Text Inspector - API version 2

Introduction

This document describes the current programmatic API (version 2) that is made available to some users of the Text Inspector product created by WebLingua.

The document is intended for a technical audience, and assumes an understanding of RESTful APIs, HTTPS, JSON and other technical details.

Further information

For more information or to discuss getting access to the Text Inspector API, please contact your Web Lingua or Text Inspector account manager, or mail the Text Inspector help team: textinspectorhelpteam@gmail.com

Changes for API v2 compared to v1

1. JSON key names in the API responses have been changed to be consistently in camelCase syntax - previously there was a mixture of syntax styles used for the key names.
2. Friendly names have been added to all response values to aid in the understanding and presentation of results.
3. The “tiprofile” parser has been renamed “scorecard” to be consistent with the website.
4. A new “scorecardrating” method has been added to retrieve ***just*** the overall score card rating.
5. Version number in all request URLs changed from v1 to v2

Using the API

In order to use the API, if you haven’t already you will need to register on the main Text Inspector site and then request API access via the “API for Developers” page on the site.

Please do this by going to the following two links.

1. Text Inspector Registration: https://textinspector.com/account/register
2. Request API Access: https://textinspector.com/help/?page_id=974

API URI

The base URI for all API requests is:

`https://textinspector.com/api/v2/`

e.g.

```
GET https://textinspector.com/api/v2/createsession
```

Text Analysis API flow

To analyse a piece of text and obtain the analysis results for the different Text Inspector parsers requires multiple requests to be made to the API.

You first need to authenticate with the API to create a user session (*createsession*) and obtain a "sessionid". That "sessionid" is then passed in all subsequent API requests.

During a session you can submit any number of texts for analysis - if you want to analyse multiple texts you do not need to create a new session for each.

Once you have a session you then submit your text for analysis. On successful submission of your text you will be returned what we call a "context id" (or "ctxId"). This context id is unique for that piece of texts analysis in your session, and is used for obtaining the analysis results from the different Text Inspector parsers.

Once you have a context id you would then request the results from the parser(s) that you are wanting the analysis results for.

An example flow of API requests might be:

1. Create session and obtain your "sessionid"
2. Submit new text for analysis and obtain the analysis "ctxId"
3. Get statistics parser results
4. Get tagger parser results
5. Submit second text for analysis and obtain the analysis "ctxId"
6. Get scorecard parser results
7. etc.

API methods

`GET /api/v2/createsession`

To use the API you need to first authenticate with the API to create a user session - doing this will return you a **sessionid**. That session ID needs to be remembered and passed as a cookie header to all subsequent requests in the session.

During a session you can analyse and get the results for any number of different texts.

To authenticate and get the Session ID you have to make a ***createsession*** API call

Authentication is done with HTTP Basic Authentication, and as such you can supply the username/password either inline in the URL, or in an “Authentication” HTTP header.

Inline username/password

e.g. https://username:password@textinspector.com/api/v2/createsession

When passing the username/password inline in the URL, both username and password need to be URL (percent) encoded.

e.g. the “@” in the email username needs to be encoded as “%40”.

In PHP for example you would use the urlencode() function to do this.

As an example, if your username was “test@example.com” and your password “abc!123” then these encoded and added inline to the createsession APU URL would look like:

https://test%40example.com:abc%21123@textinspector.com/api/v2/createsession

Curl example
`curl -X GET “https://{urlencoded_username}:{urlencoded_password}@textinspector.com/api/v2/createsession" -H "accept: application/json" -L`

“Authentication” HTTP header

To send the username/password in an “Authentication” HTTP header the username and password need to be concatenated with a colon (“:”) and then Base64 encoded.

e.g.

“test@example.com:abc!123” -> base64 encode -> “dGVzdEBleGFtcGxlLmNvbTphYmMhMTIz”

The resulting encoded string is added as an http “Authorization” header to your request, with the value being “Basic [encoded_string]”.

e.g.

Authorization: Basic dGVzdEBleGFtcGxlLmNvbTphYmMhMTIz

Curl example

```
curl -X GET “https://textinspector.com/api/v2/createsession" -H
"accept: application/json" -H "Authorization: Basic
dGVzdEBleGFtcGxlLmNvbTphYmMhMTIz" -L
```

The response from a createsession API call will return you a “**sessionid**”.

Example response

```
{

    "sessionid" : "7319991722849942216896943646052423328"

}
```

This “**sessionid**” should be stored and needs to be passed in all subsequent requests during your session as a “textinspector.session” cookie http header.

E.g.

```
Cookie: textinspector.session={sessionid};
```

POST /api/v2/newanalysis

Submit a document for processing.

You can submit one or multiple texts at the same time. If submitting multiple texts then they should be separated by a “delimiter” character (specified in the request, defaulting to “#”), and the “split” parameter should be set to “1”.

The “newanalysis” API call returns a **ctxId** and the count of the number of documents processed. Use the **ctxId** and appropriate document number(s) in subsequent requests to get results from different parsers.

Curl example

```
curl -X POST "https://textinspector.com/api/v2/newanalysis" -H
"Content-Type: application/json" -H "accept: application/json" -H
"Cookie: textinspector.session={sessionid};" -L --data
'{"text":"Try+the+tool+out+with+this+paragraph+of+text+or+you+can+r
eplace+this+whole+text+with+some+text+of+your+choosing.%23This+is+a
+second+document","delimiter":"%23","split":"1","textmode":"Writing"}
'
```

Example response

```
{
   "response" : {
      "ctxId" : "AFD52E08-4DD5-11E8-9E8C-26C6F64FBF04",
      "doc_count" : 2,
      "resultType" : "api_start",
      "errors" : []
   },
   "name" : "api_start",
   "type" : "api_start"
}
```

JSON input parameters - submitted in the request body		
name	**use**	**example**
`text` `(required)`	The document text to process.	`"text":"This is a test doc#And this is another"`
`split`	Set to “1” if submitting more than one text - multiple texts should be separated with the “delimiter” you specify. Default=0	`"split":"0"`
`delimiter`	The delimiter to use if submitting multiple texts. Only required if also using "split". Default=#	`"delimiter":"#"`
`textmode`	The text mode for the analysis - Writing, Reading, Listening. Default=Writing	`"textmode":`"Writing"

JSON response values		
name	**use**	**example**
`ctxId`	The Context to use for future requests. A context can contain >1 document	`"ctxId":"AFD52E08-5DD5-11E8-9E8C-26C6F64FBF04"`
`doc_count`	The number of documents found in the input text using the delimiters specified, if any	`"doc_count":"2"`
`resultType`	The type of response being sent	`"resultType":"api_start"`

Having submitted a document for processing, you can then request results from the different parsers.

To do this use the **ctxId** returned from the newanalysis POST and specify which document number that you'd like to parse.

Note: if you submitted a single text then the document number that you use in all other API calls will always be “1”. If however you submitted multiple texts with a delimiter character then the document numbers will allow you to get the parser results for each of the separate documents.

E.g.

