unigram_Model_Synchronized_Training_ - Builder::Unigram_Model_Synchronized_Training_Builder \(\)](#)

12.51.1.2 [Unigram_Model_Synchronized_Training_ - Builder::~~Unigram_Model_Synchronized_Training_Builder \(\)](#) [virtual]

12.51.2 Member Function Documentation

12.51.2.1 [Execution_Strategy & Unigram_Model_Synchronized_Training_ - Builder::create_execution_strategy \(Pipeline & pipeline\)](#) [virtual]

Reimplemented from [Unigram_Model_Training_Builder](#).

12.51.3 Member Data Documentation

12.51.3.1 Synchronizer_Helper* Unigram_Model_Synchronized_Training_Builder::_sync_helper [protected]

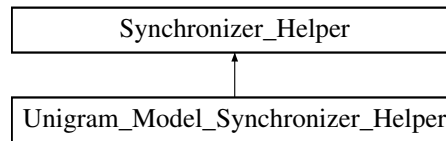
The documentation for this class was generated from the following files:

- [src/Unigram_Model/TopicLearner/Unigram_Model_Synchronized_Training_Builder.h](#)
- [src/Unigram_Model/TopicLearner/Unigram_Model_Synchronized_Training_Builder.cpp](#)

12.52 Unigram_Model_Synchronizer_Helper Class Reference

```
#include <Unigram_Model_Synchronizer_Helper.h>
```

Inheritance diagram for Unigram_Model_Synchronizer_Helper:



Public Member Functions

- [Unigram_Model_Synchronizer_Helper](#) ([TypeTopicCounts](#) &ttc, [WordIndexDictionary](#) &dict)
- virtual [~Unigram_Model_Synchronizer_Helper](#) ()
- void [initialize](#) ()
- bool [has_to_synchronize](#) ()
- void [reset_to_synchronize](#) ()
- void [synchronize](#) ()
- void [end_putNget](#) (const std::string &word, const std::string &counts)

12.52.1 Constructor & Destructor Documentation

12.52.1.1 [Unigram_Model_Synchronizer_Helper::Unigram_Model_Synchronizer_Helper](#) ([TypeTopicCounts](#) & *ttc*, [WordIndexDictionary](#) & *dict*)

12.52.1.2 [Unigram_Model_Synchronizer_Helper::~~Unigram_Model_Synchronizer_Helper](#) () [[virtual](#)]

12.52.2 Member Function Documentation

12.52.2.1 void [Unigram_Model_Synchronizer_Helper::end_putNget](#) (const std::string & *word*, const std::string & *counts*) [[virtual](#)]

This is a callback from the [Client](#) when an [async_putNget](#) is used on the [Client](#). So when a [Client](#) is instantiated, you need to pass a reference of (*this)

Implements [Synchronizer_Helper](#).

**12.52.2.2 bool Unigram_Model_Synchronizer_Helper::has_to_synchronize ()
[virtual]**

Returns true as long as all items to be synchronized are not synchronized

Implements [Synchronizer_Helper](#).

**12.52.2.3 void Unigram_Model_Synchronizer_Helper::initialize ()
[virtual]**

Implements [Synchronizer_Helper](#).

**12.52.2.4 void Unigram_Model_Synchronizer_Helper::reset_to_synchronize ()
[virtual]**

After this call, [has_to_synchronize\(\)](#) should return true

Implements [Synchronizer_Helper](#).

**12.52.2.5 void Unigram_Model_Synchronizer_Helper::synchronize ()
[virtual]**

Implements [Synchronizer_Helper](#).

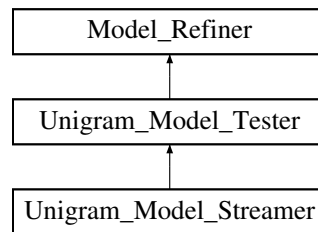
The documentation for this class was generated from the following files:

- [src/Unigram_Model/TopicLearner/Unigram_Model_Synchronizer_Helper.h](#)
- [src/Unigram_Model/TopicLearner/Unigram_Model_Synchronizer_Helper.cpp](#)

12.53 Unigram_Model_Tester Class Reference

```
#include <Unigram_Model_Tester.h>
```

Inheritance diagram for Unigram_Model_Tester:



Public Member Functions

- `Unigram_Model_Tester` (`TypeTopicCounts` &, `Parameter` &, `Parameter` &, `WordIndexDictionary` &, `bool` no_init=false)
- `virtual ~Unigram_Model_Tester` ()
- `google::protobuf::Message *` `allocate_document_buffer` (`size_t`)
- `void deallocate_document_buffer` (`google::protobuf::Message *`)
- `google::protobuf::Message *` `get_nth_document` (`google::protobuf::Message *` docs, `size_t` n)
- `void *` `read` (`google::protobuf::Message &`)
- `void *` `sample` (`void *`)
- `void *` `update` (`void *`)
- `void *` `optimize` (`void *`)
- `void *` `eval` (`void *`, `double &`)
- `void write` (`void *`)
- `void iteration_done` ()
- `void *` `test` (`void *`)

Static Public Attributes

- `static long` `doc_index`

Protected Attributes

- `TypeTopicCounts` & `_ttc`
- `Parameter` & `_alpha`
- `Parameter` & `_beta`

- bool [ignore_old_topic](#)
- int [_num_words](#)
- int [_num_topics](#)
- [DocumentReader](#) * [_wdoc_rdr](#)
- [DocumentWriter](#) * [_tdoc_writer](#)

12.53.1 Constructor & Destructor Documentation

12.53.1.1 [Unigram_Model_Tester::Unigram_Model_Tester](#) ([TypeTopicCounts](#) & *ttc*, [Parameter](#) & *alpha*, [Parameter](#) & *beta*, [WordIndexDictionary](#) & *local_dict*, bool *no_init* = **false**)

12.53.1.2 [Unigram_Model_Tester::~~Unigram_Model_Tester](#) () [**virtual**]

12.53.2 Member Function Documentation

12.53.2.1 [google::protobuf::Message](#) * [Unigram_Model_Tester::allocate_document_buffer](#) ([size_t](#) *num_docs*) [**virtual**]

Implements [Model_Refiner](#).

12.53.2.2 void [Unigram_Model_Tester::deallocate_document_buffer](#) ([google::protobuf::Message](#) * *docs*) [**virtual**]

Implements [Model_Refiner](#).

12.53.2.3 void * [Unigram_Model_Tester::eval](#) (void * *token*, double & *eval_value*) [**virtual**]

Implements [Model_Refiner](#).

12.53.2.4 [google::protobuf::Message](#) * [Unigram_Model_Tester::get_nth_document](#) ([google::protobuf::Message](#) * *docs*, [size_t](#) *n*) [**virtual**]

Implements [Model_Refiner](#).

12.53.2.5 void [Unigram_Model_Tester::iteration_done](#) () [**virtual**]

Implements [Model_Refiner](#).

Reimplemented in [Unigram_Model_Streamers](#).

12.53.2.6 `void * Unigram_Model_Tester::optimize (void *)` **[virtual]**

Implements [Model_Refiner](#).

12.53.2.7 `void * Unigram_Model_Tester::read (google::protobuf::Message & doc)` **[virtual]**

Implements [Model_Refiner](#).

Reimplemented in [Unigram_Model_Streamers](#).

12.53.2.8 `void * Unigram_Model_Tester::sample (void *)` **[virtual]**

Implements [Model_Refiner](#).

12.53.2.9 `void * Unigram_Model_Tester::test (void * token)` **[virtual]**

Implements [Model_Refiner](#).

12.53.2.10 `void * Unigram_Model_Tester::update (void *)` **[virtual]**

Implements [Model_Refiner](#).

12.53.2.11 `void Unigram_Model_Tester::write (void * token)` **[virtual]**

Implements [Model_Refiner](#).

Reimplemented in [Unigram_Model_Streamers](#).

12.53.3 Member Data Documentation

12.53.3.1 `Parameter& Unigram_Model_Tester::_alpha` `[protected]`

12.53.3.2 `Parameter& Unigram_Model_Tester::_beta` `[protected]`

12.53.3.3 `int Unigram_Model_Tester::_num_topics` `[protected]`

12.53.3.4 `int Unigram_Model_Tester::_num_words` `[protected]`

12.53.3.5 `DocumentWriter* Unigram_Model_Tester::_tdoc_writer`
`[protected]`

12.53.3.6 `TypeTopicCounts& Unigram_Model_Tester::_ttc` `[protected]`

12.53.3.7 `DocumentReader* Unigram_Model_Tester::_wdoc_rdr`
`[protected]`

12.53.3.8 `long Unigram_Model_Tester::doc_index` `[static]`

12.53.3.9 `bool Unigram_Model_Tester::ignore_old_topic` `[protected]`

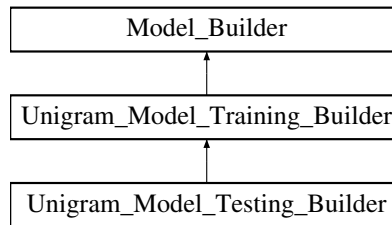
The documentation for this class was generated from the following files:

- `src/Unigram_Model/TopicLearner/Unigram_Model_Tester.h`
- `src/Unigram_Model/TopicLearner/Unigram_Model_Tester.cpp`

12.54 Unigram_Model_Testing_Builder Class Reference

```
#include <Unigram_Model_Testing_Builder.h>
```

Inheritance diagram for Unigram_Model_Testing_Builder:



Public Member Functions

- [Unigram_Model_Testing_Builder \(\)](#)
- virtual [~Unigram_Model_Testing_Builder \(\)](#)
- virtual [Model_Refiner & create_model_refiner \(\)](#)
- virtual [Execution_Strategy & create_execution_strategy \(Pipeline &\)](#)

12.54.1 Constructor & Destructor Documentation

12.54.1.1 [Unigram_Model_Testing_Builder::Unigram_Model_Testing_Builder \(\)](#)

12.54.1.2 [Unigram_Model_Testing_Builder::~~Unigram_Model_Testing_Builder \(\)](#) [**virtual**]

12.54.2 Member Function Documentation

12.54.2.1 [Execution_Strategy & Unigram_Model_Testing_Builder::create_execution_strategy \(Pipeline & pipeline\)](#) [**virtual**]

Reimplemented from [Unigram_Model_Training_Builder](#).

12.54.2.2 [Model_Refiner & Unigram_Model_Testing_Builder::create_model_refiner \(\)](#) [**virtual**]

Reimplemented from [Unigram_Model_Training_Builder](#).

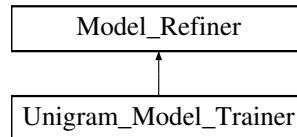
The documentation for this class was generated from the following files:

- [src/Unigram_Model/TopicLearner/Unigram_Model_Testing_Builder.h](#)
- [src/Unigram_Model/TopicLearner/Unigram_Model_Testing_Builder.cpp](#)

12.55 Unigram_Model_Trainer Class Reference

```
#include <Unigram_Model_Trainer.h>
```

Inheritance diagram for Unigram_Model_Trainer:



Public Member Functions

- [Unigram_Model_Trainer](#) ([TypeTopicCounts](#) &, [Parameter](#) &, [Parameter](#) &)
 - virtual [~Unigram_Model_Trainer](#) ()
 - [google::protobuf::Message *](#) [allocate_document_buffer](#) ([size_t](#))
 - void [deallocate_document_buffer](#) ([google::protobuf::Message *](#))
 - [google::protobuf::Message *](#) [get_nth_document](#) ([google::protobuf::Message *](#)
[*docs](#), [size_t](#) [n](#))
 - void * [read](#) ([google::protobuf::Message &](#))
 - void * [sample](#) (void *)
 - void * [update](#) (void *)
 - void * [optimize](#) (void *)
 - void * [eval](#) (void *, [double](#) &)
- Compute the document portion of the log-likelihood.*
- void [write](#) (void *)
 - void [iteration_done](#) ()
 - void * [test](#) (void *)

Static Public Attributes

- static long [doc_index](#) = -1

12.55.1 Detailed Description

The default implementation of [Model_Refiner](#) for the Unigram model

12.55.2 Constructor & Destructor Documentation

12.55.2.1 Unigram_Model_Trainer::Unigram_Model_Trainer
(TypeTopicCounts & *ttc*, Parameter & *alpha*, Parameter & *beta*)

12.55.2.2 Unigram_Model_Trainer::~~Unigram_Model_Trainer ()
[**virtual**]

12.55.3 Member Function Documentation

12.55.3.1 google::protobuf::Message * Unigram_Model_Trainer::allocate_document_buffer (size_t *num_docs*)
[**virtual**]

Implements [Model_Refiner](#).

12.55.3.2 void Unigram_Model_Trainer::deallocate_document_buffer
(google::protobuf::Message * *docs*) [**virtual**]

Implements [Model_Refiner](#).

12.55.3.3 void * Unigram_Model_Trainer::eval (void * *token*, double &
eval_value) [**virtual**]

Compute the document portion of the log-likelihood.

Implements [Model_Refiner](#).

12.55.3.4 google::protobuf::Message * Unigram_Model_Trainer::get_nth_document (google::protobuf::Message * *docs*, size_t *n*)
[**virtual**]

Implements [Model_Refiner](#).

12.55.3.5 void Unigram_Model_Trainer::iteration_done () [**virtual**]

Implements [Model_Refiner](#).

**12.55.3.6 void * Unigram_Model_Trainer::optimize (void * *token*)
[virtual]**

Performs stochastic GD to optimize the alphas. The gradients are accumulated for tau docs and then the global alphas are updated.

Implements [Model_Refiner](#).

12.55.3.7 void * Unigram_Model_Trainer::read (google::protobuf::Message & *doc*) [virtual]

Reads a document from the protobuf format word & topic files using [DocumentReader](#)

Implements [Model_Refiner](#).

12.55.3.8 void * Unigram_Model_Trainer::sample (void * *token*) [virtual]

Does Gibbs sampling using [sampler.cpp](#) to figure out new topic assignments to each word present in the document passed in the msg

Implements [Model_Refiner](#).

12.55.3.9 void * Unigram_Model_Trainer::test (void * *token*) [virtual]

Implements [Model_Refiner](#).

**12.55.3.10 void * Unigram_Model_Trainer::update (void * *token*)
[virtual]**

Takes a msg which contains the document to be processed and the updated topics for each word in the document as a vector. It then processes each update by just calling upd_count on the [TypeTopicCounts](#) object with the update details

Implements [Model_Refiner](#).

12.55.3.11 void Unigram_Model_Trainer::write (void * *token*) [virtual]

Takes the document and writes it to disk. Here we use a simple optimization of not writing the body/words in the document but only the topics. This is because the words in the document never change. Its only the topics that change. The documents are written using a [DocumentWriter](#) to disk

Implements [Model_Refiner](#).

