Q1:a

Original text:

This is a random collection of 200 words designed to see if nltk can properly detect proper tokens. I should try typing more complex words like U.S.A. , M.sc, and fan-dom. Maybe nltk can properly do it. I've typed maybe 50 words so far. I wonder if it matters whether I put spaces for my dots and commas.Like this sentence . I also want to test commas,so I will do that in this sentence. I also want to test if it can detect tokens through other special characters like ';'. I like oranges;I hate rap music ; Is this the real life? ; Maybe I should type the rest of words as complicated as I can. Hmm.... Lets see how I can do that. I can try detecting dashes as well. Rocket-man is not the right way to type that particular word. M*A*S*H* is a hard word to tokenize wonder how nltk does. Writing programming/scripting languages is probably a good idea: C# , C/C++ , HTML5 ,Python. Tokenizing seems to be the first step to text analytics. Proper tokens seems like the key for analysing text. Tokens' properties must be perfect for error-free operation. Stuck in a landslide no escape from reality!

Tokens:

['This', 'is', 'a', 'random', 'collection', 'of', '200', 'words', 'designed', 'to', 'see', 'if', 'nltk', 'can', 'properly', 'detect', 'proper', 'tokens', '.', 'I', 'should', 'try', 'typing', 'more', 'complex', 'words', 'like', 'U.S.A.', ',', 'M.sc', ',', 'and', 'fan-dom', '.', 'Maybe', 'nltk', 'can', 'properly', 'do', 'it', '.', 'I', "'ve", 'typed', 'maybe', '50', 'words', 'so', 'far', '.', 'I', 'wonder', 'if', 'it', 'matters', 'whether', 'I', 'put', 'spaces', 'for', 'my', 'dots', 'and', 'commas.Like', 'this', 'sentence', '.', 'I', 'also', 'want', 'to', 'test', 'commas', ',', 'so', 'I', 'will', 'do', 'that', 'in', 'this', 'sentence', '.', 'I', 'also', 'want', 'to', 'test', 'if', 'it', 'can', 'detect', 'tokens', 'through', 'other', 'special', 'characters', 'like', '"', ';', '"', '.', 'I', 'like', 'oranges', ';', 'I', 'hate', 'rap', 'music', ';', 'Is', 'this', 'the', 'real', 'life', '?', ';', 'Maybe', 'I', 'should', 'type', 'the', 'rest', 'of', 'words', 'as', 'complicated', 'as', 'I', 'can', '.', 'Hmm', '...', '.', 'Lets', 'see', 'how', 'I', 'can', 'do', 'that', '.', 'I', 'can', 'try', 'detecting', 'dashes', 'as', 'well', '.', 'Rocket-man', 'is', 'not', 'the', 'right', 'way', 'to', 'type', 'that', 'particular', 'word', '.', 'M*A*S*H*', 'is', 'a', 'hard', 'word', 'to', 'tokenize', 'wonder', 'how', 'nltk', 'does', '.', 'Writing', 'programming/scripting', 'languages', 'is', 'probably', 'a', 'good', 'idea', ':', 'C', '#', ',', 'C/C++', ',', 'HTML5', ',', 'Python', '.', 'Tokenizing', 'seems', 'to', 'be', 'the', 'first', 'step', 'to', 'text', 'analytics', '.', 'Proper', 'tokens', 'seems', 'like', 'the', 'key', 'for', 'analysing', 'text', '.', 'Tokens', '"', 'properties', 'must', 'be', 'perfect', 'for', 'error-free', 'operation', '.', 'Stuck', 'in', 'a', 'landslide', 'no', 'escape', 'from', 'reality', '!']

Issues:

Nltk struggles when user forgets a space after punctuation ‘commas.Like’

Nltk struggles when text has special characters .It resolves it by simply creating new tokens for the characters ‘!’ or just ignores them ‘programming/scripting’

Nltk has issues with words having inbuilt special chars in the words itself ‘c’ and ‘#’ instead of ‘C#’

It has issues with I’ve and splits it into two tokens. This is probably not intended as ‘I’ ‘’ve’ but as ‘I’ve’

Also nltk treats capitals as different words.