First text#second text#third text

Results in 3 documents

doc1#doc2#doc3

GET /api/v2/{ctxID}/doc{docNum}/statistics

Get the results for the “Statistics” parser

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctxID}/doc{docNum}/statistics"
-H "accept: application/json" -H "Cookie:
textinspector.session={sessionid};" -L
```

Example response - statistics

```
{

   "name" : "Statistics",

   "type" : "statistics",

   "response" : {

      "ctxId" : "AC07A97C-9C1C-11E9-AB5C-DDF6D7F2BACF",

      "summary" : {

         "avgSentenceLength" : {

            "friendlyName" : "Average sentence length (words)",

            "value" : "16.67"

         },
        "tokenCount" : {
           "value" : "100",
           "friendlyName" : "Token count (excluding numbers)"
        },
        "wordsMoreThanTwoSyllablesPercentage" : {
           "friendlyName" : "Words with more than 2 syllables -
Percentage",
           "value" : "10.00"
        },
        "sentenceCount" : {
           "value" : "6",
           "friendlyName" : "Sentence Count"
        },
        "syllablesPerHundredWords" : {
           "friendlyName" : "Syllables per 100 words",
           "value" : "146.00"
        },
        "averageSyllablesPerSentence" : {
           "friendlyName" : "Average syllables per sentence",
           "value" : "24.33"
        },
        "excludeDigits" : {
           "friendlyName" : "Digits Excluded?",
           "value" : "false"
        },
        "typeTokenRatio" : {
           "value" : "0.69",
            "friendlyName" : "Type/token ratio"
         },
         "wordsMoreThanTwoSyllables" : {
            "value" : "10",
            "friendlyName" : "Words with more than 2 syllables"
         },
         "numberCount" : {
            "value" : "0",
            "friendlyName" : "Number Count"
         },
         "averageSyllablesPerWord" : {
            "value" : "1.46",
            "friendlyName" : "Average syllables per word"
         },
         "syllableCount" : {
            "friendlyName" : "Syllable Count",
            "value" : "146"
         },
         "totalDigitsCount" : {
            "friendlyName" : "Total Digits Count",
            "value" : "0"
         },
         "typeCount" : {
            "value" : "69",
            "friendlyName" : "Type count (unique tokens, excluding
numbers)"
         }
```

```
      },
      "documentId" : "doc1",
      "readabilityScores" : {
         "gunningFogIndex" : {
            "friendlyName" : "Gunning Fog index",
            "value" : "10.67"
         },
         "friendlyName" : "Readability Scores",
         "fleschReadingEase" : {
            "value" : "66.40",
            "friendlyName" : "Flesch Reading Ease"
         },
         "fleschKincaidGrade" : {
            "friendlyName" : "Flesch-Kincaid Grade",
            "value" : "8.14"
         }
      }
   },
   "exists" : 1
}
```

GET /api/v2/{ctxID}/doc{docNum}/errors

Get the results for the “Errors” parser.

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctdId}/doc{docNum}/errors" -H
"accept: application/json" -H "Cookie:
```

```
textinspector.session={sessionid};" -L
```

Example response - errors

```
{
   "name" : "Errors",
   "type" : "errors",
   "response" : {
      "ctxId" : "E44235B4-9C1C-11E9-AB5C-DDF6D7F2BACF",
      "documentId" : "doc1",
      "summary" : {
         "total" : {
            "friendlyName" : "Spelling errors total",
            "value" : "1"
         },
         "perSentence" : {
            "friendlyName" : "Spelling errors per sentence",
            "value" : "0.17"
         },
         "perHundredWords" : {
            "value" : "1.00",
            "friendlyName" : "Spelling errors per 100 words"
         }
      },
      "spellingErrors" : [
         "Analyse"
      ]
   },
   "exists" : 1

}
```

GET /api/v2/{ctxID}/doc{docNum}/diversity

Get the results for the “Lexical Diversity” parser.

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctdId}/doc{docNum}/diversity" -H
"accept: application/json" -H "Cookie:
textinspector.session={sessionid};" -L
```

Example response - diversity

```
{

   "exists" : 1,

   "response" : {

      "ctxId" : "13C2AB3E-9C1D-11E9-AB5C-DDF6D7F2BACF",

      "lexicalDiversity" : {

         "friendlyName" : "Lexical Diversity",

         "vocd" : {

            "value" : "79.91",

            "friendlyName" : "VOCD Lexical Diversity"

         },
         "mtld" : {
            "friendlyName" : "MTLD Lexical Diversity",
            "value" : "65.68"
         }
      },
      "documentId" : "doc1"
   },
   "type" : "diversity",
   "name" : "Lexical Diversity"
}
```

GET /api/v2/{ctxID}/doc{docNum}/tagger

Get the results for the “Tagger” parser.

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctdId}/doc{docNum}/tagger" -H
"accept: application/json" -H "Cookie:
textinspector.session={sessionid};" -L
```

Example response - tagger

```
{
   "type" : "tagger",
   "name" : "Tagger",
   "exists" : 1,
   "response" : {
      "documentId" : "doc1",
      "partsOfSpeech" : {
         "vvg" : {
            "friendlyName" : "VVG",
            "uniqueTokens" : {
               "friendlyName" : "Unique Tokens",
               "value" : "3"
            },
            "description" : "verb, gerund/participle",
            "tokens" : {
               "friendlyName" : "Tokens",
               "value" : "5"
            }
         },
         "vbz" : {
            "tokens" : {
               "value" : "2",
               "friendlyName" : "Tokens"
            },
            "description" : "verb be, pres, 3rd p. sing",
            "uniqueTokens" : {
               "value" : "1",
               "friendlyName" : "Unique Tokens"
            },
            "friendlyName" : "VBZ"
         },
           "co" : {
              "uniqueTokens" : {
                 "friendlyName" : "Unique Tokens",
                 "value" : "3"
              },
              "description" : "coordinating conjunction",
              "friendlyName" : "CO",
              "tokens" : {
                 "friendlyName" : "Tokens",
                 "value" : "9"
              }
           },
           "nn" : {
              "tokens" : {
                 "friendlyName" : "Tokens",
                 "value" : "20"
              },
              "description" : "noun, singular or mass",
              "uniqueTokens" : {
                 "value" : "15",
                 "friendlyName" : "Unique Tokens"
              },
              "friendlyName" : "NN"
           },
           "md" : {
              "tokens" : {
                 "friendlyName" : "Tokens",
               "value" : "1"
            },
            "description" : "modal",
            "uniqueTokens" : {
               "friendlyName" : "Unique Tokens",
               "value" : "1"
            },
            "friendlyName" : "MD"
         },
         "to" : {
            "uniqueTokens" : {
               "friendlyName" : "Unique Tokens",
               "value" : "1"
            },
            "description" : "to",
            "friendlyName" : "TO",
            "tokens" : {
               "friendlyName" : "Tokens",
               "value" : "2"
            }
         },
         "vvn" : {
            "friendlyName" : "VVN",
            "uniqueTokens" : {
               "value" : "1",
               "friendlyName" : "Unique Tokens"
            },
           "description" : "verb, past participle",
           "tokens" : {
               "value" : "1",
               "friendlyName" : "Tokens"
           }
       },
       "vv" : {
           "description" : "verb, base form",
           "uniqueTokens" : {
               "value" : "7",
               "friendlyName" : "Unique Tokens"
           },
           "friendlyName" : "VV",
           "tokens" : {
               "value" : "7",
               "friendlyName" : "Tokens"
           }
       },
       "uh" : {
           "friendlyName" : "UH",
           "uniqueTokens" : {
               "value" : "1",
               "friendlyName" : "Unique Tokens"
           },
           "description" : "interjection",
           "tokens" : {
               "friendlyName" : "Tokens",
```