12.55.4 Member Data Documentation

12.55.4.1 `long Unigram_Model_Trainer::doc_index = -1` `[static]`

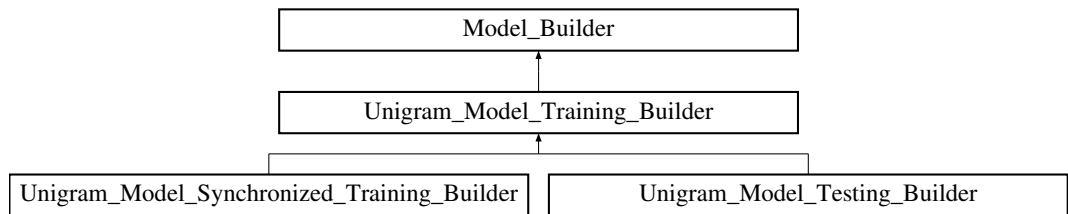
The documentation for this class was generated from the following files:

- `src/Unigram_Model/TopicLearner/Unigram_Model_Trainer.h`
- `src/Unigram_Model/TopicLearner/Unigram_Model_Trainer.cpp`

12.56 Unigram_Model_Training_Builder Class Reference

```
#include <Unigram_Model_Training_Builder.h>
```

Inheritance diagram for Unigram_Model_Training_Builder:



Public Member Functions

- [Unigram_Model_Training_Builder \(\)](#)
- virtual [~Unigram_Model_Training_Builder \(\)](#)
- virtual [Model_Refiner & create_model_refiner \(\)](#)
- virtual [Pipeline & create_pipeline \(Model_Refiner &\)](#)
- virtual [Execution_Strategy & create_execution_strategy \(Pipeline &\)](#)
- void [create_output \(\)](#)
- [Model & get_model \(\)](#)

Protected Member Functions

- void [init_dict \(\)](#)
- [WordIndexDictionary & get_dict \(\)](#)
- void [initialize_topics \(string, string, int\)](#)

Protected Attributes

- [Unigram_Model * _model](#)
- [WordIndexDictionary * _dict](#)
- [Model_Refiner * _refiner](#)
- [Pipeline * _pipeline](#)
- [Checkpointner * _checkpointner](#)
- [Execution_Strategy * _strategy](#)

12.56.1 Constructor & Destructor Documentation

12.56.1.1 Unigram_Model_Training_Builder::Unigram_Model_Training_Builder ()

12.56.1.2 Unigram_Model_Training_Builder::~~Unigram_Model_Training_Builder () [virtual]

12.56.2 Member Function Documentation

12.56.2.1 Execution_Strategy & Unigram_Model_Training_Builder::create_execution_strategy (Pipeline & *pipeline*) [virtual]

Implements [Model_Builder](#).

Reimplemented in [Unigram_Model_Synchronized_Training_Builder](#), and [Unigram_Model_Testing_Builder](#).

12.56.2.2 Model_Refiner & Unigram_Model_Training_Builder::create_model_refiner () [virtual]

Implements [Model_Builder](#).

Reimplemented in [Unigram_Model_Testing_Builder](#).

12.56.2.3 void Unigram_Model_Training_Builder::create_output () [virtual]

Implements [Model_Builder](#).

12.56.2.4 Pipeline & Unigram_Model_Training_Builder::create_pipeline (Model_Refiner & *refiner*) [virtual]

Implements [Model_Builder](#).

12.56.2.5 WordIndexDictionary & Unigram_Model_Training_Builder::get_dict () [protected]

12.56.2.6 Model & Unigram_Model_Training_Builder::get_model () [virtual]

Implements [Model_Builder](#).

12.56.2.7 `void Unigram_Model_Training_Builder::init_dict ()` **[protected]**

12.56.2.8 `void Unigram_Model_Training_Builder::initialize_topics (string input_w, string input_t, int num_topics)` **[protected]**

12.56.3 Member Data Documentation

12.56.3.1 `Checkpointer* Unigram_Model_Training_Builder::_checkpointer`
[protected]

12.56.3.2 `WordIndexDictionary* Unigram_Model_Training_Builder::_dict`
[protected]

12.56.3.3 `Unigram_Model* Unigram_Model_Training_Builder::_model`
[protected]

12.56.3.4 `Pipeline* Unigram_Model_Training_Builder::_pipeline`
[protected]

12.56.3.5 `Model_Refiner* Unigram_Model_Training_Builder::_refiner`
[protected]

12.56.3.6 `Execution_Strategy* Unigram_Model_Training_Builder::_strategy`
[protected]

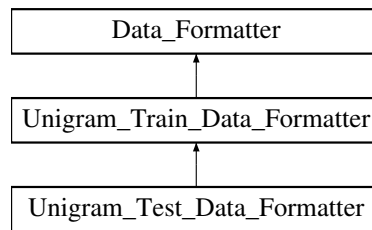
The documentation for this class was generated from the following files:

- `src/Unigram_Model/TopicLearner/Unigram_Model_Training_Builder.h`
- `src/Unigram_Model/TopicLearner/Unigram_Model_Training_Builder.cpp`

12.57 Unigram_Test_Data_Formatter Class Reference

```
#include <Unigram_Test_Data_Formatter.h>
```

Inheritance diagram for Unigram_Test_Data_Formatter:



Public Member Functions

- [Unigram_Test_Data_Formatter \(\)](#)
- virtual [~Unigram_Test_Data_Formatter \(\)](#)

Protected Member Functions

- int [insert_word_to_dict](#) (std::string word)

12.57.1 Constructor & Destructor Documentation

12.57.1.1 [Unigram_Test_Data_Formatter::Unigram_Test_Data_Formatter \(\)](#)

12.57.1.2 [Unigram_Test_Data_Formatter::~~Unigram_Test_Data_Formatter \(\)](#)
[**virtual**]

12.57.2 Member Function Documentation

12.57.2.1 [int Unigram_Test_Data_Formatter::insert_word_to_dict \(std::string word\)](#) [**protected**, **virtual**]

Reimplemented from [Unigram_Train_Data_Formatter](#).

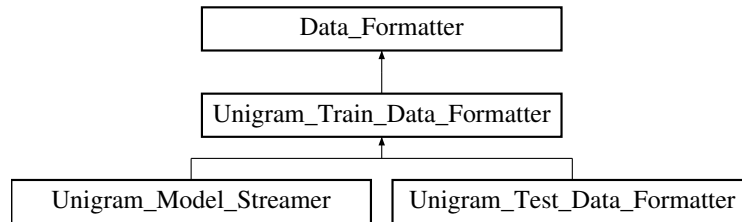
The documentation for this class was generated from the following files:

- [src/Unigram_Model/Formatter/Unigram_Test_Data_Formatter.h](#)
- [src/Unigram_Model/Formatter/Unigram_Test_Data_Formatter.cpp](#)

12.58 Unigram_Train_Data_Formatter Class Reference

```
#include <Unigram_Train_Data_Formatter.h>
```

Inheritance diagram for Unigram_Train_Data_Formatter:



Public Member Functions

- [Unigram_Train_Data_Formatter \(\)](#)
- virtual [~Unigram_Train_Data_Formatter \(\)](#)
- void [format \(\)](#)
Perform the actual formatting.
- [WordIndexDictionary & get_dictionary \(\)](#)
Return the dictionary being used by the formatter.
- int [get_num_docs \(\)](#)
The number of documents formatted.
- int [get_total_num_words \(\)](#)
The total number of words found.

Protected Member Functions

- virtual int [insert_word_to_dict](#) (std::string word)
- int [read_from_inp](#) (LDA::unigram_document &wdoc, std::istream &inp)

Protected Attributes

- [WordIndexDictionary _dict](#)
- int [_num_docs](#)

- [int _num_words_in_all_docs](#)
- [boost::unordered_set< string > _stopWords](#)
- [std::ifstream _in](#)
- [DocumentWriter * _doc_writer](#)

12.58.1 Constructor & Destructor Documentation

12.58.1.1 [Unigram_Train_Data_Formatter::Unigram_Train_Data_Formatter\(\)](#)

12.58.1.2 [Unigram_Train_Data_Formatter::~~Unigram_Train_Data_Formatter\(\)](#) [**virtual**]

12.58.2 Member Function Documentation

12.58.2.1 [void Unigram_Train_Data_Formatter::format\(\)](#) [**virtual**]

Perform the actual formatting.

Implements [Data_Formatter](#).

12.58.2.2 [WordIndexDictionary & Unigram_Train_Data_Formatter::get_dictionary\(\)](#) [**virtual**]

Return the dictionary being used by the formatter.

Implements [Data_Formatter](#).

12.58.2.3 [int Unigram_Train_Data_Formatter::get_num_docs\(\)](#) [**virtual**]

The number of documents formatted.

Implements [Data_Formatter](#).

12.58.2.4 [int Unigram_Train_Data_Formatter::get_total_num_words\(\)](#) [**virtual**]

The total number of words found.

Implements [Data_Formatter](#).

12.58.2.5 `virtual int Unigram_Train_Data_Formatter::insert_word_to_dict
(std::string word) [protected, virtual]`

Reimplemented in [Unigram_Test_Data_Formatter](#), and [Unigram_Model_Streammer](#).

12.58.2.6 `int Unigram_Train_Data_Formatter::read_from_inp
(LDA::unigram_document & wdoc, std::istream & inp)
[protected]`

12.58.3 Member Data Documentation

12.58.3.1 `WordIndexDictionary Unigram_Train_Data_Formatter::_dict
[protected]`

12.58.3.2 `DocumentWriter* Unigram_Train_Data_Formatter::_doc_writer
[protected]`

12.58.3.3 `std::ifstream Unigram_Train_Data_Formatter::_in [protected]`

12.58.3.4 `int Unigram_Train_Data_Formatter::_num_docs [protected]`

12.58.3.5 `int Unigram_Train_Data_Formatter::_num_words_in_all_docs
[protected]`

12.58.3.6 `boost::unordered_set<string> Unigram_Train_Data_Formatter::_
stopWords [protected]`

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

- [src/Unigram_Model/Formatter/Unigram_Train_Data_Formatter.h](#)
- [src/Unigram_Model/Formatter/Unigram_Train_Data_Formatter.cpp](#)

12.59 WordIndexDictionary Class Reference

A two way dictionary of words to indices.

```
#include <WordIndexDictionary.h>
```

Public Member Functions

- [WordIndexDictionary \(\)](#)
- virtual [~WordIndexDictionary \(\)](#)
- int [get_index](#) (string word)
- string [get_word](#) (int index)
- int [insert_word](#) (string word)
- int [get_num_words](#) () const
- void [print](#) ()
- bool [match_word_index](#) ()
- void [dump](#) (string fname)
- void [initialize_from_dict](#) (WordIndexDictionary *dict, bool sort=false)
- void [initialize_from_dump](#) (string fname, int num_words=INT_MAX, bool sort=false)
- void [initialize_from_dumps](#) (string prefix, int dumps)
- size_t [size](#) ()
- int [get_prev_index](#) (int new_id)
- int [get_freq](#) (int index)

Public Attributes

- vector< id2freq_t > [frequencies](#)

12.59.1 Detailed Description

A two way dictionary of words to indices. Provides a two way dictionary mapping words as strings to a unique int index and vice versa. The hashtable implementation of boost/unordered_map is used.

12.59.2 Constructor & Destructor Documentation

12.59.2.1 WordIndexDictionary::WordIndexDictionary ()

Constructs an empty dictionary

12.59.2.2 WordIndexDictionary::~~WordIndexDictionary () [virtual]**12.59.3 Member Function Documentation****12.59.3.1 void WordIndexDictionary::dump (string *fname*)**

Dumps the dictionary onto disk in protobuf binary format so that a new dictionary can be initialized later from the dump using `initialize_from_dump`. Also the dump does batch write to disk to optimize io. Batches 1000 (word,index) pairs and then writes them to disk using [DocumentWriter](#)

12.59.3.2 int WordIndexDictionary::get_freq (int *index*)**12.59.3.3 int WordIndexDictionary::get_index (string *word*)**

Find the unique index assigned to word

12.59.3.4 int WordIndexDictionary::get_num_words () const**12.59.3.5 int WordIndexDictionary::get_prev_index (int *new_id*)****12.59.3.6 string WordIndexDictionary::get_word (int *index*)**

Find the word having index as its index

**12.59.3.7 void WordIndexDictionary::initialize_from_dict
(WordIndexDictionary * *dict*, bool *sort* = false)****12.59.3.8 void WordIndexDictionary::initialize_from_dump (string *fname*, int
num_words = INT_MAX, bool *sort* = false)**

Initializes from a dump file produced by `dump`. Reads the (word,index) pairs from the file & populates the maps

**12.59.3.9 void WordIndexDictionary::initialize_from_dumps (string *prefix*, int
dumps)****12.59.3.10 int WordIndexDictionary::insert_word (string *word*)**

Insert the word into the dictionary if it doesn't exist. This automatically manages assigning unique indices

12.59.3.11 bool WordIndexDictionary::match_word_index ()

This is a method aiding testing. This tests for the uniqueness of indices. It also does this by making the assumption that the indices have to sequential and reduces the complexity of testing uniqueness by checking the actual sum of the indices assigned to the expected sum computed as $\sigma(\text{last_index_assigned})$. Should always return true. If you change the logic for assigning unique indices, make sure you modify this method to verify it.

12.59.3.12 void WordIndexDictionary::print ()

Log the dictionary to log(INFO)

12.59.3.13 size_t WordIndexDictionary::size ()**12.59.4 Member Data Documentation****12.59.4.1 vector<id2freq_t> WordIndexDictionary::frequencies**

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

- [src/commons/WordIndexDictionary.h](#)
- [src/commons/WordIndexDictionary.cpp](#)

Chapter 13

File Documentation

13.1 `src/architecture.h` File Reference

13.2 src/commons/Client.h File Reference

```
#include <string>
```

Classes

- class [Client](#)

13.3 src/commons/comparator.cpp File Reference

```
#include "types.h"
#include <vector>
#include "tbb/spin_rw_mutex.h"
#include <string>
#include <google/protobuf/stubs/common.h>
#include <google/protobuf/generated_message_util.h>
#include <google/protobuf/repeated_field.h>
#include <google/protobuf/extension_set.h>
#include <google/protobuf/generated_message_reflection.h>
#include "tbb/tick_count.h"
#include <sstream>
#include <boost/random/mersenne_twister.hpp>
#include "glog/logging.h"
#include "boost/unordered_map.hpp"
#include "tbb/atomic.h"
```

Functions

- bool [cnt_cmp](#) (packed_t i, packed_t j)
- bool [cnt_cmp_ttc](#) (cnt_topic_t i, cnt_topic_t j)
- bool [prob_cmp](#) (tppair v1, tppair v2)
- bool [freq_cmp](#) (id2freq_t v1, id2freq_t v2)

13.3.1 Function Documentation

13.3.1.1 bool [cnt_cmp](#) (packed_t i, packed_t j)

13.3.1.2 bool [cnt_cmp_ttc](#) (cnt_topic_t i, cnt_topic_t j)

13.3.1.3 bool [freq_cmp](#) (id2freq_t v1, id2freq_t v2)

13.3.1.4 bool [prob_cmp](#) (tppair v1, tppair v2)

13.4 src/commons/comparator.h File Reference

Functions

- bool [cnt_cmp](#) (packed_t i, packed_t j)
- bool [cnt_cmp_ttc](#) (cnt_topic_t i, cnt_topic_t j)
- bool [prob_cmp](#) (tppair v1, tppair v2)
- bool [freq_cmp](#) (id2freq_t v1, id2freq_t v2)

13.4.1 Function Documentation

13.4.1.1 bool [cnt_cmp](#) (packed_t *i*, packed_t *j*)

13.4.1.2 bool [cnt_cmp_ttc](#) (cnt_topic_t *i*, cnt_topic_t *j*)

13.4.1.3 bool [freq_cmp](#) (id2freq_t *v1*, id2freq_t *v2*)

13.4.1.4 bool [prob_cmp](#) (tppair *v1*, tppair *v2*)

13.5 src/commons/constants.h File Reference

```
#include "types.h"
```

13.6 src/commons/Context.cpp File Reference

```
#include "Context.h"  
#include "boost/unordered_map.hpp"  
#include <string>  
#include "gflags/gflags.h"  
#include <vector>  
#include <cstdlib>  
#include "glog/logging.h"
```

13.7 src/commons/Context.h File Reference

```
#include "boost/unordered_map.hpp"  
#include <string>
```

Classes

- class [Context](#)
An object that maintains the context for the executing code.

