Contents

Academic Knowledge API

8/1/2019 • 2 minutes to read • Edit Online

Welcome to the Academic Knowledge API. With this service, you will be able to interpret user queries for academic intent and retrieve rich information from the Microsoft Academic Graph (MAG). The MAG knowledge base is a web-scale heterogeneous entity graph comprised of entities that model scholarly activities: field of study, author, institution, paper, venue, and event.

The MAG data is mined from the Bing web index as well as an in-house knowledge base from Bing. As a result of on-going Bing indexing, this API will contain fresh information from the Web following discovery and indexing by Bing. Based on this dataset, the Academic Knowledge APIs enables a knowledge-driven, interactive dialog that seamlessly combines reactive search with proactive suggestion experiences, rich research paper graph search results, and histogram distributions of the attribute values for a set of papers and related entities.

For more information on the Microsoft Academic Graph, see https://aka.ms/academicgraph.

The Academic Knowledge API has moved from Cognitive Services Preview to Cognitive Services Labs. The new homepage for the project is: https://labs.cognitive.microsoft.com/en-us/project-academic-knowledge. Your existing API key will continue working until May 24th, 2018. After this date, please generate a new API key. Please note that paid preview will no longer be available once your existing key expires. Please contact our team if the free tier of the API is not sufficient for your purposes.

Features

The Academic Knowledge API consists of four related REST endpoints:

1. **interpret** – Interprets a natural language user query string. Returns annotated interpretations to enable rich search-box auto-completion experiences that anticipate what the user is typing.
2. **evaluate** – Evaluates a query expression and returns Academic Knowledge entity results.
3. **calchistogram** – Calculates a histogram of the distribution of attribute values for the academic entities returned by a query expression, such as the distribution of citations by year for a given author.

Used together, these API methods allow you to create a rich semantic search experience. Given a user query string, the **interpret** method provides you with an annotated version of the query and a structured query expression, while optionally completing the user's query based on the semantics of the underlying academic data. For example, if a user types the string *latent s*, the **interpret** method can provide a set of ranked interpretations, suggesting that the user might be searching for the field of study *latent semantic analysis*, the paper *latent structure analysis*, or other entity expressions starting with *latent s*. This information can be used to quickly guide the user to the desired search results.

The **evaluate** method can be used to retrieve a set of matching paper entities from the academic knowledge base, and the **calchistogram** method can be used to calculate the distribution of attribute values for a set of paper entities which can be used to further filter the search results.

Getting Started

Please see the subtopics at the left for detailed documentation. Note that to improve the readability of the examples, the REST API calls contain characters (such as spaces) that have not been URL-encoded. Your code will need to apply the appropriate URL-encodings.

Entity Attributes

8/1/2019 • 2 minutes to read • Edit Online

The academic graph is composed of 7 types of entity. All entities will have an Entity ID and an Entity type.

Common Entity Attributes

NAME	DESCRIPTION	TYPE	OPERATIONS
Id	Entity ID	Int64	Equals
Ty	Entity type	enum	Equals

Entity type enum

NAME	VALUE
Paper	0
Author	1
Journal	2
Conference Series	3
Conference Instance	4
Affiliation	5
Field Of Study	6

Paper Entity

10/30/2019 • 2 minutes to read • Edit Online

*Below attributes are specific to paper entity. (Ty = '0')

NAME	DESCRIPTION	TYPE	OPERATIONS
AA.AfId	Author affiliation ID	Int64	Equals
AA.AfN	Author affiliation name	String	Equals, StartsWith
AA.AuId	Author ID	Int64	Equals
AA.AuN	Normalized author name	String	Equals, StartsWith
AA.DAuN	Original author name	String	None
AA.DAfN	Original affiliation name	String	None
AA.S	Numeric position in author list	Int32	Equals
CC	Citation count	Int32	None
C.CId	Conference series ID	Int64	Equals
C.CN	Conference series name	String	Equals, StartsWith
D	Date published in YYYY-MM-DD format	Date	Equals, IsBetween
E	Extended metadata (see table below)	String	N/A
ECC	Estimated citation count	Int32	None
F.DFN	Original field of study name	String	None
F.FId	Field of study ID	Int64	Equals
F.FN	Normalized field of study name	String	Equals, StartsWith
Id	Paper ID	Int64	Equals
J.JId	Journal ID	Int64	Equals
J.JN	Journal name	String	Equals, StartsWith