},

```
            "value" : "1"
         }
      },
      "dt" : {
         "uniqueTokens" : {
            "friendlyName" : "Unique Tokens",
            "value" : "3"
         },
         "description" : "determiner",
         "friendlyName" : "DT",
         "tokens" : {
            "value" : "5",
            "friendlyName" : "Tokens"
         }
      },
      "nps" : {
         "uniqueTokens" : {
            "value" : "1",
            "friendlyName" : "Unique Tokens"
         },
         "description" : "proper noun, plural",
         "friendlyName" : "NPS",
         "tokens" : {
            "value" : "1",
            "friendlyName" : "Tokens"
         }
      },
        "vvz" : {
           "description" : "verb, present 3d p. sing.",
           "uniqueTokens" : {
              "value" : "1",
              "friendlyName" : "Unique Tokens"
           },
           "friendlyName" : "VVZ",
           "tokens" : {
              "value" : "1",
              "friendlyName" : "Tokens"
           }
        },
        "nns" : {
           "friendlyName" : "NNS",
           "uniqueTokens" : {
              "value" : "5",
              "friendlyName" : "Unique Tokens"
           },
           "description" : "noun plural",
           "tokens" : {
              "value" : "7",
              "friendlyName" : "Tokens"
           }
        },
        "in" : {
           "uniqueTokens" : {
              "value" : "7",
          "friendlyName" : "Unique Tokens"
       },
       "description" : "preposition/subord. conj.",
       "friendlyName" : "IN",
       "tokens" : {
          "friendlyName" : "Tokens",
          "value" : "12"
       }
    },
    "rb" : {
       "tokens" : {
          "friendlyName" : "Tokens",
          "value" : "5"
       },
       "description" : "adverb",
       "uniqueTokens" : {
          "friendlyName" : "Unique Tokens",
          "value" : "5"
       },
       "friendlyName" : "RB"
    },
    "qq" : {
       "uniqueTokens" : {
          "value" : "1",
          "friendlyName" : "Unique Tokens"
       },
       "description" : "particle",
         "friendlyName" : "QQ",
         "tokens" : {
            "value" : "1",
            "friendlyName" : "Tokens"
         }
      },
      "jj" : {
         "friendlyName" : "JJ",
         "description" : "adjective",
         "uniqueTokens" : {
            "value" : "6",
            "friendlyName" : "Unique Tokens"
         },
         "tokens" : {
            "friendlyName" : "Tokens",
            "value" : "6"
         }
      },
      "pp" : {
         "tokens" : {
            "value" : "3",
            "friendlyName" : "Tokens"
         },
         "description" : "possessive pronoun",
         "uniqueTokens" : {
            "friendlyName" : "Unique Tokens",
            "value" : "2"
      },
      "friendlyName" : "PP$"
   },
   "dat" : {
      "tokens" : {
         "value" : "3",
         "friendlyName" : "Tokens"
      },
      "uniqueTokens" : {
         "friendlyName" : "Unique Tokens",
         "value" : "1"
      },
      "description" : "determiner, article",
      "friendlyName" : "DAT"
   },
   "np" : {
      "tokens" : {
         "friendlyName" : "Tokens",
         "value" : "5"
      },
      "uniqueTokens" : {
         "value" : "4",
         "friendlyName" : "Unique Tokens"
      },
      "description" : "proper noun, singular",
      "friendlyName" : "NP"
   }
```

```
      },
      "summary" : {
         "elements" : {
            "value" : "72",
            "friendlyName" : "Elements (includes homonyms)"
         },
         "coverageNotCounted" : {
            "friendlyName" : "Coverage - Not Counted",
            "value" : "0"
         },
         "totalNounElements" : {
            "friendlyName" : "Total Noun Elements",
            "value" : "25"
         },
         "types" : {
            "value" : "69",
            "friendlyName" : "Types (unique items)"
         },
         "totalVerbalElements" : {
            "value" : "14",
            "friendlyName" : "Total Verbal Elements"
         },
         "tokenRatio" : {
            "friendlyName" : "Type/token ratio",
            "value" : "0.69"
         },
         "tokens" : {
            "friendlyName" : "Tokens (total items)",
            "value" : "100"
         },
         "verbalElementsPerSentence" : {
            "value" : "2.33",
            "friendlyName" : "Verbal Elements per Sentence"
         },
         "tokensPerType" : {
            "value" : "1.45",
            "friendlyName" : "Tokens per type"
         },
         "coverageCounted" : {
            "friendlyName" : "Coverage - Counted",
            "value" : "100"
         },
         "coverage" : {
            "value" : "100.00",
            "friendlyName" : "Coverage"
         },
         "nounElementsPerSentence" : {
            "value" : "4.17",
            "friendlyName" : "Noun Elements per Sentence"
         }
      },
      "ctxId" : "39A03A6A-9C1D-11E9-AB5C-DDF6D7F2BACF"
   }
}
```

GET /api/v2/{ctxID}/tagger/taggeddata

Get the tagged data for the “Tagger” parser. This will give you the results for all documents in the analysis - you don’t need to request each document separately.

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctdId}/tagger/taggeddata" -H
"accept: application/json" -H "Cookie:
textinspector.session={sessionid};" -L
```

Example response - tagger tagged data

```
{

  "doc1": [

    {

      "Tag": "VV",

      "Token": "Try"

    },

    {

      "Tag": "DAT",

      "Token": "the"

    },

    {

      "Token": "tool",

      "Tag": "NN"

    },

    {

      "Token": "out",
      "Tag": "QQ"
    },
    {
      "Token": "with",
      "Tag": "IN"
    },
    {
      "Token": "this",
      "Tag": "DT"
    },
    {
      "Token": "paragraph",
      "Tag": "NN"
    },
    {
      "Token": "of",
      "Tag": "IN"
    },
    {
      "Tag": "NN",
      "Token": "text"
    },
    {
      "Token": "or",
      "Tag": "CO"
    },
    {
      "Tag": "PP",
      "Token": "you"
    },
    {
      "Token": "can",
      "Tag": "MD"
    },
    {
      "Token": "replace",
      "Tag": "VV"
    },
    {
      "Tag": "DT",
      "Token": "this"
    },
    {
      "Tag": "JJ",
      "Token": "whole"
    },
    {
      "Tag": "NN",
      "Token": "text"
    },
    {
      "Token": "with",
      "Tag": "IN"
    },
    {
      "Token": "some",
      "Tag": "DT"
    },
    {
      "Token": "text",
      "Tag": "NN"
    },
    {
      "Tag": "IN",
      "Token": "of"
    },
    {
      "Tag": "PP$",
      "Token": "your"
    },
    {
      "Tag": "NN",
      "Token": "choosing"
    }
  ],
  "doc2": [
    {
      "Tag": "DT",
      "Token": "This"
    },
    {
      "Tag": "VBZ",
      "Token": "is"
    },
    {
      "Token": "a",
      "Tag": "DAT"
    },
    {
      "Tag": "JJ",
      "Token": "second"
    },
    {
      "Tag": "NN",
      "Token": "document"
    }
  ]
}
```

GET /api/v2/{ctxID}/doc{docNum}/metadiscourse

Get the results for the “Metadiscourse” parser.

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctdId}/doc{docNum}/metadiscourse
" -H "accept: application/json" -H "Cookie:
textinspector.session={sessionid};" -L
```