13.8 src/commons/document.pb.cc File Reference

```
#include "document.pb.h"
#include <google/protobuf/stubs/once.h>
#include <google/protobuf/io/coded_stream.h>
#include <google/protobuf/wire_format_lite_inl.h>
#include <google/protobuf/descriptor.h>
#include <google/protobuf/reflection_ops.h>
#include <google/protobuf/wire_format.h>
```

Defines

- #define [INTERNAL_SUPPRESS_PROTOBUF_FIELD_DEPRECATION](#)
- #define [DO_\(EXPRESSION\)](#) if (!(EXPRESSION)) return false
- #define [DO_\(EXPRESSION\)](#) if (!(EXPRESSION)) return false
- #define [DO_\(EXPRESSION\)](#) if (!(EXPRESSION)) return false
- #define [DO_\(EXPRESSION\)](#) if (!(EXPRESSION)) return false
- #define [DO_\(EXPRESSION\)](#) if (!(EXPRESSION)) return false

13.8.1 Define Documentation

13.8.1.1 #define [DO_\(EXPRESSION\)](#) if (!(EXPRESSION)) return false

13.8.1.2 #define [DO_\(EXPRESSION\)](#) if (!(EXPRESSION)) return false

13.8.1.3 #define [DO_\(EXPRESSION\)](#) if (!(EXPRESSION)) return false

13.8.1.4 #define [DO_\(EXPRESSION\)](#) if (!(EXPRESSION)) return false

13.8.1.5 #define [DO_\(EXPRESSION\)](#) if (!(EXPRESSION)) return false

13.8.1.6 #define [INTERNAL_SUPPRESS_PROTOBUF_FIELD_DEPRECATION](#)

13.9 src/commons/document.pb.h File Reference

```
#include <string>
#include <google/protobuf/stubs/common.h>
#include <google/protobuf/generated_message_util.h>
#include <google/protobuf/repeated_field.h>
#include <google/protobuf/extension_set.h>
#include <google/protobuf/generated_message_-\
reflection.h>
```

13.10 src/commons/DocumentReader.cpp File Reference

```
#include "DocumentReader.h"  
#include <fstream>  
#include "google/protobuf/message.h"  
#include "types.h"
```

13.11 src/commons/DocumentReader.h File Reference

```
#include <fstream>
#include "google/protobuf/message.h"
#include "constants.h"
```

Classes

- class [DocumentReader](#)

13.12 src/commons/DocumentWriter.cpp File Reference

```
#include "DocumentWriter.h"  
#include <fstream>  
#include "google/protobuf/message.h"  
#include "constants.h"
```

13.13 src/commons/DocumentWriter.h File Reference

```
#include <fstream>
#include "google/protobuf/message.h"
#include "constants.h"
```

Classes

- class [DocumentWriter](#)

13.14 src/commons/Formatter/Controller.cpp File Reference

```
#include "FormatData_flags_define.h"
#include "gflags/gflags.h"
#include "glog/logging.h"
#include <stdlib.h>
#include "WordIndexDictionary.h"
#include "Formatter/Unigram_Train_Data_Formatter.h"
#include "Formatter/Unigram_Test_Data_Formatter.h"
#include "Context.h"
#include "TopicLearner/Model.h"
```

Functions

- [Data_Formatter](#) * [get_data_formatter](#) ()
- void [release_data_formatter](#) ([Data_Formatter](#) *formatter)
- int [main](#) (int argc, char *argv[])

13.14.1 Function Documentation

13.14.1.1 Data_Formatter* get_data_formatter ()

Returns the formatter for the chosen model

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

Main

13.14.1.3 void release_data_formatter (Data_Formatter *formatter)

Deallocate resources held by the formatter

13.15 src/commons/TopicLearner/Controller.cpp File Reference

```
#include "Main_flags_define.h"
#include "gflags/gflags.h"
#include "constants.h"
#include "tbb/task_scheduler_init.h"
#include "glog/logging.h"
#include "TopicLearner/Unigram_Model_Training_Builder.h"
#include "TopicLearner/Unigram_Model_Synchronized_
Training_Builder.h"
#include "TopicLearner/Unigram_Model_Testing_Builder.h"
#include "TopicLearner/Unigram_Model_Streaming_Builder.h"
#include "google/protobuf/message.h"
#include "TopicLearner/Model_Refiner.h"
#include "WordIndexDictionary.h"
#include "Model.h"
#include "DocumentReader.h"
#include "document.pb.h"
```

Functions

- int [main](#) (int argc, char *argv[])

13.15.1 Function Documentation

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

13.16 src/commons/Formatter/Data_Formatter.h File Reference

```
#include "WordIndexDictionary.h"
```

Classes

- class [Data_Formatter](#)
An interface for formatter objects.

13.17 src/commons/Formatter/FormatData_flags_define.h File Reference

```
#include "gflags/gflags.h"
```

Functions

- [DEFINE_int32](#) (model, 1, "Unigram-1 or some other model")
- [DEFINE_string](#) (corpus, "specify", "A text file with space separated fields containing doc-id, category, contents of the doc")
- [DEFINE_string](#) (outputprefix, "lda", "A prefix that will be used with all files output by the program")
- [DEFINE_string](#) (dumpfile, "specify", "The dump of the dictionary to be used instead of creating afresh")

13.17.1 Function Documentation

13.17.1.1 [DEFINE_int32](#) (model, 1, "Unigram-1 or some other model")

13.17.1.2 [DEFINE_string](#) (dumpfile, "specify", "The dump of the dictionary to be used instead of creating afresh")

13.17.1.3 [DEFINE_string](#) (outputprefix, "lda", "A prefix that will be used with all files output by the program")

13.17.1.4 [DEFINE_string](#) (corpus, "specify", "A text file with space separated fields containing doc- *id*, category, contents of the doc")

13.18 src/commons/LDAUtil.cpp File Reference

```
#include "LDAUtil.h"
#include <vector>
#include <string>
#include <sstream>
#include <algorithm>
#include <iterator>
#include <iostream>
```

Namespaces

- namespace [LDAUtil](#)

13.19 src/commons/LDAUtil.h File Reference

```
#include <vector>
#include <string>
#include <sstream>
```

Classes

- class [LDAUtil::StringTokenizer](#)
- class [LDAUtil::StringTrimmer](#)
- class [LDAUtil::Itoa](#)
- class [LDAUtil::DM_Server_Names](#)

Namespaces

- namespace [LDAUtil](#)

13.20 src/commons/Server/DistributedMap.cpp File Reference

```
#include <DistributedMap.h>
#include <Ice/LocalException.h>
#include <Ice/ObjectFactory.h>
#include <Ice/BasicStream.h>
#include <IceUtil/Iterator.h>
```

13.21 src/commons/Server/DistributedMap.h File Reference

```
#include <Ice/LocalObjectF.h>
#include <Ice/ProxyF.h>
#include <Ice/ObjectF.h>
#include <Ice/Exception.h>
#include <Ice/LocalObject.h>
#include <Ice/Proxy.h>
#include <Ice/Object.h>
#include <Ice/Outgoing.h>
#include <Ice/OutgoingAsync.h>
#include <Ice/Incoming.h>
#include <Ice/IncomingAsync.h>
#include <Ice/Direct.h>
#include <IceUtil/ScopedArray.h>
#include <Ice/StreamF.h>
#include <Ice/UndefSysMacros.h>
```

13.22 src/commons/Server/DM_Server.cpp File Reference

```
#include "DM_Server.h"
#include <Ice/Ice.h>
#include <Ice/LocalObjectF.h>
#include <Ice/ProxyF.h>
#include <Ice/ObjectF.h>
#include <Ice/Exception.h>
#include <Ice/LocalObject.h>
#include <Ice/Proxy.h>
#include <Ice/Object.h>
#include <Ice/Outgoing.h>
#include <Ice/OutgoingAsync.h>
#include <Ice/Incoming.h>
#include <Ice/IncomingAsync.h>
#include <Ice/Direct.h>
#include <IceUtil/ScopedArray.h>
#include <Ice/StreamF.h>
#include <Ice/UndefSysMacros.h>
#include "boost/unordered_map.hpp"
#include "types.h"
#include "tbb/spin_rw_mutex.h"
#include <IceUtil/Mutex.h>
#include <IceUtil/Handle.h>
#include <list>
#include <string>
#include "LDAUtil.h"
#include "tbb/atomic.h"
#include "TopicLearner/Model.h"
#include "Server/Unigram_Model_Server_Helper.h"
```

Functions

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

13.22.1 Function Documentation

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

13.23 src/commons/Server/DM_Server.h File Reference

```
#include <Ice/Ice.h>
#include "DistributedMap.h"
#include "Hashmap_Array.h"
#include <IceUtil/Mutex.h>
#include <IceUtil/Handle.h>
#include <list>
#include "Server_Helper.h"
```

Classes

- class [PNGJob](#)
- class [DM_Server](#)

The Server class that implements the DistributedMap Ice interface.

Typedefs

- typedef IceUtil::Handle< [PNGJob](#) > [PNGJobPtr](#)

Variables

- const int [QUE_FULL](#) = 50
- const int [NUM_CONSUMERS](#) = 3
- const int [CHUNK](#) = 10

13.23.1 Typedef Documentation

13.23.1.1 `typedef IceUtil::Handle<PNGJob> PNGJobPtr`

13.23.2 Variable Documentation

13.23.2.1 `const int CHUNK = 10`

13.23.2.2 `const int NUM_CONSUMERS = 3`

13.23.2.3 `const int QUE_FULL = 50`

13.24 src/commons/Server/Hashmap_Array.h File Reference

```
#include "boost/unordered_map.hpp"
#include "types.h"
#include "tbb/spin_rw_mutex.h"
```

Classes

- class [Hashmap_Array< Key, Value >](#)
- class [Hashmap_Array< Key, Value >::iterator](#)

13.25 src/commons/Server/Server_Helper.h File Reference

```
#include <string>
```

Classes

- class [Server_Helper](#)
Server Helper interface.

13.26 src/commons/TopicLearner/Checkpoint.h File Reference

```
#include <string>
```

Classes

- class [Checkpoint](#)
Used to implement failure recovery.

13.27 src/commons/TopicLearner/Dirichlet.cpp File Reference

```
#include <cmath>
#include "Dirichlet.h"
#include "constants.h"
```

Functions

- double [log_gamma](#) (double z)
- double [digamma](#) (double z)

13.27.1 Function Documentation

13.27.1.1 double digamma (double z)

13.27.1.2 double log_gamma (double z)

13.28 src/commons/TopicLearner/Dirichlet.h File Reference

```
#include "constants.h"
```

Functions

- double [log_gamma](#) (double z)
- double [digamma](#) (double z)

13.28.1 Function Documentation

13.28.1.1 double [digamma](#) (double z)

13.28.1.2 double [log_gamma](#) (double z)

13.29 src/commons/TopicLearner/DM_Client.cpp File Reference

```
#include "DM_Client.h"
#include <Ice/Ice.h>
#include <vector>
#include "Server/DistributedMap.h"
#include <IceUtil/Handle.h>
#include "TopicLearner/Synchronizer_Helper.h"
#include "Client.h"
#include "boost/unordered_map.hpp"
#include "constants.h"
#include "LDAUtil.h"
#include <algorithm>
```

13.30 src/commons/TopicLearner/DM_Client.h File Reference

```
#include <Ice/Ice.h>
#include <vector>
#include "Server/DistributedMap.h"
#include <IceUtil/Handle.h>
#include "TopicLearner/Synchronizer_Helper.h"
#include "Client.h"
#include "boost/unordered_map.hpp"
#include "constants.h"
```

Classes

- struct [Cookie](#)
- class [PNGCallback](#)

The call back object for the Put and Get AML.

- class [DM_Client](#)

The client used to access the Distributed Map.

Typedefs

- typedef IceUtil::Handle< [Cookie](#) > [CookiePtr](#)
- typedef IceUtil::Handle< [PNGCallback](#) > [PNGCallbackPtr](#)

13.30.1 Typedef Documentation

13.30.1.1 typedef IceUtil::Handle<Cookie> CookiePtr

13.30.1.2 typedef IceUtil::Handle<PNGCallback> PNGCallbackPtr

13.31 src/commons/TopicLearner/Execution_Strategy.h File Reference

Classes

- class [Execution_Strategy](#)
Interface for strategy objects.

13.32 src/commons/TopicLearner/Filter_Eval.cpp File Reference

```
#include "Filter_Eval.h"  
#include "tbb/pipeline.h"  
#include "Model_Refiner.h"
```

13.33 src/commons/TopicLearner/Filter_Eval.h File Reference

```
#include "tbb/pipeline.h"  
#include "Model_Refiner.h"
```

Classes

- class [Filter_Eval](#)
A filter in the TBB pipeline.

13.34 src/commons/TopicLearner/Filter_ Optimizer.cpp File Reference

```
#include "Filter_Optimizer.h"  
#include "tbb/pipeline.h"  
#include "Model_Refiner.h"
```

13.35 src/commons/TopicLearner/Filter_Optimizer.h File Reference

```
#include "tbb/pipeline.h"  
#include "Model_Refiner.h"
```

Classes

- class [Filter_Optimizer](#)
A filter in the TBB pipeline.

13.36 src/commons/TopicLearner/Filter_Reader.cpp

File Reference

```
#include "Filter_Reader.h"
#include "tbb/pipeline.h"
#include "Model_Refiner.h"
#include "google/protobuf/message.h"
#include "Context.h"
```

13.37 src/commons/TopicLearner/Filter_Reader.h File Reference

```
#include "tbb/pipeline.h"
#include "Model_Refiner.h"
#include "google/protobuf/message.h"
```

Classes

- class [Filter_Reader](#)
A filter in the TBB pipeline.

13.38 src/commons/TopicLearner/Filter_Sampler.cpp

File Reference

```
#include "Filter_Sampler.h"  
#include "tbb/pipeline.h"  
#include "Model_Refiner.h"
```


13.39 src/commons/TopicLearner/Filter_Sampler.h File Reference

```
#include "tbb/pipeline.h"  
#include "Model_Refiner.h"
```

Classes

- class [Filter_Sampler](#)
A filter in the TBB pipeline.

13.40 src/commons/TopicLearner/Filter_Tester.cpp

File Reference

```
#include "Filter_Tester.h"  
#include "tbb/pipeline.h"  
#include "Model_Refiner.h"
```

13.41 src/commons/TopicLearner/Filter_Tester.h File Reference

```
#include "tbb/pipeline.h"  
#include "Model_Refiner.h"
```

Classes

- class [Filter_Tester](#)
A filter in the TBB pipeline.

13.42 src/commons/TopicLearner/Filter_Updater.cpp

File Reference

```
#include "Filter_Updater.h"  
#include "tbb/pipeline.h"  
#include "Model_Refiner.h"
```

13.43 src/commons/TopicLearner/Filter_Updater.h File Reference

```
#include "tbb/pipeline.h"  
#include "Model_Refiner.h"
```

Classes

- class [Filter_Updater](#)

13.44 src/commons/TopicLearner/Filter_Writer.cpp

File Reference

```
#include "Filter_Writer.h"  
#include "tbb/pipeline.h"  
#include "Model_Refiner.h"
```

13.45 src/commons/TopicLearner/Filter_Writer.h File Reference

```
#include "tbb/pipeline.h"  
#include "Model_Refiner.h"
```

Classes

- class [Filter_Writer](#)
A filter in the TBB pipeline.

13.46 src/commons/TopicLearner/GenericTopKList.h File Reference

```
#include <queue>
#include <stack>
#include <fstream>
```

Classes

- class [GenericTopKList< T, GreaterThan >](#)
A list that maintains top K elements.