NAME	DESCRIPTION	TYPE	OPERATIONS
Pt	Publication type (0:Unknown, 1:Journal article, 2:Patent, 3:Conference paper, 4:Book chapter, 5:Book, 6:Book reference entry, 7:Dataset, 8:Repository	String	Equals
RId	List of referenced paper IDs	Int64[]	Equals
Ti	Normalized title	String	Equals, StartsWith
W	Unique words in title	String[]	Equals
Y	Year published	Int32	Equals, IsBetween

Extended Metadata Attributes

NAME	DESCRIPTION
BT	BibTex document type ('a':Journal article, 'b':Book, 'c':Book chapter, 'p':Conference paper)
BV	BibTex venue name
CC	Citation Contexts – List of referenced paper ID's and the corresponding context in the paper (e.g. [{123:["brown foxes are known for jumping as referenced in paper 123", "the lazy dogs are a historical misnomer as shown in paper 123"]})
DN	Original paper title
DOI	Digital Object Identifier
FP	First page of paper in publication
I	Publication issue
IA	Inverted Abstract
IA.IndexLength	Number of items in the index (abstract's word count)
IA.InvertedIndex	List of abstract words and their corresponding position in the original abstract (e.g. [{"the":[0, 15, 30]}, {"brown":[1]}, {"fox":[2]}])
LP	Last page of paper in publication
PB	Publisher
S	Sources - list of web sources of the paper, sorted by static rank

NAME	DESCRIPTION
S.Ty	Source Type (1:HTML, 2:Text, 3:PDF, 4:DOC, 5:PPT, 6:XLS, 7:PS)
S.U	Source URL
V	Publication volume
VFN	Full name of the Journal or Conference venue
VSN	Short name of the Journal or Conference venue

Author Entity

8/1/2019 • 2 minutes to read • Edit Online

*Following attributes are specific to author entity. (Ty = '1')

NAME	DESCRIPTION	TYPE	OPERATIONS
Id	Entity ID	Int64	Equals
AuN	Author normalized name	String	Equals
DAuN	Author display name	String	none
CC	Author total citation count	Int32	none
ECC	Author total estimated citation count	Int32	none
E	Extended metadata (see "Extended Meta Attributes" table)	String	none

Extended Metadata Attributes

NAME	DESCRIPTION
LKA.Afn	affiliation's display name associated with the author
LKA.AfId	affiliation's entity ID associated with the author

Affiliation Entity

8/1/2019 • 2 minutes to read • Edit Online

*Following attributes are specific to affiliation entity. (Ty = '5')

NAME	DESCRIPTION	TYPE	OPERATIONS
Id	Entity ID	Int64	Equals
AfN	Affiliation normalized name	String	Equals
DAfN	Affiliation display name	String	none
CC	Affiliation total citation count	Int32	none
ECC	Affiliation total estimated citation count	Int32	none

Extended Metadata Attributes

NAME	DESCRIPTION
PC	Affiliation's paper count

Field Of Study Entity

8/1/2019 • 2 minutes to read • Edit Online

*Following attributes are specific to field of study entity. (Ty = '6')

NAME	DESCRIPTION	TYPE	OPERATIONS
Id	Entity ID	Int64	Equals
FN	Field of study normalized name	String	Equals
DFN	Field of study display name	String	none
CC	Field of study total citation count	Int32	none
ECC	Field of total estimated citation count	Int32	none
FL	Level in fields of study hierarchy	Int32	Equals, IsBetween
FP.FN	Parent field of study name	String	Equals
FP.FId	Parent field of study ID	Int64	Equals
FC.FN	Child field of study name	String	Equals
FC.FId	Child field of study ID	Int64	Equals