Example response - metadiscourse

```
{
   "type" : "metadiscourse",
   "name" : "Metadiscourse",
   "exists" : 1,
   "response" : {
      "documentId" : "doc1",
      "summary" : {
         "unlisted" : {
            "tokensPercent" : {
               "value" : "86.00",
               "friendlyName" : "Tokens Percent"
            },
            "friendlyName" : "Unlisted",
            "types" : {
               "value" : "62",
               "friendlyName" : "Types"
            },
            "tokens" : {
               "value" : "86",
               "friendlyName" : "Tokens"
            },
            "typesPercent" : {
               "value" : "89.86",
               "friendlyName" : "Types Percent"
            }
```

```
        },
        "personMarker" : {
           "tokensPercent" : {
              "value" : "1.00",
              "friendlyName" : "Tokens Percent"
           },
           "friendlyName" : "Person marker",
           "tokens" : {
              "friendlyName" : "Tokens",
              "value" : "1"
           },
           "types" : {
              "value" : "1",
              "friendlyName" : "Types"
           },
           "typesPercent" : {
              "value" : "1.45",
              "friendlyName" : "Types Percent"
           }
        },
        "total" : {
           "friendlyName" : "Totals",
           "tokensPercent" : {
              "friendlyName" : "Tokens Percent",
              "value" : "14.00"
           },
           "typesPercent" : {
            "value" : "10.14",
            "friendlyName" : "Types Percent"
         },
         "types" : {
            "value" : "7",
            "friendlyName" : "Types"
         },
         "tokens" : {
            "value" : "14",
            "friendlyName" : "Tokens"
         }
      },
      "endophoric" : {
         "types" : {
            "value" : "1",
            "friendlyName" : "Types"
         },
         "tokens" : {
            "friendlyName" : "Tokens",
            "value" : "1"
         },
         "typesPercent" : {
            "friendlyName" : "Types Percent",
            "value" : "1.45"
         },
         "tokensPercent" : {
            "friendlyName" : "Tokens Percent",
         "value" : "1.00"
      },
      "friendlyName" : "Endophoric"
   },
   "logicalConnective" : {
      "typesPercent" : {
         "friendlyName" : "Types Percent",
         "value" : "4.35"
      },
      "tokens" : {
         "value" : "9",
         "friendlyName" : "Tokens"
      },
      "types" : {
         "friendlyName" : "Types",
         "value" : "3"
      },
      "friendlyName" : "Logical connective",
      "tokensPercent" : {
         "friendlyName" : "Tokens Percent",
         "value" : "9.00"
      }
   },
   "relationalMarker" : {
      "tokens" : {
         "value" : "3",
         "friendlyName" : "Tokens"
```

```
            },
            "types" : {
               "value" : "2",
               "friendlyName" : "Types"
            },
            "typesPercent" : {
               "friendlyName" : "Types Percent",
               "value" : "2.90"
            },
            "tokensPercent" : {
               "value" : "3.00",
               "friendlyName" : "Tokens Percent"
            },
            "friendlyName" : "Relational marker"
         }
      },
      "ctxId" : "695AB398-9C1D-11E9-AB5C-DDF6D7F2BACF"
   }
}
```

GET /api/v2/{ctxID}/metadiscourse/taggeddata

Get the tagged data for the “Metadiscourse” parser. This will give you the results for all documents in the analysis - you don’t need to request each document separately.

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctdId}/metadiscourse/taggeddata"
-H "accept: application/json" -H "Cookie:
textinspector.session={sessionid};" -L
```

Example response - metadiscourse tagged data

```
{
  "doc1": [
    {
      "WordList": null,
      "Word": "try"
    },
    {
      "Word": "the",
      "WordList": null
    },
    {
      "Word": "tool",
      "WordList": null
    },
    {
      "Word": "out",
      "WordList": null
    },
    {
      "WordList": null,
      "Word": "with"
    },
    {
      "Word": "this",
      "WordList": null
    },
    {
      "WordList": null,
      "Word": "paragraph"
    },
    {
      "Word": "of",
      "WordList": null
    },
    {
      "Word": "text",
      "WordList": null
    },
    {
      "WordList": "Logical connective",
      "Word": "or"
    },
    {
      "WordList": "Relational marker",
      "Word": "you"
    },
    {
      "WordList": null,
      "Word": "can"
    },
    {
```

```
    "Word": "replace",
    "WordList": null
  },
  {
    "WordList": null,
    "Word": "this"
  },
  {
    "Word": "whole",
    "WordList": null
  },
  {
    "Word": "text",
    "WordList": null
  },
  {
    "WordList": null,
    "Word": "with"
  },
  {
    "WordList": null,
    "Word": "some"
  },
  {
    "WordList": null,
    "Word": "text"
  },
    {
      "WordList": null,
      "Word": "of"
    },
    {
      "WordList": "Relational marker",
      "Word": "your"
    },
    {
      "Word": "choosing",
      "WordList": null
    }
  ],
  "doc2": [
    {
      "Word": "this",
      "WordList": null
    },
    {
      "WordList": null,
      "Word": "is"
    },
    {
      "Word": "a",
      "WordList": null
    },
    {
      "WordList": "Sequencing",
      "Word": "second"
    },
    {
      "Word": "document",
      "WordList": null
    }
  ]
}
```

GET /api/v2/{ctxID}/doc{docNum}/taggedlexical-bnc

Get the results for the “Lexical: BNC” parser.

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctdId}/doc{docNum}/taggedlexical
-bnc" -H "accept: application/json" -H "Cookie:
textinspector.session={sessionid};" -L
```

Example response - taggedlexical-bnc

```
{
   "type" : "taggedlexical-bnc",
   "name" : "Lexis: BNC",
   "exists" : 1,
   "response" : {
      "documentId" : "doc1",
      "summary" : {
         "percentileTokens70th" : {
            "value" : "2580",
            "friendlyName" : "70th Percentile (tokens)"
        },
        "percentileTokens80th" : {
            "value" : "3170",
            "friendlyName" : "80th Percentile (tokens)"
        },
        "percentileTokens60th" : {
            "friendlyName" : "60th Percentile (tokens)",
            "value" : "1390"
        },
        "meanLFCPerToken" : {
            "value" : "2056.73",
            "friendlyName" : "Mean LFC per token"
        },
        "percentileTypes60th" : {
            "friendlyName" : "60th Percentile (types)",
            "value" : "1120"
        },
        "elementsIncludingHomonyms" : {
            "value" : "72",
            "friendlyName" : "Elements (includes homonyms)"
        },
        "percentileTypes80th" : {
            "friendlyName" : "80th Percentile (types)",
            "value" : "3450"
        },
         "percentileTypes70th" : {
            "value" : "2800",
            "friendlyName" : "70th Percentile (types)"
         },
         "totalLexicalTypesFrequencyCount" : {
            "friendlyName" : "Total Lexical Frequency Count
(types)",
            "value" : "164432"
         },
         "totalTokenItems" : {
            "value" : "100",
            "friendlyName" : "Tokens (total items)"
         },
         "uniqueCountedItemTypes" : {
            "value" : "67",
            "friendlyName" : "Types Counted (unique items)"
         },
         "totalCountedTokensItems" : {
            "friendlyName" : "Tokens Counted (total items)",
            "value" : "98"
         },
         "totalLexicalTokensFrequencyCount" : {
            "value" : "201560",
            "friendlyName" : "Total Lexical Frequency Count
(tokens)"
         },
         "uniqueItemTypes" : {
            "friendlyName" : "Types (unique items)",
            "value" : "69"
         },
         "percentileTokens50th" : {
            "value" : "550",
            "friendlyName" : "50th Percentile (tokens)"
         },
         "meanLFCPer100Tokens" : {
            "friendlyName" : "Mean LFC per 100 tokens",
            "value" : "205673.47"
         },
         "percentileTypes50th" : {
            "value" : "540",
            "friendlyName" : "50th Percentile (types)"
         },
         "meanLFCPerType" : {
            "friendlyName" : "Mean LFC per type",
            "value" : "2383.07"
         },
         "coverageTokensCounted" : {
            "friendlyName" : "Coverage - Counted",
            "value" : "98"
         },
         "meanLFCPer100Types" : {
            "value" : "238307.25",
            "friendlyName" : "Mean LFC per 100 types"
         },
         "coverageTokensNotCounted" : {
            "value" : "2",
            "friendlyName" : "Coverage - Not Counted"
         },
         "coverage" : {
            "value" : "98.00",
            "friendlyName" : "Coverage"
         }
      },
      "ctxId" : "4FA10FD2-9C1E-11E9-8312-C7672007AC3E"
   }
}
```

GET /api/v2/{ctxID}/taggedlexical-bnc/taggeddata

Get the tagged data for the “Lexical: BNC” parser. This will give you the results for all documents in the analysis - you don’t need to request each document separately.