13.47 src/commons/TopicLearner/Main_flags_define.h File Reference

```
#include "gflags/gflags.h"
#include "constants.h"
#include "tbb/task_scheduler_init.h"
```

Functions

- [DEFINE_int32](#) (iter, 1000,"Number of iterations the topic modeller should be run")
- [DEFINE_int32](#) (burnin, 299,"Number of iterations after which alpha optimization should be to be run after every <optimizestats> iterations")
- [DEFINE_int32](#) (optimizestats, 25,"Optimize hyper parameters every these many iterations")
- [DEFINE_int32](#) (printloglikelihood, 25,"Print log likelihood after every <printlogLikelihood> iterations after burn-in")
- [DEFINE_int32](#) (topics, 100,"The number of topics to be used by LDA.")
- [DEFINE_string](#) (inputprefix,"lda","The output prefix used for the FormatData routine")
- [DEFINE_string](#) (dumpprefix,"","The word-topic counts are initialized from this file which is generated by the preprocessing step or at the end of an iteration")
- [DEFINE_bool](#) (restart, false,"Indicates use of failure recovery mode. The iteration to start with should also be specified")
- [DEFINE_bool](#) (online, false,"Uses online initialization instead of random")
- [DEFINE_int32](#) (startiter, 1,"This the iteration at which failure recovery should start")
- [DEFINE_bool](#) (test, false,"Run the test pipeline. No updates are done & requires an earlier dump of the word-topic counts table")
- [DEFINE_bool](#) (teststream, false,"Run the test pipeline in streaming mode. Formatting is a part of the pipeline. No updates are done & requires an earlier dump of the word-topic counts table & dictionary")
- [DEFINE_double](#) (alpha, ALPHA_SUM,"Weight of the Dirichlet conjugate for topics")
- [DEFINE_double](#) (beta, BETA,"Weight of the Dirichlet conjugate for words")
- [DEFINE_int32](#) (chkptinterval, 25,"The topic assignments are saved every these many iterations")
- [DEFINE_string](#) (chkptdir,"","The directory to which the checkpoints need to written")
- [DEFINE_string](#) (servers,"specify","The set of all memcached servers that are storing the state. E.g. 192.168.0.1, 192.168.0.3:44, 200.132.12.34")

- [DEFINE_int32](#) (numdumps, 1, "Number of word-topic count dumps in the training data")
- [DEFINE_int32](#) (maxmemory, 2048, "The max memory that can be used")
- [DEFINE_string](#) (dictionary, "specify", "The dump of the global dictionary produced in the training run. To be use for teststream")
- [DEFINE_int32](#) (livetokens, 500, "Max Live Tokens in pipeline")
- [DEFINE_int32](#) (model, 1, "Unigram-1")
- [DEFINE_int32](#) (samplerthreads, tbb::task_scheduler_init::automatic, "The number of foreground threads that run actual LDA pipeline. Default is to figure out automatically")

13.47.1 Function Documentation

- 13.47.1.1 **DEFINE_bool** (teststream, false, "Run the test pipeline in streaming mode. Formatting is a part of the pipeline. No updates are done & requires an earlier dump of the word-topic counts table & dictionary")
- 13.47.1.2 **DEFINE_bool** (test, false, "Run the test pipeline. No updates are done & requires an earlier dump of the word-topic counts table")
- 13.47.1.3 **DEFINE_bool** (online, false, "Uses online initialization instead of random")
- 13.47.1.4 **DEFINE_bool** (restart, false, "Indicates use of failure recovery mode. The iteration to start with should also be specified")
- 13.47.1.5 **DEFINE_double** (beta, BETA, "Weight of the Dirichlet conjugate for words")
- 13.47.1.6 **DEFINE_double** (alpha, ALPHA_SUM, "Weight of the Dirichlet conjugate for topics")
- 13.47.1.7 **DEFINE_int32** (samplerthreads, tbb::task_scheduler_init::automatic, "The number of foreground threads that run actual LDA pipeline. Default is to figure out automatically")
- 13.47.1.8 **DEFINE_int32** (model, 1, "Unigram-1")
- 13.47.1.9 **DEFINE_int32** (livetokens, 500, "Max Live Tokens in pipeline")
- 13.47.1.10 **DEFINE_int32** (maxmemory, 2048, "The max memory that can be used")
- 13.47.1.11 **DEFINE_int32** (numdumps, 1, "Number of word-topic count dumps in the training data")
- 13.47.1.12 **DEFINE_int32** (chkptinterval, 25, "The topic assignments are saved every these many iterations")
- 13.47.1.13 **DEFINE_int32** (starttiter, 1, "This the iteration at which failure recovery should start")
- 13.47.1.14 **DEFINE_int32** (topics, 100, "The number of topics to be used by LDA.")
- 13.47.1.15 **DEFINE_int32** (printloglikelihood, 25, "Print log likelihood after every <printlogLikelihood> iterations after burn-in")
- 13.47.1.16 **DEFINE_int32** (optimizestats, 25, "Optimize hyper parameters every these many iterations")
- 13.47.1.17 **DEFINE_int32** (burnin, 299, "Number of iterations after which alpha optimization should be to be run after every <optimizestats> iterations")

13.48 src/commons/TopicLearner/Model.h File Reference

```
#include "WordIndexDictionary.h"
```

Classes

- class [Model](#)

13.49 src/commons/TopicLearner/Model_Builder.h

File Reference

```
#include "Model_Refiner.h"
#include "Pipeline.h"
#include "Execution_Strategy.h"
#include "Model.h"
```

Classes

- class [Model_Builder](#)

13.50 src/commons/TopicLearner/Model_Director.cpp File Reference

```
#include "Model_Director.h"  
#include "Model_Builder.h"
```

13.51 src/commons/TopicLearner/Model_Director.h

File Reference

```
#include "Model_Builder.h"  
#include "Model.h"
```

Classes

- class [Model_Director](#)

The Director class of the Builder pattern.

13.52 src/commons/TopicLearner/Model_Refiner.h File Reference

```
#include "google/protobuf/message.h"
```

Classes

- class [Model_Refiner](#)

13.53 src/commons/TopicLearner/Parameter.cpp File Reference

```
#include "Parameter.h"
#include <string>
#include "DocumentReader.h"
#include "DocumentWriter.h"
#include "document.pb.h"
```

13.54 src/commons/TopicLearner/Parameter.h File Reference

```
#include <string>
```

Classes

- struct [Parameter](#)

Typedefs

- typedef struct [Parameter](#) param

13.54.1 Typedef Documentation

13.54.1.1 typedef struct [Parameter](#) param

A class to represent the various parameters like the dirichlet conjugate weights and such

Can be vector valued parameters

This class additionally stores the sum and provides functionality to dump to and initialize from disk

13.55 src/commons/TopicLearner/Pipeline.h File Reference

```
#include "TopicLearner/Model_Refiner.h"
```

Classes

- class [Pipeline](#)

An interface that all pipeline objects must implement.

13.56

src/commons/TopicLearner/Synchronized_Training_Execution_Strategy.cpp File
Reference

235

13.56 src/commons/TopicLearner/Synchronized_ Training_Execution_Strategy.cpp File Reference

```
#include "Synchronized_Training_Execution_Strategy.h"  
#include "Execution_Strategy.h"  
#include "Pipeline.h"  
#include "Model.h"  
#include <string>  
#include "Synchronizer_Helper.h"  
#include <pthread.h>  
#include <glog/logging.h>
```

Functions

- void * [synchronize](#) (void *inp)

13.56.1 Function Documentation

13.56.1.1 void* synchronize (void * *inp*)

13.57 src/commons/TopicLearner/Synchronized_ Training_Execution_Strategy.h File Reference

```
#include "Training_Execution_Strategy.h"  
#include "Synchronizer_Helper.h"
```

Classes

- class [Synchronized_Training_Execution_Strategy](#)
Default implementation of the [Execution_Strategy](#) interface.

13.58 src/commons/TopicLearner/Synchronizer.cpp File Reference

```
#include "Synchronizer.h"  
#include "tbb/tick_count.h"  
#include "types.h"  
#include "glog/logging.h"
```

13.59 src/commons/TopicLearner/Synchronizer.h File Reference

```
#include "Synchronizer_Helper.h"
```

Classes

- class [Synchronizer](#)

13.60 src/commons/TopicLearner/Synchronizer_Helper.h File Reference

```
#include <string>
```

Classes

- class [Synchronizer_Helper](#)
A helper class for the synchronizer.

13.61 src/commons/TopicLearner/TBB_Pipeline.cpp

File Reference

```
#include "TBB_Pipeline.h"
#include "Pipeline.h"
#include "tbb/task_scheduler_init.h"
#include "tbb/pipeline.h"
#include "Model_Refiner.h"
#include "Filter_Reader.h"
#include "Filter_Sampler.h"
#include "Filter_Updater.h"
#include "Filter_Optimizer.h"
#include "Filter_Eval.h"
#include "Filter_Writer.h"
#include "Filter_Tester.h"
#include "Context.h"
```

13.62 src/commons/TopicLearner/TBB_Pipeline.h File Reference

```
#include "Pipeline.h"
#include "tbb/task_scheduler_init.h"
#include "tbb/pipeline.h"
#include "Model_Refiner.h"
```

Classes

- class [TBB_Pipeline](#)

13.63 src/commons/TopicLearner/Testing_Execution_Strategy.cpp File Reference

```
#include "Testing_Execution_Strategy.h"
#include "Execution_Strategy.h"
#include "Pipeline.h"
#include "Model.h"
#include "Context.h"
#include "tbb/tick_count.h"
#include "glog/logging.h"
```

13.64 src/commons/TopicLearner/Testing_Execution_Strategy.h File Reference

```
#include "Execution_Strategy.h"
#include "Pipeline.h"
#include "Model.h"
```

Classes

- class [Testing_Execution_Strategy](#)

13.65 src/commons/TopicLearner/Training_Execution_Strategy.cpp File Reference

```
#include "Training_Execution_Strategy.h"
#include "Context.h"
#include "tbb/tick_count.h"
#include "glog/logging.h"
```

13.66 src/commons/TopicLearner/Training_Execution_Strategy.h File Reference

```
#include "Execution_Strategy.h"
#include "Pipeline.h"
#include "Model.h"
#include "Checkpoint.h"
```

Classes

- class [Training_Execution_Strategy](#)

13.67 src/commons/types.h File Reference

```
#include <vector>
#include "tbb/spin_rw_mutex.h"
#include "document.pb.h"
#include "tbb/tick_count.h"
#include <sstream>
#include <boost/random/mercenne_twister.hpp>
#include "glog/logging.h"
#include "boost/unordered_map.hpp"
#include "tbb/atomic.h"
```

Classes

- class [InvalidOldTopicExc](#)

Defines

- #define [TIME](#)(t) tick_count t = tick_count::now()
- #define [PRINT_TIME](#)(t1, t2, str) cout << "Time taken to " #str << " : " << (t2-t1).seconds() << " secs" << endl

Typedefs

- typedef int32_t [size_int](#)
- typedef boost::mt19937 [base_generator_type](#)
- typedef std::pair< topic_t, float > [tppair](#)
- typedef std::pair< word_t, float > [wppair](#)
- typedef std::pair< bigram_key_t, float > [bigppair](#)
- typedef boost::unordered_map< topic_t, cnt_t > [mapped_vec](#)

13.67.1 Define Documentation

13.67.1.1 `#define PRINT_TIME(t1, t2, str) cout << "Time taken to " #str << " : " << (t2-t1).seconds() << " secs" << endl`

13.67.1.2 `#define TIME(t) tick_count t = tick_count::now()`

13.67.2 Typedef Documentation

13.67.2.1 `typedef boost::mt19937 base_generator_type`

13.67.2.2 `typedef std::pair<bigram_key_t, float> bigppair`

13.67.2.3 `typedef boost::unordered_map<topic_t, cnt_t> mapped_vec`

13.67.2.4 `typedef int32_t size_int`

13.67.2.5 `typedef std::pair<topic_t, float> tppair`

13.67.2.6 `typedef std::pair<word_t, float> wppair`

13.68 src/commons/WordIndexDictionary.cpp File Reference

```
#include "WordIndexDictionary.h"
#include "boost/unordered_map.hpp"
#include "types.h"
#include "DocumentReader.h"
#include <climits>
#include <iostream>
#include "document.pb.h"
#include "DocumentWriter.h"
#include "glog/logging.h"
#include <sstream>
```

13.69 src/commons/WordIndexDictionary.h File Reference

```
#include "boost/unordered_map.hpp"
#include "types.h"
#include "DocumentReader.h"
#include <climits>
```

Classes

- class [WordIndexDictionary](#)
A two way dictionary of words to indices.

13.70 src/mainpage.h File Reference

13.71 src/multi_mcahine_usage.h File Reference

13.72 src/single_machine_usage.h File Reference

13.73 src/Unigram_Model/Formatter/Unigram_Test_Data_Formatter.cpp File Reference

```
#include "Unigram_Test_Data_Formatter.h"
#include "commons/Formatter/Data_Formatter.h"
#include "commons/DocumentWriter.h"
#include <fstream>
#include "boost/unordered_set.hpp"
#include "commons/Context.h"
```

13.74 src/Unigram_Model/Formatter/Unigram_Test_Data_Formatter.h File Reference

```
#include "Unigram_Train_Data_Formatter.h"
```

Classes

- class [Unigram_Test_Data_Formatter](#)

13.75 src/Unigram_Model/Formatter/Unigram_Train_Data_Formatter.cpp File Reference

```
#include "Unigram_Train_Data_Formatter.h"
#include "glog/logging.h"
#include "commons/Context.h"
#include "commons/document.pb.h"
#include "commons/constants.h"
```

13.76 src/Unigram_Model/Formatter/Unigram_Train_Data_Formatter.h File Reference

```
#include "commons/Formatter/Data_Formatter.h"
#include "commons/DocumentWriter.h"
#include <fstream>
#include "boost/unordered_set.hpp"
```

Classes

- class [Unigram_Train_Data_Formatter](#)

13.77 src/Unigram_Model/Merge/Merge_Dictionaries.cpp File Reference

```
#include "gflags/gflags.h"
#include "glog/logging.h"
#include "WordIndexDictionary.h"
```

Functions

- [DEFINE_string](#) (dumpprefix,"specify","The dump of the dictionary to be used instead of creating afresh")
- [DEFINE_int32](#) (dictionaries,-1,"The number of dictionaries present")
- [DEFINE_string](#) (outputprefix,"lda","A prefix that will be used with all files output by the program")
- [int main](#) (int argc, char *argv[])

13.77.1 Function Documentation

13.77.1.1 [DEFINE_int32](#) (dictionaries, - 1, "The number of dictionaries present")

13.77.1.2 [DEFINE_string](#) (outputprefix, "lda", "A prefix that will be used with all files output by the program")

13.77.1.3 [DEFINE_string](#) (dumpprefix, "specify", "The dump of the dictionary to be used instead of creating afresh")

13.77.1.4 [int main](#) (int *argc*, char * *argv*[])

13.78 src/Unigram_Model/Merge/Merge_Topic_Counts.cpp File Reference

```
#include "gflags/gflags.h"
#include "glog/logging.h"
#include "WordIndexDictionary.h"
#include "TopicLearner/TypeTopicCounts.h"
#include "TopicLearner/Synchronizer_Helper.h"
#include "TopicLearner/DM_Client.h"
```