You can use regular expression to fix spacing errors. For special errors big data approach can be used to learn special words to create tokens properly.

Q1:b

Normalization by converting text to lower case.

tokens:

['this', 'is', 'a', 'random', 'collection', 'of', '200', 'words', 'designed', 'to', 'see', 'if', 'nltk', 'can', 'properly', 'detect', 'proper', 'tokens', '.', 'i', 'should', 'try', 'typing', 'more', 'complex', 'words', 'like', 'u.s.a.', ',', 'm.sc', ',', 'and', 'fan-dom', '.', 'maybe', 'nltk', 'can', 'properly', 'do', 'it', '.', 'i', "'ve", 'typed', 'maybe', '50', 'words', 'so', 'far', '.', 'i', 'wonder', 'if', 'it', 'matters', 'whether', 'i', 'put', 'spaces', 'for', 'my', 'dots', 'and', 'commas.like', 'this', 'sentence', '.', 'i', 'also', 'want', 'to', 'test', 'commas', ',', 'so', 'i', 'will', 'do', 'that', 'in', 'this', 'sentence', '.', 'i', 'also', 'want', 'to', 'test', 'if', 'it', 'can', 'detect', 'tokens', 'through', 'other', 'special', 'characters', 'like', '"', ';', '"', '.', 'i', 'like', 'oranges', ';', 'i', 'hate', 'rap', 'music', ';', 'is', 'this', 'the', 'real', 'life', '?', ';', 'maybe', 'i', 'should', 'type', 'the', 'rest', 'of', 'words', 'as', 'complicated', 'as', 'i', 'can', '.', 'hmm', '...', '.', 'lets', 'see', 'how', 'i', 'can', 'do', 'that', '.', 'i', 'can', 'try', 'detecting', 'dashes', 'as', 'well', '.', 'rocket-man', 'is', 'not', 'the', 'right', 'way', 'to', 'type', 'that', 'particular', 'word', '.', 'm*a*s*h*', 'is', 'a', 'hard', 'word', 'to', 'tokenize', 'wonder', 'how', 'nltk', 'does', '.', 'writing', 'programming/scripting', 'languages', 'is', 'probably', 'a', 'good', 'idea', ':', 'c', '#', ',', 'c/c++', ',', 'html5', ',', 'python', '.', 'tokenizing', 'seems', 'to', 'be', 'the', 'first', 'step', 'to', 'text', 'analytics', '.', 'proper', 'tokens', 'seems', 'like', 'the', 'key', 'for', 'analysing', 'text', '.', 'tokens', '"', 'properties', 'must', 'be', 'perfect', 'for', 'error-free', 'operation', '.', 'stuck', 'in', 'a', 'landslide', 'no', 'escape', 'from', 'reality', '!']

Solution: Converted all text to lower case to remove ambiguity between them.
There are issues with special capital words that have meaning, like “u.s.a” instead of U.S.A and ‘m*a*s*h’ instead of ‘M*A*S*H’

Q1:c

Applying pos tag on previous question

print(nltk.pos_tag(words))

Tagged tokens:

[('this', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('random', 'JJ'), ('collection', 'NN'), ('of', 'IN'), ('200', 'CD'), ('words', 'NNS'), ('designed', 'VBN'), ('to', 'TO'), ('see', 'VB'), ('if', 'IN'), ('nltk', 'NNS'), ('can', 'MD'), ('properly', 'RB'), ('detect', 'VB'), ('proper', 'JJ'), ('tokens', 'NNS'), ('.', '.'), ('i', 'NN'), ('should', 'MD'), ('try', 'VB'), ('typing', 'VBG'), ('more', 'RBR'), ('complex', 'JJ'), ('words', 'NNS'), ('like', 'IN'), ('u.s.a.', 'NN'), (',', ','), ('m.sc', 'NN'), (',', ','), ('and', 'CC'), ('fan-dom', 'NN'), ('.', '.'), ('maybe', 'RB'), ('nltk', 'JJ'), ('can', 'MD'), ('properly', 'RB'), ('do', 'VB'), ('it', 'PRP'), ('.', '.'), ('i', 'VB'), ("'ve", 'VBP'), ('typed', 'VBN'), ('maybe', 'RB'), ('50', 'CD'), ('words', 'NNS'), ('so', 'RB'), ('far', 'RB'), ('.', '.'), ('i', 'JJ'), ('wonder', 'VBP'), ('if', 'IN'), ('it', 'PRP'), ('matters', 'VBZ'), ('whether', 'IN'), ('i', 'NN'), ('put', 'VBD'), ('spaces', 'NNS'), ('for', 'IN'), ('my', 'PRP$'), ('dots', 'NNS'), ('and', 'CC'), ('commas.like', 'VB'), ('this', 'DT'), ('sentence', 'NN'), ('.', '.'), ('i', 'NN'), ('also', 'RB'), ('want', 'VBP'), ('to', 'TO'), ('test', 'VB'), ('commas', 'NN'), (',', ','), ('so', 'IN'), ('i', 'JJ'), ('will', 'MD'), ('do', 'VB'), ('that', 'DT'), ('in', 'IN'), ('this', 'DT'), ('sentence', 'NN'), ('.', '.'), ('i', 'NN'), ('also', 'RB'), ('want', 'VBP'), ('to', 'TO'), ('test', 'VB'), ('if', 'IN'), ('it', 'PRP'), ('can', 'MD'), ('detect', 'VB'), ('tokens', 'NNS'), ('through', 'IN'), ('other', 'JJ'), ('special', 'JJ'), ('characters', 'NNS'), ('like', 'IN'), ('"', '"'), (';', ':'), ('"', 'POS'), ('.', '.'), ('i', 'NNS'), ('like', 'IN'), ('oranges', 'NNS'), (';', ':'), ('i', 'VB'), ('hate', 'VBP'), ('rap', 'NN'), ('music', 'NN'), (';', ':'), ('is', 'VBZ'), ('this', 'DT'), ('the', 'DT'), ('real', 'JJ'), ('life', 'NN'), ('?', '.'), (';', ':'), ('maybe', 'RB'), ('i', 'NN'), ('should', 'MD'), ('type', 'VB'),

('the', 'DT'), ('rest', 'NN'), ('of', 'IN'), ('words', 'NNS'), ('as', 'RB'), ('complicated', 'VBD'), ('as', 'IN'), ('i', 'NN'), ('can', 'MD'), ('.', '.'), ('hmm', 'VB'), ('...', ':'), ('.', '.'), ('lets', 'NNS'), ('see', 'VBP'), ('how', 'WRB'), ('i', 'JJ'), ('can', 'MD'), ('do', 'VB'), ('that', 'DT'), ('.', '.'), ('i', 'NN'), ('can', 'MD'), ('try', 'VB'), ('detecting', 'VBG'), ('dashes', 'NNS'), ('as', 'RB'), ('well', 'RB'), ('.', '.'), ('rocket-man', 'NN'), ('is', 'VBZ'), ('not', 'RB'), ('the', 'DT'), ('right', 'JJ'), ('way', 'NN'), ('to', 'TO'), ('type', 'VB'), ('that', 'IN'), ('particular', 'JJ'), ('word', 'NN'), ('.', '.'), ('m*a*s*h*', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('hard', 'JJ'), ('word', 'NN'), ('to', 'TO'), ('tokenize', 'VB'), ('wonder', 'VB'), ('how', 'WRB'), ('nltk', 'JJ'), ('does', 'VBZ'), ('.', '.'), ('writing', 'VBG'), ('programming/scripting', 'VBG'), ('languages', 'NNS'), ('is', 'VBZ'), ('probably', 'RB'), ('a', 'DT'), ('good', 'JJ'), ('idea', 'NN'), (':', ':'), ('c', 'JJ'), ('#', '#'), (',', ','), ('c/c++', 'NN'), (',', ','), ('html5', 'NN'), (',', ','), ('python', 'NN'), ('.', '.'), ('tokenizing', 'VBG'), ('seems', 'VBZ'), ('to', 'TO'), ('be', 'VB'), ('the', 'DT'), ('first', 'JJ'), ('step', 'NN'), ('to', 'TO'), ('text', 'VB'), ('analytics', 'NNS'), ('.', '.'), ('proper', 'JJ'), ('tokens', 'VBZ'), ('seems', 'VBZ'), ('like', 'IN'), ('the', 'DT'), ('key', 'NN'), ('for', 'IN'), ('analysing', 'VBG'), ('text', 'NN'), ('.', '.'), ('tokens', 'NNS'), (''', 'POS'), ('properties', 'NNS'), ('must', 'MD'), ('be', 'VB'), ('perfect', 'JJ'), ('for', 'IN'), ('error-free', 'JJ'), ('operation', 'NN'), ('.', '.'), ('stuck', 'VBN'), ('in', 'IN'), ('a', 'DT'), ('landslide', 'NN'), ('no', 'DT'), ('escape', 'NN'), ('from', 'IN'), ('reality', 'NN'), ('!', '.')]