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctdId}/taggedlexical-bnc/taggedd
ata" -H "accept: application/json" -H "Cookie:
textinspector.session={sessionid};" -L
```

Example response - lexical BNC tagged data

```
{
  "doc2": [
    {
      "Score": "24",
      "Token": "This"
    },
    {
      "Token": "is",
      "Score": "9"
    },
    {
      "Score": "4",
      "Token": "a"
    },
    {
      "Score": "263",
      "Token": "second"
    },
    {
      "Score": "1942",
      "Token": "document"
    }
  ],
  "doc1": [
    {
      "Token": "Try",
      "Score": "458"
    },
    {
      "Score": "1",
      "Token": "the"
    },
    {
      "Score": "3907",
      "Token": "tool"
    },
    {
      "Token": "out",
      "Score": "70"
    },
    {
      "Token": "with",
      "Score": "16"
    },
    {
      "Score": "24",
      "Token": "this"
    },
    {
      "Token": "paragraph",
      "Score": "3450"
    },
    {
      "Token": "of",
      "Score": "2"
    },
    {
      "Score": "1388",
      "Token": "text"
    },
    {
      "Score": "3169",
      "Token": "or"
    },
    {
      "Score": "15",
      "Token": "you"
    },
    {
      "Token": "can",
      "Score": "51"
    },
    {
      "Score": "2800",
      "Token": "replace"
    },
    {
      "Score": "24",
      "Token": "this"
    },
    {
      "Token": "whole",
      "Score": "424"
    },
    {
      "Token": "text",
      "Score": "1388"
    },
    {
      "Score": "16",
      "Token": "with"
    },
    {
      "Token": "some",
      "Score": "60"
    },
    {
      "Token": "text",
      "Score": "1388"
    },
    {
      "Token": "of",
      "Score": "2"
    },
    {
      "Score": "76",
      "Token": "your"
    },
    {
      "Score": "4488",
      "Token": "choosing"
    },
    {
      "Score": null,

      "Token": "."

    }

  ]

}
```

GET /api/v2/{ctxID}/doc{docNum}/taggedlexical-coca

Get the results for the “Lexical: COCA” parser.

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctdId}/doc{docNum}/taggedlexical
-coca" -H "accept: application/json" -H "Cookie:
textinspector.session={sessionid};" -L
```

Example response - taggedlexical-coca

```
{

   "name" : "Lexis: COCA",

   "type" : "taggedlexical-coca",

   "response" : {

      "ctxId" : "79426E30-9C1E-11E9-8312-C7672007AC3E",

      "summary" : {

         "totalCountedTokensItems" : {

            "value" : "98",

            "friendlyName" : "Tokens Counted (total items)"

         },

         "totalLexicalTokensFrequencyCount" : {
            "friendlyName" : "Total Lexical Frequency Count
(tokens)",

            "value" : "258734"

         },

         "uniqueItemTypes" : {

            "friendlyName" : "Types (unique items)",

            "value" : "69"

         },

         "percentileTokens50th" : {

            "friendlyName" : "50th Percentile (tokens)",

            "value" : "610"

         },

         "meanLFCPer100Tokens" : {

            "value" : "264014.29",

            "friendlyName" : "Mean LFC per 100 tokens"

         },

         "percentileTypes50th" : {

            "value" : "550",

            "friendlyName" : "50th Percentile (types)"

         },

         "meanLFCPerType" : {

            "friendlyName" : "Mean LFC per type",

            "value" : "3208.65"

         },

         "coverageTokensCounted" : {

            "friendlyName" : "Coverage - Counted",

            "value" : "98"
```

```
        },
        "coverageTokensNotCounted" : {
           "friendlyName" : "Coverage - Not Counted",
           "value" : "2"
        },
        "meanLFCPer100Types" : {
           "friendlyName" : "Mean LFC per 100 types",
           "value" : "320865.22"
        },
        "coverage" : {
           "friendlyName" : "Coverage",
           "value" : "98.00"
        },
        "percentileTokens70th" : {
           "value" : "1720",
           "friendlyName" : "70th Percentile (tokens)"
        },
        "percentileTokens80th" : {
           "value" : "3080",
           "friendlyName" : "80th Percentile (tokens)"
        },
        "percentileTokens60th" : {
           "friendlyName" : "60th Percentile (tokens)",
           "value" : "1540"
        },
        "meanLFCPerToken" : {
           "friendlyName" : "Mean LFC per token",
            "value" : "2640.14"
         },
         "percentileTypes60th" : {
            "value" : "1090",
            "friendlyName" : "60th Percentile (types)"
         },
         "elementsIncludingHomonyms" : {
            "value" : "72",
            "friendlyName" : "Elements (includes homonyms)"
         },
         "percentileTypes80th" : {
            "value" : "3260",
            "friendlyName" : "80th Percentile (types)"
         },
         "percentileTypes70th" : {
            "friendlyName" : "70th Percentile (types)",
            "value" : "1950"
         },
         "totalLexicalTypesFrequencyCount" : {
            "value" : "221397",
            "friendlyName" : "Total Lexical Frequency Count
(types)"
         },
         "totalTokenItems" : {
            "friendlyName" : "Tokens (total items)",
            "value" : "100"
         },
         "uniqueCountedItemTypes" : {

            "friendlyName" : "Types Counted (unique items)",

            "value" : "67"

         }

      },

      "documentId" : "doc1"

   },

   "exists" : 1

}
```

GET /api/v2/{ctxID}/taggedlexical-coca/taggeddata

Get the tagged data for the “Lexical: COCA” parser. This will give you the results for all documents in the analysis - you don’t need to request each document separately.

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctdId}/taggedlexical-coca/tagged
data" -H "accept: application/json" -H "Cookie:
textinspector.session={sessionid};" -L
```

Example response - lexical COCA tagged data

```
{

  "doc2": [

    {

      "Score": "20",

      "Token": "This"

    },
    {
      "Token": "is",
      "Score": "10"
    },
    {
      "Score": "4",
      "Token": "a"
    },
    {
      "Score": "358",
      "Token": "second"
    },
    {
      "Score": "3593",
      "Token": "document"
    }
  ],
  "doc1": [
    {
      "Score": "383",
      "Token": "Try"
    },
    {
      "Token": "the",
      "Score": "1"
    },
    {
```

```
    "Score": "2632",
    "Token": "tool"
  },
  {
    "Token": "out",
    "Score": "67"
  },
  {
    "Score": "16",
    "Token": "with"
  },
  {
    "Token": "this",
    "Score": "20"
  },
  {
    "Score": "9386",
    "Token": "paragraph"
  },
  {
    "Token": "of",
    "Score": "3"
  },
  {
    "Score": "1721",
    "Token": "text"
  },
    {
      "Score": "3258",
      "Token": "or"
    },
    {
      "Token": "you",
      "Score": "13"
    },
    {
      "Score": "51",
      "Token": "can"
    },
    {
      "Score": "2898",
      "Token": "replace"
    },
    {
      "Score": "20",
      "Token": "this"
    },
    {
      "Token": "whole",
      "Score": "431"
    },
    {
      "Score": "1721",
      "Token": "text"
    },
    {
      "Token": "with",
      "Score": "16"
    },
    {
      "Token": "some",
      "Score": "69"
    },
    {
      "Token": "text",
      "Score": "1721"
    },
    {
      "Token": "of",
      "Score": "3"
    },
    {
      "Score": "68",
      "Token": "your"
    },
    {
      "Token": "choosing",
      "Score": "4505"
    },
    {
      "Token": ".",
      "Score": null
    }
  ]
}
```

GET /api/v2/{ctxID}/doc{docNum}/academicwordlist

Get the results for the “Lexical: AWL” parser.

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctdId}/doc{docNum}/academicwordl
ist" -H "accept: application/json" -H "Cookie:
textinspector.session={sessionid};" -L
```