Functions

- [DEFINE_int32](#) (clientid,-1,"The id that this client should use and means that this is now a part of the multi-machine setup")
- [DEFINE_string](#) (servers,"specify","The set of all memcached servers that are storing the state. E.g. 192.168.0.1, 192.168.0.3:44, 200.132.12.34")
- [DEFINE_string](#) (outputprefix,"lda","A prefix that will be used with all files output by the program")
- [DEFINE_string](#) (globaldictionary,"specify","The global dictionary; topic counts of parts of which have to be retrieved")
- [DEFINE_int32](#) (topics, 100,"The number of topics to be used by LDA.")
- [int main](#) (int argc, char *argv[])

13.78.1 Function Documentation

- 13.78.1.1 **DEFINE_int32** (topics, 100, "The number of topics to be used by LDA.")
- 13.78.1.2 **DEFINE_int32** (clientid, -1, "The id that this client should use and means that this is now a part of the multi-machine setup")
- 13.78.1.3 **DEFINE_string** (globaldictionary, "specify", "The global dictionary; topic counts of parts of which have to be retrieved")
- 13.78.1.4 **DEFINE_string** (outputprefix, "lda", "A prefix that will be used with all files output by the program")
- 13.78.1.5 **DEFINE_string** (servers, "specify", "The set of all memcached servers that are storing the state. E.g. 192.168.0.1, 192.168.0.3:44, 200.132.12.34")
- 13.78.1.6 **int main** (int *argc*, char * *argv* [])

13.79 src/Unigram_Model/Server/Unigram_Model_Server_Helper.cpp File Reference

```
#include "Unigram_Model_Server_Helper.h"  
#include "Server/Server_Helper.h"  
#include "types.h"  
#include "Unigram_Model/TopicLearner/TopicCounts.h"
```

13.80 src/Unigram_Model/Server/Unigram_Model_ Server_Helper.h File Reference

```
#include "Server/Server_Helper.h"  
#include "types.h"
```

Classes

- class [Unigram_Model_Server_Helper](#)

13.81 src/Unigram_Model/TopicLearner/eff_small_map.cpp File Reference

```
#include "eff_small_map.h"
#include "types.h"
#include <algorithm>
#include "boost/unordered_map.hpp"
#include "tbb/atomic.h"
#include "comparator.h"
#include <vector>
```


13.82 src/Unigram_Model/TopicLearner/eff_small_map.h File Reference

```
#include "TopicCounts.h"
```

Classes

- class [simple_map](#)

13.83 src/Unigram_Model/TopicLearner/Hadoop_- Checkpoint.cpp File Reference

```
#include "Hadoop_Checkpointer.h"  
#include "TopicLearner/Checkpoint.h"  
#include "Context.h"  
#include "glog/logging.h"
```

13.84 src/Unigram_Model/TopicLearner/Hadoop_Checkpointer.h File Reference

```
#include "Local_Checkpointer.h"
```

Classes

- class [Hadoop_Checkpointer](#)

13.85 src/Unigram_Model/TopicLearner/Local_- Checkpointner.cpp File Reference

```
#include "Context.h"  
#include "Local_Checkpointer.h"  
#include <fstream>  
#include <sstream>
```

13.86 src/Unigram_Model/TopicLearner/Local_- Checkpoint.h File Reference

```
#include "TopicLearner/Checkpoint.h"
```

Classes

- class [Local_Checkpointer](#)

13.87 src/Unigram_Model/TopicLearner/sampler.cpp

File Reference

```
#include <algorithm>
#include <iostream>
#include "sampler.h"
#include "constants.h"
#include <boost/random/variante_generator.hpp>
#include <boost/random/uniform_real.hpp>
#include "tbb/atomic.h"
#include "TopicCounts.h"
#include "Context.h"
```

Namespaces

- namespace [sampler](#)

Functions

- `topic_t sampler::sample` (const [topicCounts](#) *currentTypeTopicCounts, topic_t old_topic, const atomic< topic_t > *tokens_per_topic, const topic_t *local_topic_counts, const topic_t *local_topic_index, const int &non_zero_topics, const double &smoothingOnlyMass, const double &topicBetaMass, const double *cachedCoefficients, const double &betaSum, const double *alpha, double *topic_term_scores, const topic_t &numTopics, variate_generator< [base_generator_type](#) &, boost::uniform_real<> > *unif01)

13.88 src/Unigram_Model/TopicLearner/sampler.h File Reference

```
#include "constants.h"
#include <boost/random/variator_generator.hpp>
#include <boost/random/uniform_real.hpp>
#include "tbb/atomic.h"
#include "TopicCounts.h"
```

Namespaces

- namespace [sampler](#)

Functions

- `topic_t sampler::sample` (const [topicCounts](#) *currentTypeTopicCounts, topic_t old_topic, const atomic< topic_t > *tokens_per_topic, const topic_t *local_topic_counts, const topic_t *local_topic_index, const int &non_zero_topics, const double &smoothingOnlyMass, const double &topicBetaMass, const double *cachedCoefficients, const double &betaSum, const double *alpha, double *topic_term_scores, const topic_t &numTopics, variator_generator< [base_generator_type](#) &, boost::uniform_real<> > *unif01)

13.89 src/Unigram_Model/TopicLearner/TopicCounts.cpp

File Reference

```
#include "TopicCounts.h"  
#include "eff_small_map.h"
```


13.90 src/Unigram_Model/TopicLearner/TopicCounts.h File Reference

```
#include "types.h"
#include <algorithm>
#include "boost/unordered_map.hpp"
#include "tbb/atomic.h"
#include "comparator.h"
#include <vector>
```

Classes

- struct [TopicCounts](#)

Typedefs

- typedef struct [TopicCounts](#) [topicCounts](#)

13.90.1 Typedef Documentation

13.90.1.1 typedef struct TopicCounts topicCounts

13.91 src/Unigram_Model/TopicLearner/TopKList.cpp

File Reference

```
#include "TopKList.h"  
#include "constants.h"  
#include <algorithm>  
#include "comparator.h"  
#include <climits>  
#include <iostream>
```

13.92 src/Unigram_Model/TopicLearner/TopKList.h File Reference

```
#include "constants.h"  
#include <algorithm>  
#include "comparator.h"
```

Classes

- class [TopKList](#)

13.93 src/Unigram_Model/TopicLearner/TypeTopicCounts.cpp

File Reference

```
#include <algorithm>
#include "TypeTopicCounts.h"
#include <iostream>
#include <exception>
#include <stdio.h>
#include "TopKList.h"
#include "tbb/atomic.h"
#include "TopicCounts.h"
#include "WordIndexDictionary.h"
#include "DocumentReader.h"
#include "DocumentWriter.h"
#include "TopicLearner/Dirichlet.h"
#include "Context.h"
```

Functions

- bool [cnt_cmp](#) (packed_t *i*, packed_t *j*)
- bool [cnt_cmp_ttc](#) (cnt_topic_t *i*, cnt_topic_t *j*)

13.93.1 Function Documentation

13.93.1.1 bool [cnt_cmp](#) (packed_t *i*, packed_t *j*)

13.93.1.2 bool [cnt_cmp_ttc](#) (cnt_topic_t *i*, cnt_topic_t *j*)

13.94 src/Unigram_Model/TopicLearner/TypeTopicCounts.h File Reference

```
#include <iostream>
#include <exception>
#include <stdio.h>
#include "TopKList.h"
#include "tbb/atomic.h"
#include "TopicCounts.h"
#include "WordIndexDictionary.h"
```

Classes

- class [TypeTopicCounts](#)

13.95 src/Unigram_Model/TopicLearner/Unigram_Model.cpp File Reference

```
#include "Unigram_Model.h"
#include "TopicLearner/Model.h"
#include <string>
#include "TopicLearner/Parameter.h"
#include "TypeTopicCounts.h"
#include "TopicLearner/GenericTopKList.h"
#include "Context.h"
#include "glog/logging.h"
#include "TopicLearner/Dirichlet.h"
```

13.96 src/Unigram_Model/TopicLearner/Unigram_Model.h File Reference

```
#include "TopicLearner/Model.h"
#include <string>
#include "TopicLearner/Parameter.h"
#include "TypeTopicCounts.h"
#include "TopicLearner/GenericTopKList.h"
```

Classes

- class [Unigram_Model](#)

13.97 src/Unigram_Model/TopicLearner/Unigram_Model_Streamers.cpp File Reference

```
#include "Unigram_Model_Streamers.h"
#include "Formatter/Unigram_Train_Data_Formatter.h"
#include "TopicLearner/Model_Refiner.h"
#include "TypeTopicCounts.h"
#include "TopicLearner/Parameter.h"
#include "DocumentReader.h"
#include "DocumentWriter.h"
#include <boost/random/variante_generator.hpp>
#include <boost/random/uniform_real.hpp>
#include "WordIndexDictionary.h"
#include "Context.h"
#include "glog/logging.h"
#include "document.pb.h"
```


13.98 src/Unigram_Model/TopicLearner/Unigram_ - Model_Streamer.h File Reference

```
#include "Formatter/Unigram_Train_Data_Formatter.h"  
#include "Unigram_Model_Tester.h"
```

Classes

- class [Unigram_Model_Streamer](#)

13.99 src/Unigram_Model/TopicLearner/Unigram_Model_Streaming_Builder.cpp File Reference

```
#include "Unigram_Model_Streaming_Builder.h"
#include "TopicLearner/Model_Builder.h"
#include "Unigram_Model.h"
#include "WordIndexDictionary.h"
#include "Context.h"
#include "Unigram_Model_Streamer.h"
#include "TopicLearner/TBB_Pipeline.h"
#include "TopicLearner/Testing_Execution_Strategy.h"
```

13.100 src/Unigram_Model/TopicLearner/Unigram_Model_Streaming_Builder.h File Reference

```
#include "TopicLearner/Model_Builder.h"  
#include "Unigram_Model.h"  
#include "WordIndexDictionary.h"
```

Classes

- class [Unigram_Model_Streaming_Builder](#)

13.101 src/Unigram_Model/TopicLearner/Unigram - Model_Synchronized_Training_Builder.cpp File Reference

```
#include "Unigram_Model_Synchronized_Training_Builder.h"
#include "TopicLearner/Model_Builder.h"
#include "Unigram_Model.h"
#include "WordIndexDictionary.h"
#include "TopicLearner/Checkpointter.h"
#include "TopicLearner/Synchronizer_Helper.h"
#include "TypeTopicCounts.h"
#include "Client.h"
#include "TopicLearner/Synchronized_Training_Execution_ -
Strategy.h"
#include "Hadoop_Checkpointer.h"
#include "Context.h"
```

13.102 src/Unigram_Model/TopicLearner/Unigram_Model_Synchronized_Training_Builder.h File

Reference ²⁸³
13.102 src/Unigram_Model/TopicLearner/Unigram_Model_Synchronized_Training_Builder.h File Reference

```
#include "Unigram_Model_Training_Builder.h"  
#include "TopicLearner/Synchronizer_Helper.h"
```

Classes

- class [Unigram_Model_Synchronized_Training_Builder](#)

13.103 src/Unigram_Model/TopicLearner/Unigram_- Model_Synchronizer_Helper.cpp File Refer- ence

```
#include "Unigram_Model_Synchronizer_Helper.h"  
#include "Context.h"  
#include "TopicLearner/DM_Client.h"
```

13.104

src/Unigram_Model/TopicLearner/Unigram_Model_Synchronizer_Helper.h File
Reference

13.104 src/Unigram_Model/TopicLearner/Unigram_Model_Synchronizer_Helper.h File Reference

```
#include "TopicLearner/Synchronizer_Helper.h"
#include "TypeTopicCounts.h"
#include "WordIndexDictionary.h"
#include "Client.h"
```

Classes

- class [Unigram_Model_Synchronizer_Helper](#)

13.105 src/Unigram_Model/TopicLearner/Unigram_Model_Tester.cpp File Reference

```
#include "Unigram_Model_Tester.h"
#include "document.pb.h"
#include "sampler.h"
#include "Context.h"
#include "glog/logging.h"
#include "TopicLearner/Dirichlet.h"
```


13.106 src/Unigram_Model/TopicLearner/Unigram_ Model_Tester.h File Reference

```
#include "TopicLearner/Model_Refiner.h"
#include "TypeTopicCounts.h"
#include "TopicLearner/Parameter.h"
#include "DocumentReader.h"
#include "DocumentWriter.h"
#include <boost/random/variante_generator.hpp>
#include <boost/random/uniform_real.hpp>
#include "WordIndexDictionary.h"
```

Classes

- class [Unigram_Model_Tester](#)

13.107 src/Unigram_Model/TopicLearner/Unigram_Model_Testing_Builder.cpp File Reference

```
#include "Unigram_Model_Testing_Builder.h"
#include "Unigram_Model_Training_Builder.h"
#include "Context.h"
#include "DocumentReader.h"
#include "DocumentWriter.h"
#include "Unigram_Model_Tester.h"
#include "TopicLearner/TBB_Pipeline.h"
#include "TopicLearner/Testing_Execution_Strategy.h"
```

13.108 src/Unigram_Model/TopicLearner/Unigram_Model_Testing_Builder.h File Reference

```
#include "Unigram_Model_Training_Builder.h"
```

Classes

- class [Unigram_Model_Testing_Builder](#)

13.109 src/Unigram_Model/TopicLearner/Unigram_Model_Trainer.cpp File Reference

```
#include "Unigram_Model_Trainer.h"
#include "TopicLearner/Model_Refiner.h"
#include "TypeTopicCounts.h"
#include "TopicLearner/Parameter.h"
#include "DocumentReader.h"
#include "DocumentWriter.h"
#include <boost/random/variante_generator.hpp>
#include <boost/random/uniform_real.hpp>
#include "document.pb.h"
#include "sampler.h"
#include "Context.h"
#include "glog/logging.h"
#include "TopicLearner/Dirichlet.h"
```

13.110 src/Unigram_Model/TopicLearner/Unigram_ Model_Trainer.h File Reference

```
#include "TopicLearner/Model_Refiner.h"
#include "TypeTopicCounts.h"
#include "TopicLearner/Parameter.h"
#include "DocumentReader.h"
#include "DocumentWriter.h"
#include <boost/random/variante_generator.hpp>
#include <boost/random/uniform_real.hpp>
```

Classes

- class [Unigram_Model_Trainer](#)

13.111 src/Unigram_Model/TopicLearner/Unigram - Model_Training_Builder.cpp File Reference

```
#include "Unigram_Model_Training_Builder.h"
#include "Context.h"
#include "DocumentReader.h"
#include "DocumentWriter.h"
#include "Unigram_Model_Trainer.h"
#include "TopicLearner/TBB_Pipeline.h"
#include "TopicLearner/Training_Execution_Strategy.h"
#include "Local_Checkpointer.h"
```

13.112 src/Unigram_Model/TopicLearner/Unigram_Model_Training_Builder.h File Reference

```
#include "TopicLearner/Model_Builder.h"
#include "Unigram_Model.h"
#include "WordIndexDictionary.h"
#include "TopicLearner/Checkpoint.h"
```

Classes

- class [Unigram_Model_Training_Builder](#)

