Fact Checker Report

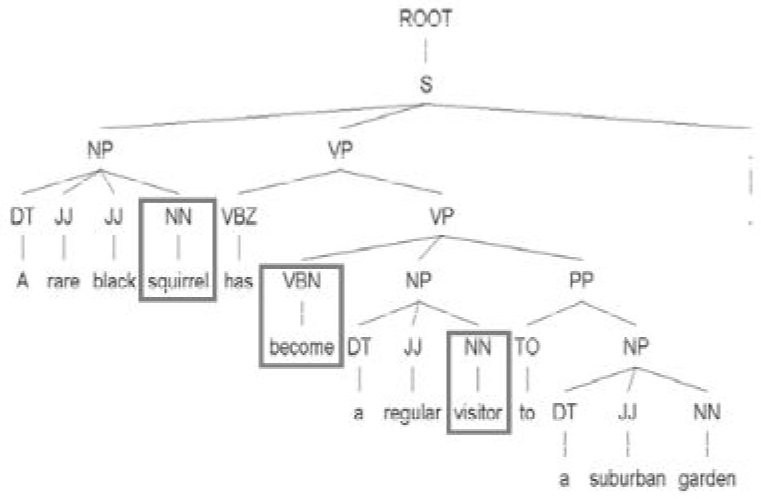
Approach/Method to be used (in short)

- Create **triplets** (*subject-verb-object*) of all the sentences separately in the document and store them. A **subject** is a "*real-world entity*" such as people, places or things. A **predicate/verb** describes an "*attribute*" of that entity and the **object** is "*an entity, a string, a numerical value, or a date.*"
    - **Parse** the sentence using **Stanford parser**.
    - Find the subject, verb and object using **BFS** (breadth first search) and other techniques.
- Create a triplet of the fact.
- Find **similarity** between the triplet of the fact with each of the triplets of the document(s).
- Use a **scoring function** to compare the fact triplet against the document triplets.
- The measure of similarity (score function) would tell us whether the fact is true or false.

Algorithm (in detail)

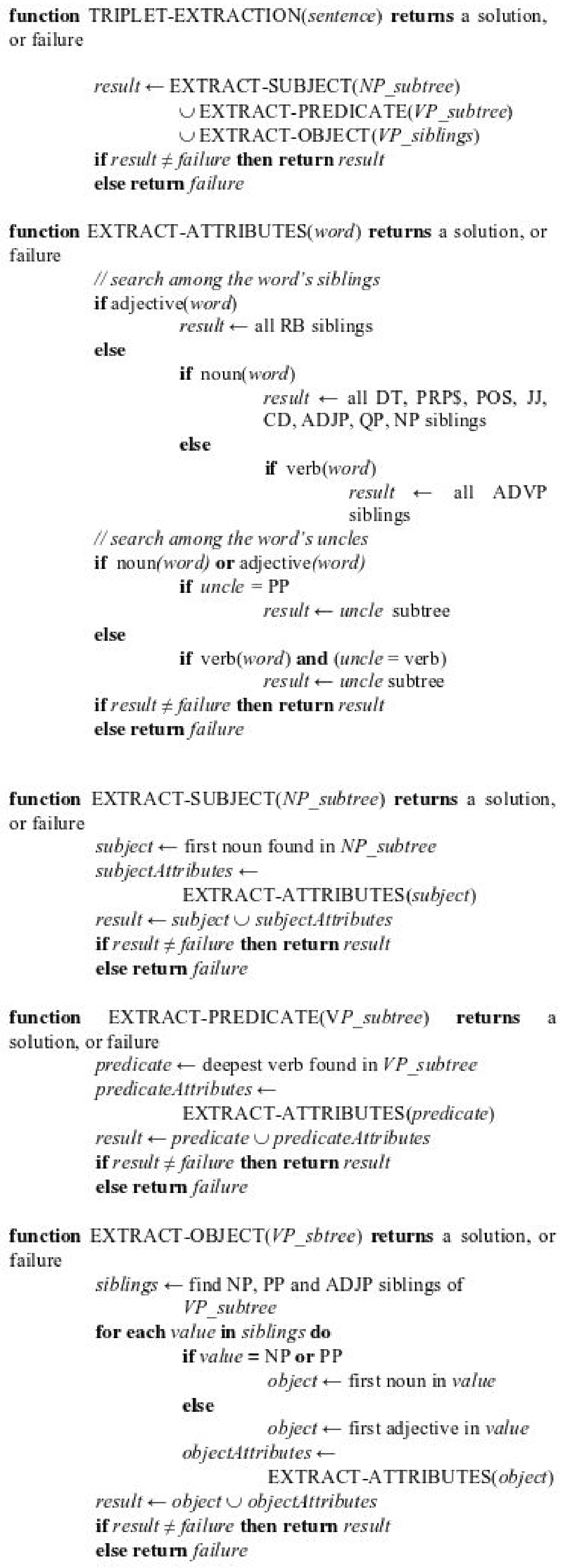
- **Creating/Finding a triplet**
    - Run the sentence through the stanford parser.
    - A sentence (S) is represented by the parser as a tree having three children: a noun phrase (NP), a verbal phrase (VP) and the full stop (.). The root of the tree will be S.
    - Firstly we find the subject of the sentence. In order to find it, we are going to search in the NP subtree. The subject will be found by performing breadth first search (BFS) and selecting the first descendent of NP that is a noun.
    - For determining the predicate of the sentence, a search will be performed in the VP subtree. The deepest verb descendent of the verb phrase will give the second element of the triplet.
    - Thirdly, we look for objects. These can be found in three different subtrees, all siblings of the VP subtree containing the predicate. The subtrees are: PP (prepositional phrase), NP and ADJP (adjective phrase). In NP and PP we search for the first noun, while in ADJP we find the first adjective. An