Conference Series Entity

8/1/2019 • 2 minutes to read • Edit Online

*Following attributes are specific to conference series entity. (Ty = '3')

NAME	DESCRIPTION	TYPE	OPERATIONS
Id	Entity ID	Int64	Equals
CN	Conference series normalized name	String	Equals
DCN	Conference series display name	String	none
CC	Conference series total citation count	Int32	none
ECC	Conference series total estimated citation count	Int32	none
F.FId	Field of study entity ID associated with the conference series	Int64	Equals
F.FN	Field of study name associated with the conference series	Equals, StartsWith	

Conference Instance Entity

8/1/2019 • 2 minutes to read • Edit Online

*Following attributes are specific to conference instance entity. (Ty = '4')

NAME	DESCRIPTION	TYPE	OPERATIONS
Id	Entity ID	Int64	Equals
CIN	Conference instance normalized name ({ConferenceSeriesNormalizedName} {ConferenceInstanceYear})	String	Equals
DCN	Conference instance display name ({ConferenceSeriesName} : {ConferenceInstanceYear})	String	none
CIL	Location of the conference instance	String	Equals, StartsWith
CISD	Start date of the conference instance	Date	Equals, IsBetween
CIED	End date of the conference instance	Date	Equals, IsBetween
CIARD	Abstract registration due date of the conference instance	Date	Equals, IsBetween
CISDD	Submission due date of the conference instance	Date	Equals, IsBetween
CIFVD	Final version due date of the conference instance	Date	Equals, IsBetween
CINDD	Notification date of the conference instance	Date	Equals, IsBetween
CD.T	Title of a conference instance event	Date	Equals, IsBetween
CD.D	Date of a conference instance event	Date	Equals, IsBetween
PCS.CN	Conference series name of the instance	String	Equals

NAME	DESCRIPTION	TYPE	OPERATIONS
PCS.CId	Conference series ID of the instance	Int64	Equals
CC	Conference instance total citation count	Int32	none
ECC	Conference instance total estimated citation count	Int32	none

Extended Metadata Attributes

NAME	DESCRIPTION
FN	Conference instance full name

Journal Entity

8/1/2019 • 2 minutes to read • Edit Online

*Following attributes are specific to journal entity. (Ty = '2')

NAME	DESCRIPTION	TYPE	OPERATIONS
Id	Entity ID	Int64	Equals
DJN	Journal normalized name	String	none
JN	Journal display name	String	Equals
CC	Journal total citation count	Int32	none
ECC	Journal total estimated citation count	Int32	none

CalcHistogram Method

8/1/2019 • 2 minutes to read • Edit Online

The **calchistogram** REST API is used to calculate the distribution of attribute values for a set of paper entities.

REST endpoint:

```
https:// westus.api.cognitive.microsoft.com/academic/v1.0/calchistogram?
```

Request Parameters

NAME	VALUE	REQUIRED?	DESCRIPTION
expr	Text string	Yes	A query expression that specifies the entities over which to calculate histograms.
model	Text string	No	Select the name of the model that you wish to query. Currently, the value defaults to *latest*.
attributes	Text string	No default:	A comma-delimited list that specifies the attribute values that are included in the response. Attribute names are case-sensitive.
count	Number	No Default: 10	Number of results to return.
offset	Number	No Default: 0	Index of the first result to return.
timeout	Number	No Default: 1000	Timeout in milliseconds. Only interpretations found before the timeout has elapsed are returned.

Response (JSON)

NAME	DESCRIPTION
expr	The expr parameter from the request.
num_entities	Total number of matching entities.

NAME	DESCRIPTION
histograms	An array of histograms, one for each attribute specified in the request.
histograms[x].attribute	Name of the attribute over which the histogram was computed.
histograms[x].distinct_values	Number of distinct values among matching entities for this attribute.
histograms[x].total_count	Total number of value instances among matching entities for this attribute.
histograms[x].histogram	Histogram data for this attribute.
histograms[x].histogram[y].value	A value for the attribute.
histograms[x].histogram[y].logprob	Total natural log probability of matching entities with this attribute value.
histograms[x].histogram[y].count	Number of matching entities with this attribute value.
aborted	True if the request timed out.