Example response - academicwordlist

```
{
   "name" : "Lexis: AWL",
   "type" : "academicwordlist",
   "response" : {
      "documentId" : "doc1",
      "ctxId" : "D8A74472-9C1E-11E9-8312-C7672007AC3E",
      "summary" : {
         "unlisted" : {
            "tokensPercent" : {
               "value" : "83.00",
               "friendlyName" : "Tokens Percent"
```

```
        },
        "types" : {
            "value" : "59",
            "friendlyName" : "Types"
        },
        "tokens" : {
            "value" : "83",
            "friendlyName" : "Tokens"
        },
        "typesPercent" : {
            "value" : "85.51",
            "friendlyName" : "Types Percent"
        },
        "friendlyName" : "Unlisted"
    },
    "awl4" : {
        "types" : {
            "friendlyName" : "Types",
            "value" : "1"
        },
        "tokens" : {
            "friendlyName" : "Tokens",
            "value" : "1"
        },
        "tokensPercent" : {
            "value" : "1.00",
            "friendlyName" : "Tokens Percent"
```

```
      },
      "typesPercent" : {
         "value" : "1.45",
         "friendlyName" : "Types Percent"
      },
      "friendlyName" : "AWL 4"
   },
   "awl1" : {
      "typesPercent" : {
         "friendlyName" : "Types Percent",
         "value" : "2.90"
      },
      "friendlyName" : "AWL 1",
      "types" : {
         "friendlyName" : "Types",
         "value" : "2"
      },
      "tokens" : {
         "friendlyName" : "Tokens",
         "value" : "2"
      },
      "tokensPercent" : {
         "value" : "2.00",
         "friendlyName" : "Tokens Percent"
      }
   },
   "awl2" : {
```

```
            "tokens" : {
               "value" : "10",
               "friendlyName" : "Tokens"
            },
            "types" : {
               "value" : "3",
               "friendlyName" : "Types"
            },
            "tokensPercent" : {
               "friendlyName" : "Tokens Percent",
               "value" : "10.00"
            },
            "friendlyName" : "AWL 2",
            "typesPercent" : {
               "value" : "4.35",
               "friendlyName" : "Types Percent"
            }
         },
         "awl6" : {
            "friendlyName" : "AWL 6",
            "typesPercent" : {
               "value" : "1.45",
               "friendlyName" : "Types Percent"
            },
            "tokensPercent" : {
               "friendlyName" : "Tokens Percent",
               "value" : "1.00"
         },
         "tokens" : {
            "value" : "1",
            "friendlyName" : "Tokens"
         },
         "types" : {
            "friendlyName" : "Types",
            "value" : "1"
         }
      },
      "awlTotal" : {
         "tokensPercent" : {
            "friendlyName" : "Tokens Percent",
            "value" : "17.00"
         },
         "tokens" : {
            "friendlyName" : "Tokens",
            "value" : "17"
         },
         "types" : {
            "friendlyName" : "Types",
            "value" : "10"
         },
         "friendlyName" : "AWL Total",
         "typesPercent" : {
            "value" : "14.49",
            "friendlyName" : "Types Percent"
         }
      },
      "awl8" : {
         "friendlyName" : "AWL 8",
         "typesPercent" : {
            "friendlyName" : "Types Percent",
            "value" : "2.90"
         },
         "tokensPercent" : {
            "friendlyName" : "Tokens Percent",
            "value" : "2.00"
         },
         "tokens" : {
            "value" : "2",
            "friendlyName" : "Tokens"
         },
         "types" : {
            "friendlyName" : "Types",
            "value" : "2"
         }
      },
      "awl7" : {
         "types" : {
            "value" : "1",
            "friendlyName" : "Types"
         },
         "tokens" : {
                "value" : "1",
                "friendlyName" : "Tokens"
             },
             "tokensPercent" : {
                "value" : "1.00",
                "friendlyName" : "Tokens Percent"
             },
             "typesPercent" : {
                "friendlyName" : "Types Percent",
                "value" : "1.45"
             },
             "friendlyName" : "AWL 7"
          }
       }
    },
    "exists" : 1
}
```

GET /api/v2/{ctxID}/academicwordlist/taggeddata

Get the tagged data for the “Lexis: AWL” parser. This will give you the results for all documents in the analysis - you don’t need to request each document separately.

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctdId}/academicwordlist/taggedda
ta" -H "accept: application/json" -H "Cookie:
textinspector.session={sessionid};" -L
```

Example response - lexis AWL tagged data

```
{
  "doc2": [
    {
      "Token": "this",
      "AWL sublist": null
    },
    {
      "Token": "is",
      "AWL sublist": null
    },
    {
      "Token": "a",
      "AWL sublist": null
    },
    {
      "Token": "second",
      "AWL sublist": null
    },
    {
      "AWL sublist": "AWL 3",
      "Token": "document"
    }
  ],
  "doc1": [
    {
      "AWL sublist": null,
      "Token": "try"
    },
    {
      "AWL sublist": null,
      "Token": "the"
    },
    {
      "AWL sublist": null,
      "Token": "tool"
    },
    {
      "Token": "out",
      "AWL sublist": null
    },
    {
      "Token": "with",
      "AWL sublist": null
    },
    {
      "AWL sublist": null,
      "Token": "this"
    },
    {
      "AWL sublist": "AWL 8",
      "Token": "paragraph"
    },
    {

      "AWL sublist": null,

      "Token": "of"

    },

    {

      "AWL sublist": "AWL 2",

      "Token": "text"

    },

    {

      "Token": "or",

      "AWL sublist": null

    },

    {

      "Token": "you",

      "AWL sublist": null

    },

    {

      "AWL sublist": null,

      "Token": "can"

    },

    {

      "Token": "replace",

      "AWL sublist": null

    },

    {

      "AWL sublist": null,

      "Token": "this"
    },
    {
      "Token": "whole",
      "AWL sublist": null
    },
    {
      "AWL sublist": "AWL 2",
      "Token": "text"
    },
    {
      "Token": "with",
      "AWL sublist": null
    },
    {
      "Token": "some",
      "AWL sublist": null
    },
    {
      "AWL sublist": "AWL 2",
      "Token": "text"
    },
    {
      "Token": "of",
      "AWL sublist": null
    },
    {
      "AWL sublist": null,
      "Token": "your"
    },
    {
      "AWL sublist": null,
      "Token": "choosing"
    }
  ]
}
```

GET /api/v2/{ctxID}/doc{docNum}/lexical

Get the results for the “Lexis: EVP” parser.

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctdId}/doc{docNum}/lexical" -H
"accept: application/json" -H "Cookie:
textinspector.session={sessionid};" -L
```

