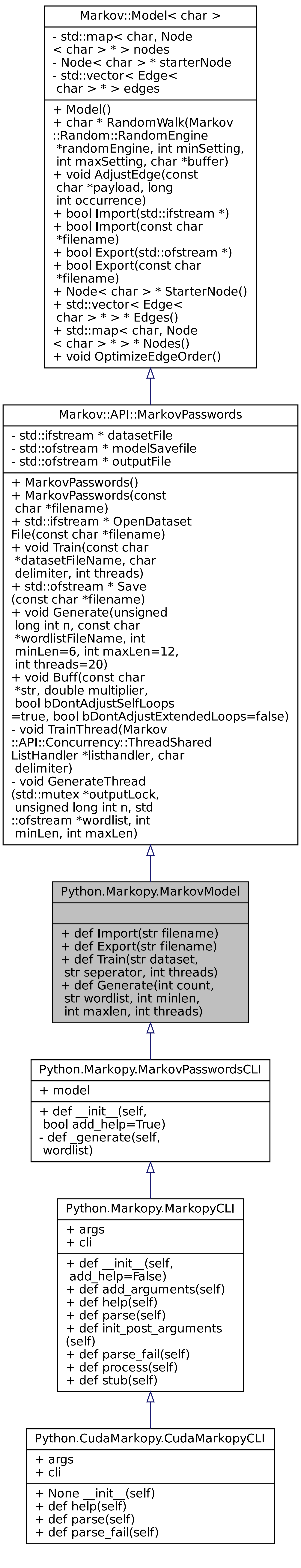
Markov::Model< char >
- std::map< char, Node < char > * > nodes
- Node< char > * starterNode
- std::vector< Edge< char > * > edges
+ Model()
+ char * RandomWalk(Markov ::Random::RandomEngine *randomEngine, int minSetting, int maxSetting, char *buffer)
+ void AdjustEdge(const char *payload, long int occurrence)
+ bool Import(std::ifstream *)
+ bool Import(const char *filename)
+ bool Export(std::ofstream *)
+ bool Export(const char *filename)
+ Node< char > * StarterNode()
+ std::vector< Edge< char > * > * Edges()
+ std::map< char, Node < char > * > * Nodes()
+ void OptimizeEdgeOrder()
Markov::API::MarkovPasswords
- std::ifstream * datasetFile
- std::ofstream * modelSavefile
- std::ofstream * outputFile
+ MarkovPasswords()
+ MarkovPasswords(const char *filename)
+ std::ifstream * OpenDataset File(const char *filename)
+ void Train(const char *datasetFileName, char delimiter, int threads)
+ std::ofstream * Save (const char *filename)
+ void Generate(unsigned long int n, const char *wordlistFileName, int minLen=6, int maxLen=12, int threads=20)
+ void Buff(const char *str, double multiplier, bool bDontAdjustSelfLoops =true, bool bDontAdjustExtendedLoops=false)
- void TrainThread(Markov ::API::Concurrency::ThreadShared ListHandler *listhandler, char delimiter)
- void GenerateThread (std::mutex *outputLock, unsigned long int n, std ::ofstream *wordlist, int minLen, int maxLen)
Python.Markopy.MarkovModel
+ def Import(str filename)
+ def Export(str filename)
+ def Train(str dataset, str seperator, int threads)
+ def Generate(int count, str wordlist, int minlen, int maxlen, int threads)
Python.Markopy.MarkovPasswordsCLI
+ model
+ def __init__(self, bool add_help=True)
- def _generate(self, wordlist)
Python.Markopy.MarkopyCLI
+ args
+ cli
+ def __init__(self, add_help=False)
+ def add_arguments(self)
+ def help(self)
+ def parse(self)
+ def init_post_arguments (self)
+ def parse_fail(self)
+ def process(self)
+ def stub(self)
Python.CudaMarkopy.CudaMarkopyCLI
+ args
+ cli
+ None __init__(self)
+ def help(self)
+ def parse(self)
+ def parse_fail(self)