13.113 `src/usage.h` File Reference

Index

- ~DM_Client
 - DM_Client, [67](#)
- ~DM_Server
 - DM_Server, [70](#)
- ~DocumentReader
 - DocumentReader, [73](#)
- ~DocumentWriter
 - DocumentWriter, [74](#)
- ~Filter_Eval
 - Filter_Eval, [76](#)
- ~Filter_Optimizer
 - Filter_Optimizer, [77](#)
- ~Filter_Reader
 - Filter_Reader, [78](#)
- ~Filter_Sampler
 - Filter_Sampler, [79](#)
- ~Filter_Tester
 - Filter_Tester, [80](#)
- ~Filter_Updater
 - Filter_Updater, [81](#)
- ~Filter_Writer
 - Filter_Writer, [82](#)
- ~GenericTopKList
 - GenericTopKList, [84](#)
- ~Hadoop_Checkpointer
 - Hadoop_Checkpointer, [85](#)
- ~HashMap_Array
 - HashMap_Array, [87](#)
- ~Local_Checkpointer
 - Local_Checkpointer, [92](#)
- ~Model_Director
 - Model_Director, [98](#)
- ~PNGCallback
 - PNGCallback, [108](#)
- ~Parameter
 - Parameter, [103](#)
- ~Synchronized_Training_Execution_Strategy
 - Synchronized_Training_Execution_Strategy, [116](#)
- ~Synchronizer
 - Synchronizer, [117](#)
- ~TBB_Pipeline
 - TBB_Pipeline, [122](#)
- ~Testing_Execution_Strategy
 - Testing_Execution_Strategy, [125](#)
- ~TopKList
 - TopKList, [132](#)
- ~TopicCounts
 - TopicCounts, [127](#)
- ~Training_Execution_Strategy
 - Training_Execution_Strategy, [134](#)
- ~TypeTopicCounts
 - TypeTopicCounts, [137](#)
- ~Unigram_Model
 - Unigram_Model, [143](#)
- ~Unigram_Model_Server_Helper
 - Unigram_Model_Server_Helper, [144](#)
- ~Unigram_Model_Streammer
 - Unigram_Model_Streammer, [146](#)
- ~Unigram_Model_Streaming_Builder
 - Unigram_Model_Streaming_Builder, [148](#)
- ~Unigram_Model_Synchronized_Training_Builder
 - Unigram_Model_Synchronized_Training_Builder, [150](#)
- ~Unigram_Model_Synchronizer_Helper
 - Unigram_Model_Synchronizer_Helper, [152](#)
- ~Unigram_Model_Tester
 - Unigram_Model_Tester, [155](#)

- ~Unigram_Model_Testing_Builder
 - Unigram_Model_Testing_Builder, 158
- ~Unigram_Model_Trainer
 - Unigram_Model_Trainer, 161
- ~Unigram_Model_Training_Builder
 - Unigram_Model_Training_Builder, 165
- ~Unigram_Test_Data_Formatter
 - Unigram_Test_Data_Formatter, 167
- ~Unigram_Train_Data_Formatter
 - Unigram_Train_Data_Formatter, 169
- ~WordIndexDictionary
 - WordIndexDictionary, 171
- ~simple_map
 - simple_map, 112
- _alpha
 - Unigram_Model_Test, 157
- _beta
 - Unigram_Model_Test, 157
- _checkpoint
 - Unigram_Model_Training_Builder, 166
- _dict
 - Unigram_Model_Streaming_Builder, 149
 - Unigram_Model_Training_Builder, 166
 - Unigram_Train_Data_Formatter, 170
- _doc_writer
 - Unigram_Train_Data_Formatter, 170
- _eval
 - TBB_Pipeline, 124
- _global_dict
 - Unigram_Model_Streaming_Builder, 149
- _in
 - Unigram_Train_Data_Formatter, 170
- _init
 - TBB_Pipeline, 124
- _model
 - Unigram_Model_Streaming_Builder, 149
 - Unigram_Model_Training_Builder, 166
- _num_docs
 - Unigram_Train_Data_Formatter, 170
- _num_topics
 - Unigram_Model_Test, 157
- _num_words
 - Unigram_Model_Test, 157
- _num_words_in_all_docs
 - Unigram_Train_Data_Formatter, 170
- _optimizer
 - TBB_Pipeline, 124
- _pipeline
 - TBB_Pipeline, 124
 - Unigram_Model_Streaming_Builder, 149
 - Unigram_Model_Training_Builder, 166
- _reader
 - TBB_Pipeline, 124
- _refiner
 - TBB_Pipeline, 124
 - Unigram_Model_Streaming_Builder, 149
 - Unigram_Model_Training_Builder, 166
- _sampler
 - TBB_Pipeline, 124
- _stopWords
 - Unigram_Train_Data_Formatter, 170
- _strategy
 - Unigram_Model_Streaming_Builder, 149
 - Unigram_Model_Training_Builder, 166
- _sync_helper
 - Unigram_Model_Synchronized_Training_Builder, 151
- _tdoc_writer
 - Unigram_Model_Test, 157
- _tester
 - Unigram_Model_Test, 157

- TBB_Pipeline, 124
- _ttc
 - Unigram_Model_Tester, 157
- _updater
 - TBB_Pipeline, 124
- _wdoc_rdr
 - Unigram_Model_Tester, 157
- _writer
 - TBB_Pipeline, 124
- act_map
 - Hashmap_Array, 87
- act_map_iter
 - Hashmap_Array, 87
- add_eval
 - Pipeline, 104
 - TBB_Pipeline, 122
- add_optimizer
 - Pipeline, 104
 - TBB_Pipeline, 122
- add_reader
 - Pipeline, 105
 - TBB_Pipeline, 122
- add_sampler
 - Pipeline, 105
 - TBB_Pipeline, 122
- add_tester
 - Pipeline, 105
 - TBB_Pipeline, 122
- add_updater
 - Pipeline, 105
 - TBB_Pipeline, 122
- add_writer
 - Pipeline, 105
 - TBB_Pipeline, 122
- addNewTop
 - TopicCounts, 127
- addNewTopAftChk
 - TopicCounts, 127
- allocate_document_buffer
 - Model_Refiner, 100
 - Unigram_Model_Tester, 155
 - Unigram_Model_Trainer, 161
- ALPHA
 - Unigram_Model, 143
- assign
 - TopicCounts, 128
- base_generator_type
 - types.h, 247
- begin
 - Hashmap_Array, 87
- begin_putNget
 - Client, 59
 - DM_Client, 67
- BETA
 - Unigram_Model, 143
- bigppair
 - types.h, 247
- build_model
 - Model_Director, 98
- checkpoint
 - Checkpointter, 58
 - Hadoop_Checkpointter, 85
 - Local_Checkpointter, 92
- Checkpointter, 57
 - checkpoint, 58
 - load_metadata, 58
 - save_metadata, 58
- CHUNK
 - DM_Server.h, 199
- clear
 - GenericTopKList, 84
 - Pipeline, 105
 - simple_map, 112
 - TBB_Pipeline, 123
 - TopKList, 132
- clear_stats
 - TypeTopicCounts, 137
- Client, 59
 - begin_putNget, 59
 - get, 59
 - put, 60
 - remove, 60
 - set, 60
 - wait_for_all, 60
 - wait_till_done, 60
- cnt_cmp
 - comparator.cpp, 177
 - comparator.h, 178
 - TypeTopicCounts.cpp, 274

- cnt_cmp_ttc
 - comparator.cpp, 177
 - comparator.h, 178
 - TypeTopicCounts.cpp, 274
- combine
 - Server_Helper, 111
 - Unigram_Model_Server_Helper, 144
- compact
 - TopicCounts, 128
- comparator.cpp
 - cnt_cmp, 177
 - cnt_cmp_ttc, 177
 - freq_cmp, 177
 - prob_cmp, 177
- comparator.h
 - cnt_cmp, 178
 - cnt_cmp_ttc, 178
 - freq_cmp, 178
 - prob_cmp, 178
- Context, 61
 - get_bool, 61
 - get_double, 61
 - get_instance, 62
 - get_int, 62
 - get_string, 62
 - put_string, 62
- convertTo
 - TopicCounts, 128
- convertTo_d
 - TopicCounts, 128
- Cookie, 63
 - entity, 63
 - msg_id, 63
- CookiePtr
 - DM_Client.h, 206
- count
 - HashMap_Array, 87
- create_execution_strategy
 - Model_Builder, 96
 - Unigram_Model_Streaming_Builder, 148
 - Unigram_Model_Synchronized_Training_Builder, 150
 - Unigram_Model_Testing_Builder, 158
 - Unigram_Model_Training_Builder, 165
- create_model_refiner
 - Model_Builder, 96
 - Unigram_Model_Streaming_Builder, 148
 - Unigram_Model_Testing_Builder, 158
 - Unigram_Model_Training_Builder, 165
- create_output
 - Model_Builder, 97
 - Unigram_Model_Streaming_Builder, 148
 - Unigram_Model_Training_Builder, 165
- create_pipeline
 - Model_Builder, 97
 - Unigram_Model_Streaming_Builder, 148
 - Unigram_Model_Training_Builder, 165
- current_iter
 - HashMap_Array::iterator, 89
- Data_Formatter, 64
 - format, 64
 - get_dictionary, 65
 - get_num_docs, 65
 - get_total_num_words, 65
- deallocate_document_buffer
 - Model_Refiner, 100
 - Unigram_Model_Tester, 155
 - Unigram_Model_Trainer, 161
- decrement
 - TopicCounts, 128
- DEFINE_bool
 - Main_flags_define.h, 226
- DEFINE_double
 - Main_flags_define.h, 226
- DEFINE_int32
 - FormatData_flags_define.h, 191
 - Main_flags_define.h, 226
 - Merge_Dictionaries.cpp, 257
 - Merge_Topic_Counts.cpp, 259
- DEFINE_string

- FormatData_flags_define.h, [191](#)
- Main_flags_define.h, [226](#)
- Merge_Dictionaries.cpp, [257](#)
- Merge_Topic_Counts.cpp, [259](#)
- delta
 - PNGJob, [109](#)
- destroy
 - Pipeline, [105](#)
 - TBB_Pipeline, [123](#)
 - TypeTopicCounts, [137](#)
- digamma
 - Dirichlet.cpp, [203](#)
 - Dirichlet.h, [204](#)
- Dirichlet.cpp
 - digamma, [203](#)
 - log_gamma, [203](#)
- Dirichlet.h
 - digamma, [204](#)
 - log_gamma, [204](#)
- DM_Client, [66](#)
 - ~DM_Client, [67](#)
 - begin_putNget, [67](#)
 - DM_Client, [67](#)
 - DM_Client, [67](#)
 - get, [67](#)
 - get_num_servers, [67](#)
 - put, [67](#)
 - remove, [67](#)
 - set, [67](#)
 - wait_for_all, [68](#)
 - wait_till_done, [68](#)
- DM_Client.h
 - CookiePtr, [206](#)
 - PNGCallbackPtr, [206](#)
- DM_Server, [69](#)
 - ~DM_Server, [70](#)
 - DM_Server, [70](#)
 - DM_Server, [70](#)
 - get, [70](#)
 - put, [70](#)
 - putNget_async, [70](#)
 - remove, [70](#)
 - set, [71](#)
 - waitForAllClients_async, [71](#)
- DM_Server.cpp
 - main, [197](#)
- DM_Server.h
 - CHUNK, [199](#)
 - NUM_CONSUMERS, [199](#)
 - PNGJobPtr, [199](#)
 - QUE_FULL, [199](#)
- DO_
 - document.pb.cc, [182](#)
- doc_index
 - Unigram_Model_Tester, [157](#)
 - Unigram_Model_Trainer, [163](#)
- document.pb.cc
 - DO_, [182](#)
 - INTERNAL_SUPPRESS_
PROTOBUF_FIELD_
DEPRECATION, [182](#)
- DocumentReader, [73](#)
 - ~DocumentReader, [73](#)
 - DocumentReader, [73](#)
 - read, [73](#)
- DocumentWriter, [74](#)
 - ~DocumentWriter, [74](#)
 - DocumentWriter, [74](#)
 - write, [74](#)
- dump
 - Parameter, [103](#)
 - TypeTopicCounts, [138](#)
 - WordIndexDictionary, [172](#)
- empty
 - GenericTopKList, [84](#)
- end
 - HashMap_Array, [87](#)
- end_putNget
 - Synchronizer_Helper, [119](#)
 - Unigram_Model_Synchronizer_
Helper, [152](#)
- entity
 - Cookie, [63](#)
- equal
 - TopicCounts, [128](#)
 - TypeTopicCounts, [138](#)
- erase
 - HashMap_Array, [87](#)
- estimate_alphas
 - TypeTopicCounts, [138](#)
- estimate_fit

- TypeTopicCounts, 138
- estimate_memoryn_warn
 - TypeTopicCounts, 138
- eval
 - Model_Refiner, 100
 - Unigram_Model_Tester, 155
 - Unigram_Model_Trainer, 161
- execute
 - Execution_Strategy, 75
 - Synchronized_Training_ -
 - Execution_Strategy, 116
 - Testing_Execution_Strategy, 125
 - Training_Execution_Strategy, 134
- Execution_Strategy, 75
 - execute, 75
- Filter_Eval, 76
 - ~Filter_Eval, 76
 - Filter_Eval, 76
 - Filter_Eval, 76
 - get_eval, 76
 - operator(), 76
- Filter_Optimizer, 77
 - ~Filter_Optimizer, 77
 - Filter_Optimizer, 77
 - Filter_Optimizer, 77
 - operator(), 77
- Filter_Reader, 78
 - ~Filter_Reader, 78
 - Filter_Reader, 78
 - Filter_Reader, 78
 - operator(), 78
- Filter_Sampler, 79
 - ~Filter_Sampler, 79
 - Filter_Sampler, 79
 - Filter_Sampler, 79
 - operator(), 79
- Filter_Tester, 80
 - ~Filter_Tester, 80
 - Filter_Tester, 80
 - Filter_Tester, 80
 - operator(), 80
- Filter_Updater, 81
 - ~Filter_Updater, 81
 - Filter_Updater, 81
 - Filter_Updater, 81
- operator(), 81
- Filter_Writer, 82
 - ~Filter_Writer, 82
 - Filter_Writer, 82
 - Filter_Writer, 82
 - operator(), 82
- findAndDecrement
 - TopicCounts, 129
- findAndIncrement
 - TopicCounts, 129
- findOldnNew
 - TopicCounts, 129
- finished
 - PNGCallback, 108
- format
 - Data_Formatter, 64
 - Unigram_Train_Data_Formatter, 169
- FormatData_flags_define.h
 - DEFINE_int32, 191
 - DEFINE_string, 191
- Formatter/Controller.cpp
 - get_data_formatter, 188
 - main, 188
 - release_data_formatter, 188
- freq_cmp
 - comparator.cpp, 177
 - comparator.h, 178
- frequencies
 - WordIndexDictionary, 173
- frequency
 - TopicCounts, 131
- GenericTopKList, 83
 - ~GenericTopKList, 84
 - clear, 84
 - empty, 84
 - GenericTopKList, 84
 - pop, 84
 - print, 84
 - push, 84
 - top, 84
- get
 - Client, 59
 - DM_Client, 67
 - DM_Server, 70

- simple_map, 112
- get_beg
 - TopKList, 132
- get_bool
 - Context, 61
- get_counts
 - TopicCounts, 129
 - TypeTopicCounts, 138
- get_data_formatter
 - Formatter/Controller.cpp, 188
- get_dict
 - Unigram_Model_Streaming_
Builder, 148
 - Unigram_Model_Training_Builder,
165
- get_dictionary
 - Data_Formatter, 65
 - Unigram_Train_Data_Formatter,
169
- get_double
 - Context, 61
- get_end
 - TopKList, 132
- get_eval
 - Filter_Eval, 76
 - Model, 94
 - Pipeline, 105
 - TBB_Pipeline, 123
 - Unigram_Model, 143
- get_freq
 - WordIndexDictionary, 172
- get_frequency
 - TopicCounts, 129
- get_index
 - WordIndexDictionary, 172
- get_instance
 - Context, 62
- get_int
 - Context, 62
- get_lock
 - HashMap_Array, 87
 - TypeTopicCounts, 138
- get_max
 - TopKList, 133
- get_model
 - Model_Builder, 97
 - Unigram_Model_Streaming_
Builder, 148
 - Unigram_Model_Training_Builder,
165
- get_nth_document
 - Model_Refiner, 100
 - Unigram_Model_Tester, 155
 - Unigram_Model_Trainer, 161
- get_num_docs
 - Data_Formatter, 65
 - Unigram_Train_Data_Formatter,
169
- get_num_servers
 - DM_Client, 67
- get_num_topics
 - TypeTopicCounts, 138
- get_num_words
 - TypeTopicCounts, 139
 - WordIndexDictionary, 172
- get_parameter
 - Unigram_Model, 143
- get_prev_index
 - WordIndexDictionary, 172
- get_refiner
 - Pipeline, 105
 - TBB_Pipeline, 123
- get_servant_name
 - LDAUtil::DM_Server_Names, 72
- get_server_endpoint
 - LDAUtil::DM_Server_Names, 72
- get_string
 - Context, 62
 - LDAUtil::Itoa, 91
- get_structure_lock
 - HashMap_Array, 87
- get_topic_stats
 - TypeTopicCounts, 139
- get_total_num_words
 - Data_Formatter, 65
 - Unigram_Train_Data_Formatter,
169
- get_ttc
 - Unigram_Model, 143
- get_word
 - WordIndexDictionary, 172