Example:

```
https:// westus.api.cognitive.microsoft.com/academic/v1.0/calchistogram?expr=And(Composite(AA.AuN=='jaime
teevan'),Y>2012)&attributes=Y,F.FN&count=4
```

In this example, in order to generate a histogram of the count of publications by year for a particular author since 2010, we can first generate the query expression using the **interpret** API with query string: *papers by jaime teevan after 2012*.

```
https:// westus.api.cognitive.microsoft.com/academic/v1.0/interpret?query=papers by jaime teevan after 2012
```

The expression in the first interpretation that is returned from the interpret API is *And(Composite(AA.AuN=='jaime teevan'),Y>2012)*.
This expression value is then passed in to the **calchistogram** API. The *attributes=Y,F.FN* parameter indicates that the distributions of paper counts should be by Year and Field of Study, e.g.:

```
https:// westus.api.cognitive.microsoft.com/academic/v1.0/calchistogram?expr=And(Composite(AA.AuN=='jaime
teevan'),Y>2012)&attributes=Y,F.FN&count=4
```

The response to this request first indicates that there are 37 papers that match the query expression. For the *Year* attribute, there are 3 distinct values, one for each year after 2012 (i.e. 2013, 2014, and 2015) as specified in the query. The total paper count over the 3 distinct values is 37. For each *Year*, the histogram shows the value, total natural log probability, and count of matching entities.

The histogram for *Field of Study* shows that there are 34 distinct fields of study. As a paper may be associated with

multiple fields of study, the total count (53) can be larger than the number of matching entities. Although there are 34 distinct values, the response only includes the top 4 because of the *count=4* parameter.


```
{
  "expr": "And(Composite(AA.AuN=='jaime teevan'),Y>2012)",
  "num_entities": 37,
  "histograms": [
    {
      "attribute": "Y",
      "distinct_values": 3,
      "total_count": 37,
      "histogram": [
        {
          "value": 2014,
          "logprob": -15.753,
          "count": 15
        },
        {
          "value": 2013,
          "logprob": -15.805,
          "count": 12
        },
        {
          "value": 2015,
          "logprob": -16.035,
          "count": 10
        }
      ]
    },
    {
      "attribute": "F.FN",
      "distinct_values": 34,
      "total_count": 53,
      "histogram": [
        {
          "value": "crowdsourcing",
          "logprob": -15.258,
          "count": 9
        },
        {
          "value": "information retrieval",
          "logprob": -16.002,
          "count": 4
        },
        {
          "value": "personalization",
          "logprob": -16.226,
          "count": 3
        },
        {
          "value": "mobile search",
          "logprob": -17.228,
          "count": 2
        }
      ]
    }
  ]
}
```

Evaluate Method

8/1/2019 • 2 minutes to read • Edit Online

The **evaluate** REST API is used to return a set of academic entities based on a query expression.

REST endpoint:

```
https://westus.api.cognitive.microsoft.com/academic/v1.0/evaluate?
```

Request Parameters

NAME	VALUE	REQUIRED?	DESCRIPTION
expr	Text string	Yes	A query expression that specifies which entities should be returned.
model	Text string	No	Name of the model that you wish to query. Currently, the value defaults to *latest*.
attributes	Text string	No default: Id	A comma-delimited list that specifies the attribute values that are included in the response. Attribute names are case-sensitive.
count	Number	No Default: 10	Number of results to return.
offset	Number	No Default: 0	Index of the first result to return.
orderby	Text string	No Default: by decreasing prob	Name of an attribute that is used for sorting the entities. Optionally, ascending/descending can be specified. The format is: *name:asc* or *name:desc*.

Response (JSON)

NAME	DESCRIPTION
expr	The *expr* parameter from the request.

NAME	DESCRIPTION
entities	An array of 0 or more entities that matched the query expression. Each entity contains a natural log probability value and the values of other requested attributes.
aborted	True if the request timed out.