Analysis: 'Nltk' is supposed to be a noun but in the first instance it correctly derives it but in the next instance it incorrectly assigns it as an adjective.
'I' token is hard for tagger. In the highlighted ones tagger has assigned all kinds of wrong tags including a noun in one case.
'commas.like' has been assigned verb wrongly.
'Escape' token is a verb in this context but it has labeled it as a noun.

Q2:a

New file text:

This is another complicated word-set with various twists and turns and weird words like fishing to try trick the stemmer/lemmetesir. https:/www.gmail.com is an link for the Google mail system called g-mail. If john goes to fish he will bring fishes if he fishes for more than an hour. This is one fragment;This is the next fragment of text. If i dont get a break tomorrow I will break the table till it breaks. Broke back mountain is a madeup name to test this system. The rounds turned on me when bill had played his turn. This turned my stomach giddy. He set the world record by setting his foot over the finish line. He has settled in Cork for the weekend. Canned foods is irritating to my esophagus. J.J Abrahams is a guy who really loves stars. I think I have crossed the 200 word limit time to see what it can do. He went fishing.

Porter stemming on this files tokens:
['thi', 'is', 'anoth', 'complic', 'word-set', 'with', 'variou', 'twist', 'and', 'turn', 'and', 'weird', 'word', 'like', 'fish', 'to', 'tri', 'trick', 'the', 'stemmer/lemmetesir', '.', 'http', ':', '/www.gmail.com', 'is', 'an', 'link', 'for', 'the', 'googl', 'mail', 'system', 'call', 'g-mail', '.', 'if', 'john', 'goe', 'to', 'fish', 'he', 'will', 'bring', 'fish', 'if', 'he', 'fish', 'for', 'more', 'than', 'an', 'hour', '.', 'thi', 'is', 'one', 'fragment', ';', 'thi', 'is', 'the', 'next', 'fragment', 'of', 'text', '.', 'if', 'i', 'dont', 'get', 'a', 'break', 'tomorrow', 'i', 'will', 'break', 'the', 'tabl', 'till', 'it', 'break', '.', 'broke', 'back', 'mountain', 'is', 'a', 'madeup', 'name', 'to', 'test', 'thi', 'system', '.', 'the', 'round', 'turn', 'on', 'me', 'when', 'bill', 'had', 'play', 'hi', 'turn', '.', 'thi', 'turn', 'my', 'stomach', 'giddi', '.', 'he', 'set', 'the', 'world', 'record', 'by', 'set', 'hi', 'foot', 'over', 'the', 'finish', 'line', '.', 'he', 'ha', 'settl', 'in', 'cork', 'for', 'the', 'weekend', '.', 'can', 'food', 'is', 'irrit', 'to', 'my', 'esophagu', '.', 'j.j', 'abraham', 'is', 'a', 'guy', 'who', 'realli', 'love', 'star', '.', 'i', 'think', 'i', 'have', 'cross', 'the', '200', 'word', 'limit', 'time', 'to', 'see', 'what', 'it', 'can', 'do', '.', 'he', 'went', 'fish', '.']

Analysis: This method is brutal and the output seems inaccurate . From the start itself;
‘this’ is cut to thi, ‘another’ is cut to ‘anoth’ ,’complicated’ is cut to ‘complic’, ‘various’ is cut to ‘variou’
The next sentence is a bit better. But overall there are still a lot of errors .
I am skeptic this method is actually useful because of the no. of errors it makes in a small file.