- Hadoop_Checkpointer, 85
 - ~Hadoop_Checkpointer, 85
 - checkpoint, 85
 - Hadoop_Checkpointer, 85
 - Hadoop_Checkpointer, 85
- has_to_synchronize
 - Synchronizer_Helper, 119
 - Unigram_Model_Synchronizer_Helper, 152
- hash
 - simple_map, 112
- HashMap_Array, 86
 - ~HashMap_Array, 87
 - act_map, 87
 - act_map_iter, 87
 - begin, 87
 - count, 87
 - end, 87
 - erase, 87
 - get_lock, 87
 - get_structure_lock, 87
 - HashMap_Array, 87
 - HashMap_Array, 87
 - size, 87
- HashMap_Array::iterator, 89
 - current_iter, 89
 - iterator, 89
 - operator++, 89
- ignore_old_topic
 - Unigram_Model_Tester, 157
- increment
 - TopicCounts, 129
- init
 - Pipeline, 105
 - TBB_Pipeline, 123
 - TopicCounts, 129
 - TypeTopicCounts, 139
- init_dict
 - Unigram_Model_Streaming_Builder, 148
 - Unigram_Model_Training_Builder, 165
- initialize
 - Synchronizer_Helper, 120
 - TypeTopicCounts, 139
 - Unigram_Model_Synchronizer_Helper, 153
- initialize_from_dict
 - WordIndexDictionary, 172
- initialize_from_docs
 - TypeTopicCounts, 139
- initialize_from_dump
 - Parameter, 103
 - TypeTopicCounts, 139
 - WordIndexDictionary, 172
- initialize_from_dumps
 - WordIndexDictionary, 172
- initialize_from_string
 - TypeTopicCounts, 140
- initialize_from_ttc
 - TypeTopicCounts, 140
- initialize_from_values
 - Parameter, 103
- initialize_topics
 - Unigram_Model_Training_Builder, 166
- insert_word
 - TopKList, 133
 - WordIndexDictionary, 172
- insert_word_to_dict
 - Unigram_Model_Streammer, 146
 - Unigram_Test_Data_Formatter, 167
 - Unigram_Train_Data_Formatter, 169
- INTERNAL_SUPPRESS_PROTOBUF_FIELD_DEPRECATION
 - document.pb.cc, 182
- InvalidOldTopicExc, 88
 - InvalidOldTopicExc, 88
 - what, 88
- is_all_iters_done
 - Synchronizer, 117
- is_sorted
 - TopKList, 133
- items
 - TopicCounts, 131
- iteration_done
 - Model_Refiner, 100
 - Unigram_Model_Streammer, 146
 - Unigram_Model_Tester, 155
 - Unigram_Model_Trainer, 161

- iterator
 - Hashmap_Array::iterator, 89
 - TopKList, 132
- LDAUtil, 55
- LDAUtil::DM_Server_Names, 72
 - get_servant_name, 72
 - get_server_endpoint, 72
- LDAUtil::Itoa, 91
 - get_string, 91
- LDAUtil::StringTokenizer, 113
 - tokenize, 113
- LDAUtil::StringTrimmer, 114
 - trim, 114
- length
 - Parameter, 103
 - TopicCounts, 131
- load_metadata
 - Checkpoint, 58
 - Local_Checkpointer, 92
- Local_Checkpointer, 92
 - ~Local_Checkpointer, 92
 - checkpoint, 92
 - load_metadata, 92
 - Local_Checkpointer, 92
 - Local_Checkpointer, 92
 - save_metadata, 93
- log_gamma
 - Dirichlet.cpp, 203
 - Dirichlet.h, 204
- main
 - DM_Server.cpp, 197
 - Formatter/Controller.cpp, 188
 - Merge_Dictionaries.cpp, 257
 - Merge_Topic_Counts.cpp, 259
 - TopicLearner/Controller.cpp, 189
- Main_flags_define.h
 - DEFINE_bool, 226
 - DEFINE_double, 226
 - DEFINE_int32, 226
 - DEFINE_string, 226
- mapped_vec
 - types.h, 247
- match_word_index
 - WordIndexDictionary, 172
- Memcached_Synchronizer
 - TypeTopicCounts, 141
- Merge_Dictionaries.cpp
 - DEFINE_int32, 257
 - DEFINE_string, 257
 - main, 257
- Merge_Topic_Counts.cpp
 - DEFINE_int32, 259
 - DEFINE_string, 259
 - main, 259
- Model, 94
 - get_eval, 94
 - save, 94
 - UNIGRAM, 95
 - write_statistics, 95
- Model_Builder, 96
 - create_execution_strategy, 96
 - create_model_refiner, 96
 - create_output, 97
 - create_pipeline, 97
 - get_model, 97
- Model_Director, 98
 - ~Model_Director, 98
 - build_model, 98
 - Model_Director, 98
 - Model_Director, 98
- Model_Refiner, 99
 - allocate_document_buffer, 100
 - deallocate_document_buffer, 100
 - eval, 100
 - get_nth_document, 100
 - iteration_done, 100
 - optimize, 100
 - read, 100
 - sample, 100
 - test, 101
 - update, 101
 - write, 101
- msg_id
 - Cookie, 63
- NUM_CONSUMERS
 - DM_Server.h, 199
- num_synchs
 - PNGCallback, 108
- num_topics

- TypeTopicCounts, 141
- operator()
 - Filter_Eval, 76
 - Filter_Optimizer, 77
 - Filter_Reader, 78
 - Filter_Sampler, 79
 - Filter_Tester, 80
 - Filter_Updater, 81
 - Filter_Writer, 82
- operator++
 - HashMap_Array::iterator, 89
- operator+=
 - TopicCounts, 129
- operator-=
 - TopicCounts, 129
- operator=
 - Parameter, 103
- optimize
 - Model_Refiner, 100
 - Unigram_Model_Tester, 156
 - Unigram_Model_Trainer, 161
- origLength
 - TopicCounts, 131
- param
 - Parameter.h, 233
- Parameter, 102
 - ~Parameter, 103
 - dump, 103
 - initialize_from_dump, 103
 - initialize_from_values, 103
 - length, 103
 - operator=, 103
 - Parameter, 103
 - sum, 103
 - values, 103
- Parameter.h
 - param, 233
- Pipeline, 104
 - add_eval, 104
 - add_optimizer, 104
 - add_reader, 105
 - add_sampler, 105
 - add_tester, 105
 - add_updater, 105
 - add_writer, 105
 - clear, 105
 - destroy, 105
 - get_eval, 105
 - get_refiner, 105
 - init, 105
 - run, 106
- png_cb
 - PNGJob, 109
- PNGCallback, 107
 - ~PNGCallback, 108
 - finished, 108
 - num_synchs, 108
 - PNGCallback, 108
 - set_done, 108
 - wait_till_done, 108
- PNGCallbackPtr
 - DM_Client.h, 206
- PNGJob, 109
 - delta, 109
 - png_cb, 109
 - PNGJob, 109
 - word, 109
- PNGJobPtr
 - DM_Server.h, 199
- pop
 - GenericTopKList, 84
- print
 - GenericTopKList, 84
 - simple_map, 112
 - TopicCounts, 130
 - TopKList, 133
 - TypeTopicCounts, 140
 - WordIndexDictionary, 173
- PRINT_TIME
 - types.h, 247
- prob_cmp
 - comparator.cpp, 177
 - comparator.h, 178
- push
 - GenericTopKList, 84
- put
 - Client, 60
 - DM_Client, 67
 - DM_Server, 70
 - simple_map, 112

- put_string
 - Context, [62](#)
- putNget_async
 - DM_Server, [70](#)
- QUE_FULL
 - DM_Server.h, [199](#)
- QUIT
 - TopicCounts, [131](#)
- read
 - DocumentReader, [73](#)
 - Model_Refiner, [100](#)
 - Unigram_Model_Streamer, [146](#)
 - Unigram_Model_Tester, [156](#)
 - Unigram_Model_Trainer, [162](#)
- read_from_inp
 - Unigram_Train_Data_Formatter, [170](#)
- release_data_formatter
 - Formatter/Controller.cpp, [188](#)
- remove
 - Client, [60](#)
 - DM_Client, [67](#)
 - DM_Server, [70](#)
- removeOldTop
 - TopicCounts, [130](#)
- replace
 - TopicCounts, [130](#)
 - TypeTopicCounts, [140](#)
- reset_to_synchronize
 - Synchronizer_Helper, [120](#)
 - Unigram_Model_Synchronizer_Helper, [153](#)
- run
 - Pipeline, [106](#)
 - TBB_Pipeline, [123](#)
- sample
 - Model_Refiner, [100](#)
 - sampler, [56](#)
 - Unigram_Model_Tester, [156](#)
 - Unigram_Model_Trainer, [162](#)
- sampler, [56](#)
 - sample, [56](#)
- save
 - Model, [94](#)
 - Unigram_Model, [143](#)
- save_metadata
 - Checkpoint, [58](#)
 - Local_Checkpoint, [93](#)
- Server_Helper, [111](#)
 - combine, [111](#)
- set
 - Client, [60](#)
 - DM_Client, [67](#)
 - DM_Server, [71](#)
- set_all_iters_done
 - Synchronizer, [117](#)
- set_done
 - PNGCallback, [108](#)
- set_parameter
 - Unigram_Model, [143](#)
- setLength
 - TopicCounts, [130](#)
- simple_map, [112](#)
 - ~simple_map, [112](#)
 - clear, [112](#)
 - get, [112](#)
 - hash, [112](#)
 - print, [112](#)
 - put, [112](#)
 - simple_map, [112](#)
 - simple_map, [112](#)
- size
 - HashMap_Array, [87](#)
 - WordIndexDictionary, [173](#)
- size_int
 - types.h, [247](#)
- src/architecture.h, [175](#)
- src/commons/Client.h, [176](#)
- src/commons/comparator.cpp, [177](#)
- src/commons/comparator.h, [178](#)
- src/commons/constants.h, [179](#)
- src/commons/Context.cpp, [180](#)
- src/commons/Context.h, [181](#)
- src/commons/document.pb.cc, [182](#)
- src/commons/document.pb.h, [183](#)
- src/commons/DocumentReader.cpp, [184](#)
- src/commons/DocumentReader.h, [185](#)
- src/commons/DocumentWriter.cpp, [186](#)
- src/commons/DocumentWriter.h, [187](#)

- src/commons/Formatter/Controller.cpp, 188
- src/commons/Formatter/Data_ -
 Formatter.h, 190
- src/commons/Formatter/FormatData_ -
 flags_define.h, 191
- src/commons/LDAUtil.cpp, 192
- src/commons/LDAUtil.h, 193
- src/commons/Server/DistributedMap.cpp, 194
- src/commons/Server/DistributedMap.h, 195
- src/commons/Server/DM_Server.cpp, 196
- src/commons/Server/DM_Server.h, 198
- src/commons/Server/HashMap_Array.h, 200
- src/commons/Server/Server_Helper.h, 201
- src/commons/TopicLearner/Checkpoint.h, 202
- src/commons/TopicLearner/Controller.cpp, 189
- src/commons/TopicLearner/Dirichlet.cpp, 203
- src/commons/TopicLearner/Dirichlet.h, 204
- src/commons/TopicLearner/DM_ -
 Client.cpp, 205
- src/commons/TopicLearner/DM_ -
 Client.h, 206
- src/commons/TopicLearner/Execution_ -
 Strategy.h, 207
- src/commons/TopicLearner/Filter_ -
 Eval.cpp, 208
- src/commons/TopicLearner/Filter_Eval.h, 209
- src/commons/TopicLearner/Filter_ -
 Optimizer.cpp, 210
- src/commons/TopicLearner/Filter_ -
 Optimizer.h, 211
- src/commons/TopicLearner/Filter_ -
 Reader.cpp, 212
- src/commons/TopicLearner/Filter_ -
 Reader.h, 213
- src/commons/TopicLearner/Filter_ -
 Sampler.cpp, 214
- src/commons/TopicLearner/Filter_ -
 Sampler.h, 215
- src/commons/TopicLearner/Filter_ -
 Tester.cpp, 216
- src/commons/TopicLearner/Filter_ -
 Tester.h, 217
- src/commons/TopicLearner/Filter_ -
 Updater.cpp, 218
- src/commons/TopicLearner/Filter_ -
 Updater.h, 219
- src/commons/TopicLearner/Filter_ -
 Writer.cpp, 220
- src/commons/TopicLearner/Filter_ -
 Writer.h, 221
- src/commons/TopicLearner/GenericTopKList.h, 222
- src/commons/TopicLearner/Main_flags_ -
 define.h, 223
- src/commons/TopicLearner/Model.h, 227
- src/commons/TopicLearner/Model_ -
 Builder.h, 228
- src/commons/TopicLearner/Model_ -
 Director.cpp, 229
- src/commons/TopicLearner/Model_ -
 Director.h, 230
- src/commons/TopicLearner/Model_ -
 Refiner.h, 231
- src/commons/TopicLearner/Parameter.cpp, 232
- src/commons/TopicLearner/Parameter.h, 233
- src/commons/TopicLearner/Pipeline.h, 234
- src/commons/TopicLearner/Synchronized_ -
 Training_Execution_ -
 Strategy.cpp, 235
- src/commons/TopicLearner/Synchronized_ -
 Training_Execution_Strategy.h, 236
- src/commons/TopicLearner/Synchronizer.cpp, 237
- src/commons/TopicLearner/Synchronizer.h, 238
- src/commons/TopicLearner/Synchronizer_ -
 Helper.h, 239