Example:

```
https://westus.api.cognitive.microsoft.com/academic/v1.0/evaluate?expr=
Composite(AA.AuN=='jaime teevan')&count=2&attributes=Ti,Y,CC,AA.AuN,AA.AuId
```

Typically, an expression will be obtained from a response to the **interpret** method. But you can also compose query expressions yourself (see Query Expression Syntax).

Using the *count* and *offset* parameters, a large number of results may be obtained without sending a single request that results in a huge (and potentially slow) response. In this example, the request used the expression for the first interpretation from the **interpret** API response as the *expr* value. The *count=2* parameter specifies that 2 entity results are being requested. And the *attributes=Ti,Y,CC,AA.AuN,AA.AuId* parameter indicates that the title, year, citation count, author name, and author ID are requested for each result. See Entity Attributes for a list of attributes.

```
{
  "expr": "Composite(AA.AuN=='jaime teevan')",
  "entities":
  [
    {
      "logprob": -15.08,
      "Ti": "personalizing search via automated analysis of interests and activities",
      "Y": 2005,
      "CC": 372,
      "AA": [
        {
          "AuN": "jaime teevan",
          "AuId": 1968481722
        },
        {
          "AuN": "susan t dumais",
          "AuId": 676500258
        },
        {
          "AuN": "eric horvitz",
          "AuId": 1470530979
        }
      ]
    },
    {
      "logprob": -15.389,
      "Ti": "the perfect search engine is not enough a study of orienteering behavior in directed search",
      "Y": 2004,
      "CC": 237,
      "AA": [
        {
          "AuN": "jaime teevan",
          "AuId": 1982462162
        },
        {
          "AuN": "christine alvarado",
          "AuId": 2163512453
        },
        {
          "AuN": "mark s ackerman",
          "AuId": 2055132526
        },
        {
          "AuN": "david r karger",
          "AuId": 2012534293
        }
      ]
    }
  ]
}
```

Interpret Method

8/1/2019 • 2 minutes to read • Edit Online

The **interpret** REST API takes an end user query string (i.e., a query entered by a user of your application) and returns formatted interpretations of user intent based on the Academic Graph data and the Academic Grammar.

To provide an interactive experience, you can call this method repeatedly after each character entered by the user. In that case, you should set the **complete** parameter to 1 to enable auto-complete suggestions. If your application does not need auto-completion, you should set the **complete** parameter to 0.

REST endpoint:

```
https://westus.api.cognitive.microsoft.com/academic/v1.0/interpret?
```

Request Parameters

NAME	VALUE	REQUIRED?	DESCRIPTION
query	Text string	Yes	Query entered by user. If complete is set to 1, query will be interpreted as a prefix for generating query auto-completion suggestions.
model	Text string	No	Name of the model that you wish to query. Currently, the value defaults to *latest*.
complete	0 or 1	No default:0	1 means that auto-completion suggestions are generated based on the grammar and graph data.
count	Number	No default:10	Maximum number of interpretations to return.
offset	Number	No default:0	Index of the first interpretation to return. For example, *count=2&offset=0* returns interpretations 0 and 1. *count=2&offset=2* returns interpretations 2 and 3.
timeout	Number	No default:1000	Timeout in milliseconds. Only interpretations found before the timeout has elapsed are returned.

Response (JSON)

NAME	DESCRIPTION
query	The *query* parameter from the request.
interpretations	An array of 0 or more different ways of matching user input against the grammar.
interpretations[x].logprob	The relative natural log probability of the interpretation. Larger values are more likely.
interpretations[x].parse	An XML string that shows how each part of the query was interpreted.
interpretations[x].rules	An array of 1 or more rules defined in the grammar that were invoked during interpretation. For the Academic Knowledge API, there will always be 1 rule.
interpretations[x].rules[y].name	Name of the rule.
interpretations[x].rules[y].output	Output of the rule.
interpretations[x].rules[y].output.type	The data type of the output of the rule. For the Academic Knowledge API, this will always be "query".
interpretations[x].rules[y].output.value	The output of the rule. For the Academic Knowledge API, this is a query expression string that can be passed to the evaluate and calchistogram methods.
aborted	True if the request timed out.