Q2:b

Applying wordnet lemmatizer on pos tagged text:
['this', 'n']this['is', 'v']be['another', 'n']another['complicated', 'n']complicated['word-set', 'n']word-set['with', 'n']with['various', 'n']various['twists', 'n']twist['and', 'n']and['turns', 'n']turn['and', 'n']and['weird', 'n']weird['words', 'n']word['like', 'n']like['fishing', 'n']fishing['to','n']to['try', 'n']try['trick', 'n']trick['the', 'n']the['stemmer/lemmetesir', 'n']stemmer/lemmetesir['.', 'n'].['https', 'n']http[':', 'n']:['/www.gmail.com', 'n']/www.gmail.com['is', 'v']be['an', 'n']an['link', 'n']link['for', 'n']for['the', 'n']the['google', 'n']google['mail', 'n']mail['system', 'n']system['called', 'n']called['g-mail', 'n']g-mail['.', 'n'].['if', 'n']if['john', 'n']john['goes', 'v']go['to', 'n']to['fish', 'n']fish['he', 'n']he['will', 'n']will['bring', 'n']bring['fishes', 'n']fish['if', 'n']if['he', 'n']he['fishes', 'v']fish['for', 'n']for['more', 'n']more['than', 'n']than['an', 'n']an['hour', 'n']hour['.', 'n'].['this', 'n']this['is', 'v']be['one', 'n']one['fragment', 'n']fragment[';', 'n'];['this', 'n']this['is', 'v']be['the', 'n']the['next', 'n']next['fragment', 'n']fragment['of', 'n']of['text', 'n']text['.', 'n'].['if', 'n']if['i', 'n']i['dont', 'n']dont['get', 'n']get['a', 'n']a['break', 'n']break['tomorrow', 'n']tomorrow['i', 'n']i['will', 'n']will['break', 'n']break['the', 'n']the['table', 'n']table['till', 'n']till['it', 'n']it['breaks', 'v']break['.', 'n'].['broke', 'v']break['back', 'n']back['mountain', 'n']mountain['is', 'v']be['a', 'n']a['madeup', 'n']madeup['name', 'n']name['to', 'n']to['test', 'n']test['this', 'n']this['system', 'n']system['.', 'n'].['the', 'n']the['rounds', 'n']round['turned', 'v']turn['on', 'n']on['me', 'n']me['when', 'n']when['bill', 'n']bill['had', 'v']have['played', 'n']played['his', 'n']his['turn', 'n']turn['.', 'n'].['this', 'n']this['turned', 'v']turn['my', 'n']my['stomach', 'n']stomach['giddy', 'n']giddy['.', 'n'].['he', 'n']he['set', 'v']set['the', 'n']the['world', 'n']world['record', 'n']record['by', 'n']by['setting', 'v']set['his', 'n']his['foot', 'n']foot['over', 'n']over['the', 'n']the['finish', 'n']finish['line', 'n']line['.', 'n'].['he', 'n']he['has', 'v']have['settled', 'n']settled['in', 'n']in['cork', 'n']cork['for', 'n']for['the', 'n']the['weekend', 'n']weekend['.', 'n'].['canned', 'n']canned['foods', 'n']food['is', 'v']be['irritating', 'v']irritate['to', 'n']to['my', 'n']my['esophagus', 'n']esophagus['.', 'n'].['j.j', 'n']j.j['abrahams', 'n']abraham['is', 'v']be['a', 'n']a['guy', 'n']guy['who', 'n']who['really', 'n']really['loves', 'v']love['stars', 'n']star['.', 'n'].['i', 'n']i['think', 'n']think['i', 'n']i['have', 'n']have['crossed', 'n']crossed['the', 'n']the['200',

'n']200['word', 'n']word['limit', 'n']limit['time', 'n']time['to', 'n']to['see', 'n']see['what', 'n']what['it', 'n']it['can', 'n']can['do', 'n']do['.', 'n'].['he', 'n']he['went', 'v']go['fishing', 'n']fishing['.', 'n'].

