Apologies for the poor code organization. Please understand the below before making any changes:

Coreference.py

This file contains the code for parsing the antecedentpairs2b.xml file.

Function
Split(), Separate(), Relation(), Positive() find out the entities and antecedents present in the xml file. You need not change the code for the particular functions.

Global Variable Entity [] is a list of entities present in the document.
Global Variable Antecedent [] is a list of antecedents present in the document
Global Variable Order [] tracks the order of entities, fullstops and article endings.
Global Variable String[] and list[] contain the Entity information in most rudimentary form i.e. present with tagging information.

Fullstop_and_Article_end() function is used to find out the Order [] list. You need not change the code here.

Modified_Text_Information() function is used to modify the already existing Text_Information file by appending fullstop and article information which appears as output in Modified_Text_Information.txt. Alternately the contents of Text_Information file is also present in Mention [] globally declared list.

Examples_File() functions generates positive and negative examples in format:
Corefers(text1,text2) and places the positive examples in coreference.f file
And all the negative examples are placed inside the coreference.n file.

The information that is starts_with_capital(text) which is supposed to go with the coreference.b file is also generated in the same Examples_File () function.

Contents of Modified_Text_Information.txt is in this format:

Mention no. Entity/Antecedent Entity No. Antecedent No. Text

Entity No. has a suffix "_A#" added to it where # corresponds to the article no. and similarly text has a suffix "_m#" where # corresponds to the mention no. Also, "Fullstop" is included in the file to maintain the order. This "Fullstop" information can be used to make in_same_sentence(text, text) information.

The best method to work ahead with this code is to use "Modified_Text_Information.txt" and make new functions and make predicates based on the available information. Please adhere to it and use local variables inside the function to make predicates. Append your generated predicates into the "Background.txt" file and execute the shell script after adding Rules inside the Rules.txt file.

coreference.f and coreference.n are 100% correct examples. Out of C(319,2) = 50721 total coreferences. 72 are the positive examples and remaining 50649 are the negative examples and that is the correct no. present in both the files. I have verified everything twice and the code is working fine for entire xml document.

Whenever you want to add more predicate inside the coreference.b file open "Background.txt" in append mode and write lines into the file. To include new rules, put the rules inside the Rules.txt file or predicates to include in body part of hypothesis.

There is a shell script named "myscript" which concatenates rules present in "Rules.txt" and background knowledge from "Background.txt" and puts them together into coreference.b file.