- src/commons/TopicLearner/TBB_-
Pipeline.cpp, 240
- src/commons/TopicLearner/TBB_-
Pipeline.h, 241
- src/commons/TopicLearner/Testing_-
Execution_Strategy.cpp, 242
- src/commons/TopicLearner/Testing_-
Execution_Strategy.h, 243
- src/commons/TopicLearner/Training_-
Execution_Strategy.cpp, 244
- src/commons/TopicLearner/Training_-
Execution_Strategy.h, 245
- src/commons/types.h, 246
- src/commons/WordIndexDictionary.cpp,
248
- src/commons/WordIndexDictionary.h,
249
- src/mainpage.h, 250
- src/multi_mcahine_usage.h, 251
- src/single_machine_usage.h, 252
- src/Unigram_-
Model/Formatter/Unigram_-
Test_Data_Formatter.cpp,
253
- src/Unigram_-
Model/Formatter/Unigram_-
Test_Data_Formatter.h, 254
- src/Unigram_-
Model/Formatter/Unigram_-
Train_Data_Formatter.cpp,
255
- src/Unigram_-
Model/Formatter/Unigram_-
Train_Data_Formatter.h, 256
- src/Unigram_Model/Merge/Merge_-
Dictionaries.cpp, 257
- src/Unigram_Model/Merge/Merge_-
Topic_Counts.cpp, 258
- src/Unigram_Model/Server/Unigram_-
Model_Server_Helper.cpp,
260
- src/Unigram_Model/Server/Unigram_-
Model_Server_Helper.h, 261
- src/Unigram_Model/TopicLearner/eff_-
small_map.cpp, 262
- src/Unigram_Model/TopicLearner/eff_-
small_map.h, 263
- src/Unigram_-
Model/TopicLearner/Hadoop_-
Checkpoint.cpp, 264
- src/Unigram_-
Model/TopicLearner/Hadoop_-
Checkpoint.h, 265
- src/Unigram_-
Model/TopicLearner/Local_-
Checkpoint.cpp, 266
- src/Unigram_-
Model/TopicLearner/Local_-
Checkpoint.h, 267
- src/Unigram_-
Model/TopicLearner/sampler.cpp,
268
- src/Unigram_-
Model/TopicLearner/sampler.h,
269
- src/Unigram_-
Model/TopicLearner/TopicCounts.cpp,
270
- src/Unigram_-
Model/TopicLearner/TopicCounts.h,
271
- src/Unigram_-
Model/TopicLearner/TopKList.cpp,
272
- src/Unigram_-
Model/TopicLearner/TopKList.h,
273
- src/Unigram_-
Model/TopicLearner/TypeTopicCounts.cpp,
274
- src/Unigram_-
Model/TopicLearner/TypeTopicCounts.h,
275
- src/Unigram_-
Model/TopicLearner/Unigram_-
Model.cpp, 276
- src/Unigram_-
Model/TopicLearner/Unigram_-
Model.h, 277
- src/Unigram_-
Model/TopicLearner/Unigram_-
Model_Stream.cpp, 278

- src/Unigram_-
 - Model/TopicLearner/Unigram_-
 - Model_Streamer.h, [279](#)
- src/Unigram_-
 - Model/TopicLearner/Unigram_-
 - Model_Streaming_Builder.cpp, [280](#)
- src/Unigram_-
 - Model/TopicLearner/Unigram_-
 - Model_Streaming_Builder.h, [281](#)
- src/Unigram_-
 - Model/TopicLearner/Unigram_-
 - Model_Synchronized_-
 - Training_Builder.cpp, [282](#)
- src/Unigram_-
 - Model/TopicLearner/Unigram_-
 - Model_Synchronized_-
 - Training_Builder.h, [283](#)
- src/Unigram_-
 - Model/TopicLearner/Unigram_-
 - Model_Synchronizer_-
 - Helper.cpp, [284](#)
- src/Unigram_-
 - Model/TopicLearner/Unigram_-
 - Model_Synchronizer_Helper.h, [285](#)
- src/Unigram_-
 - Model/TopicLearner/Unigram_-
 - Model_Tester.cpp, [286](#)
- src/Unigram_-
 - Model/TopicLearner/Unigram_-
 - Model_Tester.h, [287](#)
- src/Unigram_-
 - Model/TopicLearner/Unigram_-
 - Model_Testing_Builder.cpp, [288](#)
- src/Unigram_-
 - Model/TopicLearner/Unigram_-
 - Model_Testing_Builder.h, [289](#)
- src/Unigram_-
 - Model/TopicLearner/Unigram_-
 - Model_Trainer.cpp, [290](#)
- src/Unigram_-
 - Model/TopicLearner/Unigram_-
 - Model_Trainer.h, [291](#)
- src/Unigram_-
 - Model/TopicLearner/Unigram_-
 - Model_Training_Builder.cpp, [292](#)
- src/Unigram_-
 - Model/TopicLearner/Unigram_-
 - Model_Training_Builder.h, [293](#)
- src/usage.h, [294](#)
- sum
 - Parameter, [103](#)
- synchronize
 - Synchronized_Training_-
 - Execution_Strategy.cpp, [235](#)
 - Synchronizer, [118](#)
 - Synchronizer_Helper, [120](#)
 - Unigram_Model_Synchronizer_-
 - Helper, [153](#)
 - Synchronized_Training_Execution_-
 - Strategy, [115](#)
 - ~Synchronized_Training_-
 - Execution_Strategy, [116](#)
 - execute, [116](#)
 - Synchronized_Training_-
 - Execution_Strategy, [116](#)
 - Synchronized_Training_-
 - Execution_Strategy, [116](#)
 - Synchronized_Training_Execution_-
 - Strategy.cpp
 - synchronize, [235](#)
- Synchronizer, [117](#)
 - ~Synchronizer, [117](#)
 - is_all_iters_done, [117](#)
 - set_all_iters_done, [117](#)
 - synchronize, [118](#)
 - Synchronizer, [117](#)
- Synchronizer_Helper, [119](#)
 - end_putNget, [119](#)
 - has_to_synchronize, [119](#)
 - initialize, [120](#)
 - reset_to_synchronize, [120](#)
 - synchronize, [120](#)
- TBB_Pipeline, [121](#)
 - ~TBB_Pipeline, [122](#)
 - _eval, [124](#)
 - _init, [124](#)

- [_optimizer](#), 124
 - [_pipeline](#), 124
 - [_reader](#), 124
 - [_refiner](#), 124
 - [_sampler](#), 124
 - [_tester](#), 124
 - [_updater](#), 124
 - [_writer](#), 124
 - [add_eval](#), 122
 - [add_optimizer](#), 122
 - [add_reader](#), 122
 - [add_sampler](#), 122
 - [add_tester](#), 122
 - [add_updater](#), 122
 - [add_writer](#), 122
 - [clear](#), 123
 - [destroy](#), 123
 - [get_eval](#), 123
 - [get_refiner](#), 123
 - [init](#), 123
 - [run](#), 123
 - [TBB_Pipeline](#), 122
 - [TBB_Pipeline](#), 122
- test
 - [Model_Refiner](#), 101
 - [Unigram_Model_Tester](#), 156
 - [Unigram_Model_Trainer](#), 162
- Testing_Execution_Strategy, 125
 - [~Testing_Execution_Strategy](#), 125
 - [execute](#), 125
 - [Testing_Execution_Strategy](#), 125
 - [Testing_Execution_Strategy](#), 125
- TIME
 - [types.h](#), 247
- tokenize
 - [LDAUtil::StringTokenizer](#), 113
- tokens_per_topic
 - [TypeTopicCounts](#), 141
- top
 - [GenericTopKList](#), 84
- top_topics
 - [TypeTopicCounts](#), 141
- topic_stats
 - [TypeTopicCounts](#), 141
- TopicCounts, 126
 - [~TopicCounts](#), 127
 - [addNewTop](#), 127
 - [addNewTopAftChk](#), 127
 - [assign](#), 128
 - [compact](#), 128
 - [convertTo](#), 128
 - [convertTo_d](#), 128
 - [decrement](#), 128
 - [equal](#), 128
 - [findAndDecrement](#), 129
 - [findAndIncrement](#), 129
 - [findOldnNew](#), 129
 - [frequency](#), 131
 - [get_counts](#), 129
 - [get_frequency](#), 129
 - [increment](#), 129
 - [init](#), 129
 - [items](#), 131
 - [length](#), 131
 - [operator+=](#), 129
 - [operator-=](#), 129
 - [origLength](#), 131
 - [print](#), 130
 - [QUIT](#), 131
 - [removeOldTop](#), 130
 - [replace](#), 130
 - [setLength](#), 130
 - [TopicCounts](#), 127
 - [upd_count](#), 130
 - [vec_items](#), 131
- topicCounts
 - [TopicCounts.h](#), 271
- TopicCounts.h
 - [topicCounts](#), 271
- TopicLearner/Controller.cpp
 - [main](#), 189
- TopKList, 132
 - [~TopKList](#), 132
 - [clear](#), 132
 - [get_beg](#), 132
 - [get_end](#), 132
 - [get_max](#), 133
 - [insert_word](#), 133
 - [is_sorted](#), 133
 - [iterator](#), 132
 - [print](#), 133
 - [TopKList](#), 132

- tppair
 - types.h, 247
- Training_Execution_Strategy, 134
 - ~Training_Execution_Strategy, 134
 - execute, 134
 - Training_Execution_Strategy, 134
 - Training_Execution_Strategy, 134
- trim
 - LDAUtil::StringTrimmer, 114
- types.h
 - base_generator_type, 247
 - bigppair, 247
 - mapped_vec, 247
 - PRINT_TIME, 247
 - size_int, 247
 - TIME, 247
 - tppair, 247
 - wppair, 247
- TypeTopicCounts, 136
 - ~TypeTopicCounts, 137
 - clear_stats, 137
 - destroy, 137
 - dump, 138
 - equal, 138
 - estimate_alphas, 138
 - estimate_fit, 138
 - estimate_memoryn_warn, 138
 - get_counts, 138
 - get_lock, 138
 - get_num_topics, 138
 - get_num_words, 139
 - get_topic_stats, 139
 - init, 139
 - initialize, 139
 - initialize_from_docs, 139
 - initialize_from_dump, 139
 - initialize_from_string, 140
 - initialize_from_ttc, 140
 - Memcached_Synchronizer, 141
 - num_topics, 141
 - print, 140
 - replace, 140
 - tokens_per_topic, 141
 - top_topics, 141
 - topic_stats, 141
 - TypeTopicCounts, 137
 - upd_count, 140
 - verify_header, 140
- TypeTopicCounts.cpp
 - cnt_cmp, 274
 - cnt_cmp_ttc, 274
- UNIGRAM
 - Model, 95
- Unigram_Model, 142
 - ~Unigram_Model, 143
 - ALPHA, 143
 - BETA, 143
 - get_eval, 143
 - get_parameter, 143
 - get_ttc, 143
 - save, 143
 - set_parameter, 143
 - Unigram_Model, 143
 - Unigram_Model, 143
 - write_statistics, 143
- Unigram_Model_Server_Helper, 144
 - ~Unigram_Model_Server_Helper, 144
 - combine, 144
 - Unigram_Model_Server_Helper, 144
 - Unigram_Model_Server_Helper, 144
- Unigram_Model_Stream er, 145
 - ~Unigram_Model_Stream er, 146
 - insert_word_to_dict, 146
 - iteration_done, 146
 - read, 146
 - Unigram_Model_Stream er, 146
 - Unigram_Model_Stream er, 146
 - write, 146
- Unigram_Model_Streaming_Builder, 147
 - ~Unigram_Model_Streaming_Builder, 148
 - _dict, 149
 - _global_dict, 149
 - _model, 149
 - _pipeline, 149
 - _refiner, 149
 - _strategy, 149
 - create_execution_strategy, 148

- create_model_refiner, 148
- create_output, 148
- create_pipeline, 148
- get_dict, 148
- get_model, 148
- init_dict, 148
- Unigram_Model_Streaming_
Builder, 148
- Unigram_Model_Streaming_
Builder, 148
- Unigram_Model_Synchronized_
Training_Builder, 150
- ~Unigram_Model_Synchronized_
Training_Builder, 150
- _sync_helper, 151
- create_execution_strategy, 150
- Unigram_Model_Synchronized_
Training_Builder, 150
- Unigram_Model_Synchronized_
Training_Builder, 150
- Unigram_Model_Synchronizer_Helper,
152
- ~Unigram_Model_Synchronizer_
Helper, 152
- end_putNget, 152
- has_to_synchronize, 152
- initialize, 153
- reset_to_synchronize, 153
- synchronize, 153
- Unigram_Model_Synchronizer_
Helper, 152
- Unigram_Model_Synchronizer_
Helper, 152
- Unigram_Model_Tester, 154
- ~Unigram_Model_Tester, 155
- _alpha, 157
- _beta, 157
- _num_topics, 157
- _num_words, 157
- _tdoc_writer, 157
- _ttc, 157
- _wdoc_rdr, 157
- allocate_document_buffer, 155
- deallocate_document_buffer, 155
- doc_index, 157
- eval, 155
- get_nth_document, 155
- ignore_old_topic, 157
- iteration_done, 155
- optimize, 156
- read, 156
- sample, 156
- test, 156
- Unigram_Model_Tester, 155
- Unigram_Model_Tester, 155
- update, 156
- write, 156
- Unigram_Model_Testing_Builder, 158
- ~Unigram_Model_Testing_Builder,
158
- create_execution_strategy, 158
- create_model_refiner, 158
- Unigram_Model_Testing_Builder,
158
- Unigram_Model_Testing_Builder,
158
- Unigram_Model_Trainer, 160
- ~Unigram_Model_Trainer, 161
- allocate_document_buffer, 161
- deallocate_document_buffer, 161
- doc_index, 163
- eval, 161
- get_nth_document, 161
- iteration_done, 161
- optimize, 161
- read, 162
- sample, 162
- test, 162
- Unigram_Model_Trainer, 161
- Unigram_Model_Trainer, 161
- update, 162
- write, 162
- Unigram_Model_Training_Builder, 164
- ~Unigram_Model_Training_
Builder, 165
- _checkpoint, 166
- _dict, 166
- _model, 166
- _pipeline, 166
- _refiner, 166
- _strategy, 166
- create_execution_strategy, 165

- create_model_refiner, 165
- create_output, 165
- create_pipeline, 165
- get_dict, 165
- get_model, 165
- init_dict, 165
- initialize_topics, 166
- Unigram_Model_Training_Builder, 165
- Unigram_Model_Training_Builder, 165
- Unigram_Test_Data_Formatter, 167
- ~Unigram_Test_Data_Formatter, 167
- insert_word_to_dict, 167
- Unigram_Test_Data_Formatter, 167
- Unigram_Test_Data_Formatter, 167
- Unigram_Train_Data_Formatter, 168
- ~Unigram_Train_Data_Formatter, 169
- _dict, 170
- _doc_writer, 170
- _in, 170
- _num_docs, 170
- _num_words_in_all_docs, 170
- _stopWords, 170
- format, 169
- get_dictionary, 169
- get_num_docs, 169
- get_total_num_words, 169
- insert_word_to_dict, 169
- read_from_inp, 170
- Unigram_Train_Data_Formatter, 169
- Unigram_Train_Data_Formatter, 169
- upd_count
 - TopicCounts, 130
 - TypeTopicCounts, 140
- update
 - Model_Refiner, 101
 - Unigram_Model_Tester, 156
 - Unigram_Model_Trainer, 162
- values
 - Parameter, 103
- vec_items
 - TopicCounts, 131
- verify_header
 - TypeTopicCounts, 140
- wait_for_all
 - Client, 60
 - DM_Client, 68
- wait_till_done
 - Client, 60
 - DM_Client, 68
 - PNGCallback, 108
- waitForAllClients_async
 - DM_Server, 71
- what
 - InvalidOldTopicExc, 88
- word
 - PNGJob, 109
- WordIndexDictionary, 171
- ~WordIndexDictionary, 171
- dump, 172
- frequencies, 173
- get_freq, 172
- get_index, 172
- get_num_words, 172
- get_prev_index, 172
- get_word, 172
- initialize_from_dict, 172
- initialize_from_dump, 172
- initialize_from_dumps, 172
- insert_word, 172
- match_word_index, 172
- print, 173
- size, 173
- WordIndexDictionary, 171
- wppair
 - types.h, 247
- write
 - DocumentWriter, 74
 - Model_Refiner, 101
 - Unigram_Model_Streamer, 146
 - Unigram_Model_Tester, 156
 - Unigram_Model_Trainer, 162
- write_statistics
 - Model, 95
 - Unigram_Model, 143