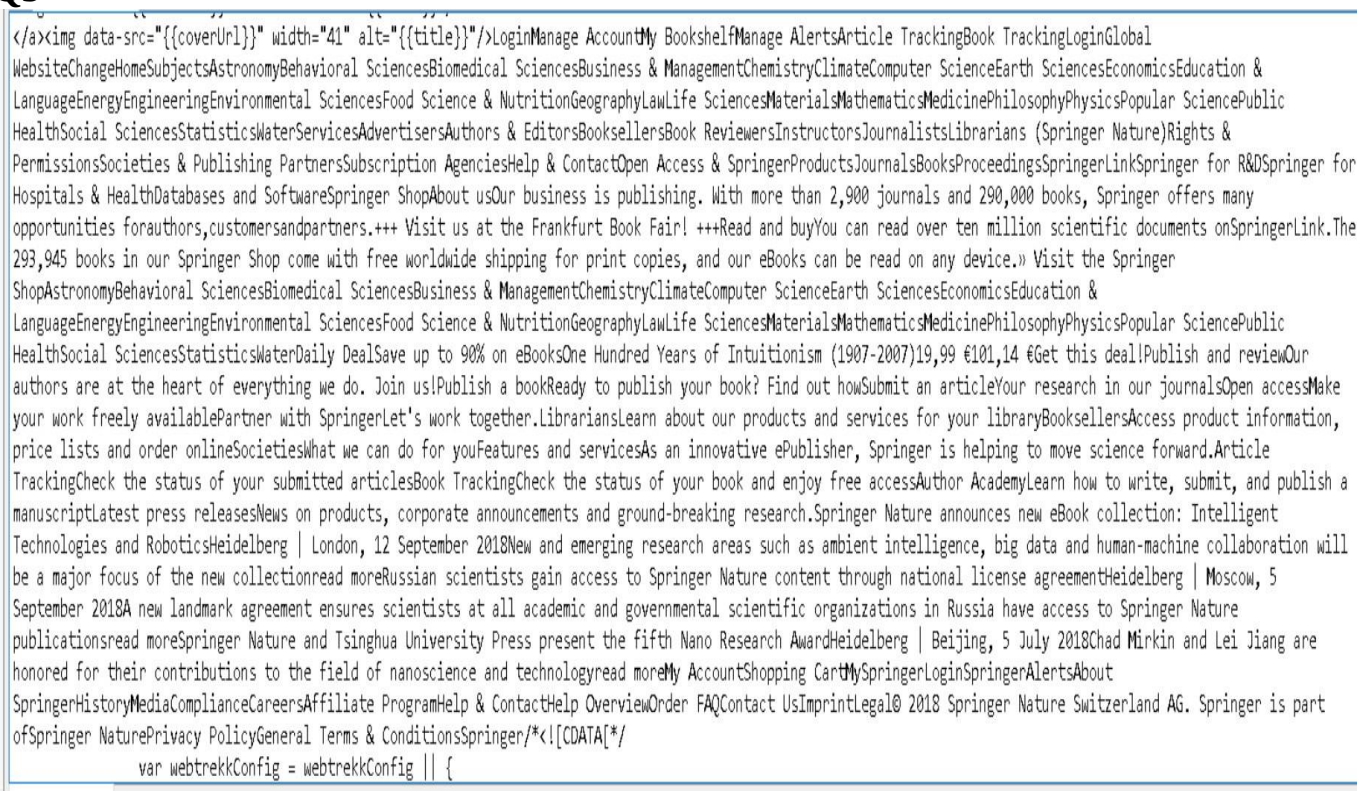
Analysis:

'broke' is assigned as break but the original text refers to' broke back mountain'. This can be fixed with the big data approach to catch majority of words with special meaning and then correctly assign labels.
Some labels like canned aren't converted to can
For the most part the labels are assigned properly in this technique

Q2c

I personally liked the output of wordnet lemmatizer better because it saved the important context. At the same time it also managed to be more accurate. But in terms of analyzing huge amounts of text it is probably expensive to use.
In such cases porter stemming will be lesser expensive to run but will tend to be less accurate.

Q3



Analysis: Useful text is extracted by beautiful soup. But some config text at the header and footer sneak in. Also it has some issues with spacing between styled texts .The site that was analysed was 'https://www.springer.com/gp'