Example response - lexical

```
{
   "type" : "lexical",
   "name" : "Lexis: EVP",
   "exists" : 1,
   "response" : {
      "documentId" : "doc1",
      "summary" : {
        "a2" : {
            "typesPercent" : {
                "value" : "22.06%",
                "friendlyName" : "Types Percent"
            },
            "friendlyName" : "A2",
            "tokensPercent" : {
                "value" : "22.45%",
                "friendlyName" : "Tokens Percent"
            },
            "types" : {
                "friendlyName" : "Types",
                "value" : "15"
            },
            "tokens" : {
                "value" : "22",
                "friendlyName" : "Tokens"
            }
        },
        "unlisted" : {
            "tokens" : {
                "friendlyName" : "Tokens",
                "value" : "6"
            },
            "types" : {
                "value" : "6",
                "friendlyName" : "Types"
            },
            "tokensPercent" : {
               "value" : "6.12%",
               "friendlyName" : "Tokens Percent"
            },
            "friendlyName" : "Unlisted",
            "typesPercent" : {
               "friendlyName" : "Types Percent",
               "value" : "8.82%"
            }
         },
         "b2" : {
            "tokens" : {
               "friendlyName" : "Tokens",
               "value" : "6"
            },
            "types" : {
               "friendlyName" : "Types",
               "value" : "6"
            },
            "tokensPercent" : {
               "value" : "6.12%",
               "friendlyName" : "Tokens Percent"
            },
            "friendlyName" : "B2",
            "typesPercent" : {
               "value" : "8.82%",
            "friendlyName" : "Types Percent"
         }
      },
      "b1" : {
         "friendlyName" : "B1",
         "typesPercent" : {
            "friendlyName" : "Types Percent",
            "value" : "10.29%"
         },
         "tokens" : {
            "value" : "7",
            "friendlyName" : "Tokens"
         },
         "types" : {
            "friendlyName" : "Types",
            "value" : "7"
         },
         "tokensPercent" : {
            "friendlyName" : "Tokens Percent",
            "value" : "7.14%"
         }
      },
      "a1" : {
         "typesPercent" : {
            "friendlyName" : "Types Percent",
            "value" : "48.53%"
         },
            "friendlyName" : "A1",
            "types" : {
               "value" : "33",
               "friendlyName" : "Types"
            },
            "tokens" : {
               "value" : "56",
               "friendlyName" : "Tokens"
            },
            "tokensPercent" : {
               "friendlyName" : "Tokens Percent",
               "value" : "57.14%"
            }
         },
         "c1" : {
            "tokensPercent" : {
               "value" : "1.02%",
               "friendlyName" : "Tokens Percent"
            },
            "tokens" : {
               "friendlyName" : "Tokens",
               "value" : "1"
            },
            "types" : {
               "friendlyName" : "Types",
               "value" : "1"
            },
            "friendlyName" : "C1",
            "typesPercent" : {
               "value" : "1.47%",
               "friendlyName" : "Types Percent"
            }
         }
      },
      "ctxId" : "21B42CDE-9C1F-11E9-8312-C7672007AC3E"
   }
}
```

GET /api/v2/{ctxID}/lexical/taggeddata

Get the tagged data for the “Lexis: EVP” parser. This will give you the results for all documents in the analysis - you don’t need to request each document separately.

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctdId}/lexical/taggeddata" -H
"accept: application/json" -H "Cookie:
textinspector.session={sessionid};" -L
```

Example response - lexical

```
{

  "doc1": [

    {

      "Word": "try",

      "Type": "verb",

      "Score": "A2",

      "Description": "ATTEMPT to attempt to do something"

    },

    {

      "Word": "the",

      "Type": "determiner",

      "Description": "ONLY ONE used before nouns when only one of
something exists",

      "Score": "A1"

    },

    {

      "Type": null,
      "Score": null,
      "Description": null,
      "Word": "tool"
    },
    {
      "Word": "out",
      "Description": "NOT THERE not in the place where you usually
live or work, especially for a short time",
      "Score": "A2",
      "Type": "adverb"
    },
    {
      "Description": "HAVING having or including something",
      "Score": "A1",
      "Type": "preposition",
      "Word": "with"
    },
    {
      "Word": "this",
      "Score": "A1",
      "Description": "ALREADY MENTIONED used to refer to something
that you have already talked about",
      "Type": "determiner"
    },
    {
      "Word": "paragraph",
      "Type": "noun",
      "Description": "a part of a text that usually contains
several sentences and begins on a new line",

      "Score": "B1"

    },

    {

      "Type": "preposition",

      "Score": "A1",

      "Description": "AMOUNT used after words which show an
amount",

      "Word": "of"

    },

    {

      "Type": "noun",

      "Score": "A2",

      "Description": "MOBILE PHONE a text message ",

      "Word": "text"

    },

    {

      "Type": "conjunction",

      "Description": "POSSIBILITIES used between possibilities, or
before the last in a list of possibilities",

      "Score": "A1",

      "Word": "or"

    },

    {

      "Word": "you",

      "Type": "pronoun",

      "Score": "A1",
      "Description": "PERSON/PEOPLE ADDRESSED used to refer to the
person or people you are talking to"

    },

    {

      "Word": "can",

      "Score": "A1",

      "Description": "ABILITY to be able to",

      "Type": "modal verb"

    },

    {

      "Word": "replace",

      "Description": "GET SOMETHING NEW to get something new
because the one you had before has been lost or damaged",

      "Score": "B1",

      "Type": "verb"

    },

    {

      "Type": "determiner",

      "Score": "A1",

      "Description": "ALREADY MENTIONED used to refer to something
that you have already talked about",

      "Word": "this"

    },

    {

      "Word": "whole",

      "Description": "complete, including every part",

      "Score": "A2",

      "Type": "adjective"
    },
    {
      "Score": "A2",
      "Description": "MOBILE PHONE a text message ",
      "Type": "noun",
      "Word": "text"
    },
    {
      "Type": "preposition",
      "Description": "HAVING having or including something",
      "Score": "A1",
      "Word": "with"
    },
    {
      "Type": "determiner",
      "Score": "A1",
      "Description": "UNKNOWN AMOUNT used to refer to an amount of
something without saying exactly how much or how many",
      "Word": "some"
    },
    {
      "Score": "A2",
      "Description": "MOBILE PHONE a text message ",
      "Type": "noun",
      "Word": "text"
    },
    {
      "Word": "of",

      "Type": "preposition",

      "Description": "AMOUNT used after words which show an
amount",

      "Score": "A1"

    },

    {

      "Word": "your",

      "Description": "PERSON/PEOPLE ADDRESSED belonging or relating
to the person or group of people being spoken or written to",

      "Score": "A1",

      "Type": "determiner"

    },

    {

      "Type": "verb",

      "Score": "A1",

      "Description": "to decide what you want from two or more
things or possibilities",

      "Word": "choosing"

    }

  ],

  "doc2": [

    {

      "Type": "determiner",

      "Score": "A1",

      "Description": "ALREADY MENTIONED used to refer to something
that you have already talked about",

      "Word": "this"

    },
    {
      "Description": "I am Spanish/a teacher, etc. used to give
information about someone or something",
      "Score": "A1",
      "Type": "verb",
      "Word": "is"
    },
    {
      "Type": "determiner",
      "Description": "ANY/EVERY used to mean any or every thing or
person of the type you are referring to",
      "Score": "A1",
      "Word": "a"
    },
    {
      "Word": "second",
      "Type": "ordinal number",
      "Score": "A1",
      "Description": "AFTER FIRST immediately after the first and
before any others"
    },
    {
      "Type": "noun",
      "Score": "A2",
      "Description": "OFFICIAL INFORMATION a piece of paper with
official information on it",
      "Word": "document"
    }
  ]
```

```
}
```

GET /api/v2/{ctxID}/doc{docNum}/scorecard

Get the results for the “Scorecard” parser.

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctdId}/doc{docNum}/scorecard" -H
"accept: application/json" -H "Cookie:
textinspector.session={sessionid};" -L
```