example of a tree with marked subject-verb-object is:



- Applying the above mentioned steps on this tree gives us squirrel-become-visitor as subject-verb-object.

- Pseudo-code:



```
function TRIPLET-EXTRACTION(sentence) returns a solution,
or failure

        result ← EXTRACT-SUBJECT(NP_subtree)
                $\cup$ EXTRACT-PREDICATE(VP_subtree)
                $\cup$ EXTRACT-OBJECT(VP_siblings)
        if result $\neq$ failure then return result
        else return failure

function EXTRACT-ATTRIBUTES(word) returns a solution, or
failure
        // search among the word's siblings
        if adjective(word)
                result ← all RB siblings
        else
                if noun(word)
                        result ← all DT, PRP$, POS, JJ,
                        CD, ADJP, QP, NP siblings
                else
                        if verb(word)
                                result ← all ADVP
                                siblings
        // search among the word's uncles
        if noun(word) or adjective(word)
                if uncle = PP
                        result ← uncle subtree
        else
                if verb(word) and (uncle = verb)
                        result ← uncle subtree
        if result $\neq$ failure then return result
        else return failure


function EXTRACT-SUBJECT(NP_subtree) returns a solution,
or failure
        subject ← first noun found in NP_subtree
        subjectAttributes ←
                EXTRACT-ATTRIBUTES(subject)
        result ← subject $\cup$ subjectAttributes
        if result $\neq$ failure then return result
        else return failure

function EXTRACT-PREDICATE(VP_subtree) returns a
solution, or failure
        predicate ← deepest verb found in VP_subtree
        predicateAttributes ←
                EXTRACT-ATTRIBUTES(predicate)
        result ← predicate $\cup$ predicateAttributes
        if result $\neq$ failure then return result
        else return failure

function EXTRACT-OBJECT(VP_sbtree) returns a solution, or
failure
        siblings ← find NP, PP and ADJP siblings of
                VP_subtree
        for each value in siblings do
                if value = NP or PP
                        object ← first noun in value
                else
                        object ← first adjective in value
                objectAttributes ←
                        EXTRACT-ATTRIBUTES(object)
        result ← object $\cup$ objectAttributes
        if result $\neq$ failure then return result
        else return failure
```

- **Comparing fact vs the document**
    - Compare fact-triplet with the triplet of each of the sentences in the document.
    - Assuming subject, verb and object as triplet[1], triplet[2] and triplet[3].
    - If fact-triplet[1] is equal to triplet[1] of some sentence (Check for synonyms/antonyms also), compare fact-triplet[2] to triplet[2] of the sentence (check for synonyms/antonyms also- We can use wordnet here), if they are similar compare the third triplet (triplet[3]).
    - If all three match anywhere (positive) we have found that the fact is correct. If triplet[3] turns out to be antonym (negative) of fact-triplet[3] anywhere, the fact is wrong.