Example response - scorecard

```
{

   "response" : {

      "documentId" : "doc1",

      "overallRating" : {

         "friendlyName" : "Overall Rating",

         "cefrLevel" : "C1",

         "percentage" : "62.92",

         "numberOfMetricsUsed" : "16"

      },

      "scoreCard" : {

         "lexicalSophistication" : {

            "evpPercentOfWordsTypesAtA1Level" : {

               "cefrLevel" : "C1+",

               "friendlyName" : "EVP: % of words (types) at A1
level"
```

```
            },
            "evpPercentOfWordsTypesAtB1Level" : {
               "friendlyName" : "EVP: % of words (types) at B1
level",
               "cefrLevel" : "B2"
            },
            "evpPercentOfWordsTokensAtB2Level" : {
               "cefrLevel" : "C2",
               "friendlyName" : "EVP: % of words (tokens) at B2
level"
            },
            "cocaMeanLfcPerType" : {
               "cefrLevel" : "C1+",
               "friendlyName" : "COCA mean LFC per type"
            },
            "friendlyName" : "Lexical Sophistication",
            "evpPercentOfWordsTypesAtB2Level" : {
               "friendlyName" : "EVP: % of words (types) at B2
level",
               "cefrLevel" : "C2"
            },
            "evpPercentOfWordsTokensAtA1Level" : {
               "cefrLevel" : "C2",
               "friendlyName" : "EVP: % of words (tokens) at A1
level"
            }
         },
         "statistics" : {
            "averageSyllablesPerWord" : {
```

```
          "friendlyName" : "Average Syllables per word",
          "cefrLevel" : "C2"
       },
       "averageWordsPerSentence" : {
          "cefrLevel" : "B1+",
          "friendlyName" : "Average words per sentence"
       }
    },
    "readability" : {
       "fleschKincaidReadingGrade" : {
          "friendlyName" : "Flesch Kincaid Reading Grade",
          "cefrLevel" : "C1"
       },
       "gunningFog" : {
          "cefrLevel" : "C1",
          "friendlyName" : "Gunning Fog"
       },
       "fleschKincaidReadingEase" : {
          "cefrLevel" : "C1+",
          "friendlyName" : "Flesch Kincaid Reading Ease"
       },
       "friendlyName" : "Readability"
    },
    "friendlyName" : "Score Card",
    "lexicalSophisticationAcademic" : {
       "awlList1TypesPercent" : {
          "cefrLevel" : "C2",
                "friendlyName" : "AWL list 1 Types %"
            },
            "awlList1TokensPercent" : {
                "friendlyName" : "AWL list 1 Tokens %",
                "cefrLevel" : "C2"
            },
            "academicWordListPercentOfAllAwlWordsTokensInTheText" :
{
                "friendlyName" : "Academic Word List: % of all AWL
words (tokens) in the text",
                "cefrLevel" : "C2"
            },
            "friendlyName" : "Lexical Sophistication (academic)",
            "academicWordListPercentOfAllAwlWordsTypesInTheText" :
{
                "friendlyName" : "Academic Word List: % of all AWL
words (types) in the text",
                "cefrLevel" : "C2"
            }
        },
        "lexicalDiversity" : {
            "lexicalDiversityMtld" : {
                "friendlyName" : "Lexical Diversity (MTLD)",
                "cefrLevel" : "A1+"
            },
            "friendlyName" : "Lexical Diversity"
        }
    },
    "ctxId" : "9B4B3D80-9C1F-11E9-8312-C7672007AC3E"
```

```
   },

   "exists" : 1,

   "name" : "Scorecard",

   "type" : "scorecard"

}
```

GET /api/v2/{ctxID}/doc{docNum}/scorecardrating

Get the results for the overall rating only from the “Scorecard” parser.

Curl example

```
curl -X GET
"https://textinspector.com/api/v2/{ctdId}/doc{docNum}/scorecardrati
ng" -H "accept: application/json" -H "Cookie:
textinspector.session={sessionid};" -L
```

Example response - scorecard

```
{

   "type" : "scorecard",

   "name" : "Scorecard",

   "exists" : 1,

   "response" : {

      "ctxId" : "400A816C-9D6C-11E9-8312-C7672007AC3E",

      "documentId" : "doc1",

      "overallRating" : {

         "numberOfMetricsUsed" : "16",

         "friendlyName" : "Overall Rating",

         "cefrLevel" : "C1",
          "percentage" : "62.92"

       }

    }

}
```

Example PHP code for using the API

```
<?php

/**
 * Example PHP code for using the Text Inspector API from Weblingua
 */

$api_base_url = 'https://textinspector.com/api/v2;

$api_username = 'YOUR_USERNAME';
$api_password = 'YOUR_PASSWORD';

// Get session ID
$ch = curl_init($api_base_url.'/createsession');
curl_setopt($ch, CURLOPT_HTTPHEADER, array('accept:
application/json'));
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);
curl_setopt($ch, CURLOPT_USERPWD, $api_username . ":" .
$api_password);
curl_setopt($ch, CURLOPT_TIMEOUT, 30);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
$return = curl_exec($ch);
curl_close($ch);
$response = json_decode($return);
$sessionid = $response->sessionid;

if (!$sessionid) {
  print "Failed to get session ID\n";
  print_r($return);
  Exit;
}
print "Got Session ID:$sessionid\n";

// Submit a new text for analysis
$fields = array(
  'text' => 'Try the tool out with this paragraph of text here, by
pressing Analyse below. Or you can replace this whole text with some
text of your choosing. Simply paste any text into this box, or else
type it in yourself! Text Inspector gives different scores for Writing
texts (student writing) and Reading and Listening texts (e.g. texts
designed for classroom reading or listening).  The default is Writing,
so if your text is for Reading or Listening, please go to Advanced
Options to change the mode and get accurate calculations. Look at our
Subscriptions section for special offers and discounts.',
  'delimiter' => '#',
  'split' => 1,
  'textmode' => 'Listening'
);
$ch = curl_init($api_base_url.'/newanalysis');
curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type:
application/json','accept: application/json', 'Cookie:
textinpsector.session='.$sessionid));
curl_setopt($ch, CURLOPT_TIMEOUT, 30);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
curl_setopt($ch,CURLOPT_POST, count($fields));
curl_setopt($ch,CURLOPT_POSTFIELDS, json_encode($fields));
$return = curl_exec($ch);
curl_close($ch);
$response = json_decode($return);
$ctxid = $response->response->ctxId;

if (!$ctxid) {
  print "Failed to get context ID\n";
  print_r($return);
  exit;
}
print "Got Context ID: $ctx\n";

// Get statistics parser results
$ch = curl_init($api_base_url.'/'.$ctxid.'/doc1/statistics');
curl_setopt($ch, CURLOPT_HTTPHEADER, array('accept:
application/json'));
curl_setopt($ch, CURLOPT_HTTPHEADER, array("Cookie:
textinpsector.session=".$sessionid));
```

```
curl_setopt($ch, CURLOPT_TIMEOUT, 30);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
$return = curl_exec($ch);
curl_close($ch);
$parser_results = json_decode($return);

print_r($parser_results);
```

Document version history

Version	Date	Status	Reviewed by
1	April 2018	Draft	Nik
2	May 2018	V1.0 published at https://docs.google.com/document/d/1iLm7ND4yKd12iYmR-itnS3thCh1yLgRSBljGzLUJV6M/edit?usp=sharing	Nik
3	June 8 2018	V1.1 - remove 250 word limit for API users. Limit now 15,000	Nik
4	June 14 2018	V1.2 - added missing statistics entries, added taggedlexical-bnc, taggedlexical-coca,tagger, metadiscourse	Nik
5	July 16 2018	V1.3 - added missing academicwordlist, lexical and tiprofile . Added sessionId as part of authentication header	Mary
6	Sept 25 2018	V1.4 - updated details about accessing API and Session ID usage	NickR
7	April 30,2019	V1.5 - fixed typo in cookie session name	Mary
8	May 21 2019	V.1.5.1 - updated for	NickR

		changes to the API for v1 of the live API - first non-beta version	
9	June 28 2019	v2.0.1 - updated for changes for v2 of the API	NickR
10	June 05 2020	v2.0.2 - rewrote introduction, createsession and newanalysis sections to try and make clearer	NickR
11	Apr 18, 2023	v2.1.0 - added tagged data export	NickR