Example:

```
https://westus.api.cognitive.microsoft.com/academic/v1.0/interpret?query=papers by jaime&complete=1&count=2
```

The response below contains the top two (because of the parameter *count=2*) most likely interpretations that complete the partial user input *papers by jaime*: *papers by jaime teevan* and *papers by jaime green*. The service generated query completions instead of considering only exact matches for the author *jaime* because the request specified *complete=1*. Note that the canonical value *j l green* matched via the synonym *jamie green*, as indicated in the parse.

```
{
  "query": "papers by jaime",
  "interpretations": [
    {
      "logprob": -12.728,
      "parse": "<rule name=\"#GetPapers\">papers by <attr name=\"academic#AA.AuN\">jaime teevan</attr></rule>",
      "rules": [
        {
          "name": "#GetPapers",
          "output": {
            "type": "query",
            "value": "Composite(AA.AuN=='jaime teevan')"
          }
        }
      ]
    },
    {
      "logprob": -12.774,
      "parse": "<rule name=\"#GetPapers\">papers by <attr name=\"academic#AA.AuN\" canonical=\"j l
green\">jaime green</attr></rule>",
      "rules": [
        {
          "name": "#GetPapers",
          "output": {
            "type": "query",
            "value": "Composite(AA.AuN=='j l green')"
          }
        }
      ]
    }
  ]
}
```


To retrieve entity results for an interpretation, use *output.value* from the **interpret** API, and pass that into the **evaluate** API via the *expr* parameter. In this example, the query for the first interpretation is:

```
evaluate?expr=Composite(AA.AuN=='jaime teevan')
```

Similarity Method

8/1/2019 • 2 minutes to read • Edit Online

The **similarity** REST API is used to calculate the academic similarity between two strings.

REST endpoint:

```
https://westus.api.cognitive.microsoft.com/academic/v1.0/similarity?
```

Request Parameters

PARAMETER	DATA TYPE	REQUIRED	DESCRIPTION
s1	String	Yes	String* to be compared
s2	String	Yes	String* to be compared

*Strings to compare have a maximum length of 1MB.

Response

NAME	DESCRIPTION
SimilarityScore	A floating point value representing the cosine similarity of s1 and s2, with values closer to 1.0 meaning more similar and values closer to -1.0 meaning less

Success/Error Conditions

HTTP STATUS	REASON	RESPONSE
200	Success	Floating point number
400	Bad request or request invalid	Error message
500	Internal server error	Error message
Timed out	Request timed out.	Error message

Example: Calculate similarity of two partial abstracts

Request:

```
https://westus.api.cognitive.microsoft.com/academic/v1.0/similarity?s1=Using complementary priors, we derive a fast greedy algorithm that can learn deep directed belief networks one layer at a time, provided the top two layers form an undirected associative memory
&s2=Deepneural nets with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks
```

In this example, we generate the similarity score between two partial abstracts using the **similarity** API.

Response:

```
0.520
```

Remarks:

The similarity score is determined by assessing the academic concepts through word embedding. In this example, 0.52 means that the two partial abstracts are somewhat similar.

Query Expression Syntax

8/1/2019 • 3 minutes to read • Edit Online

We have seen that the response to an **interpret** request includes a query expression. The grammar that interpreted the user's query created a query expression for each interpretation. A query expression can then be used to issue an **evaluate** request to retrieve entity search results.

You can also construct your own query expressions and use them in an **evaluate** request. This can be useful if you are building your own user interface which creates a query expression in response to the user's actions. To do this, you need to know the syntax for query expressions.

Each entity attribute that can be included in a query expression has a specific data type and a set of possible query operators. The set of entity attributes and supported operators for each attribute is specified in Entity Attributes. A single-value query requires the attribute to support the *Equals* operation. A prefix query requires the attribute to support the *StartsWith* operation. Numeric range queries requires the attribute to support the *IsBetween* operation.

Some of the entity data are stored as composite attributes, as indicated by a dot '.' in the attribute name. For example, Author/Affiliation information is represented as a composite attribute. It contains 4 components: AuN, AuId, AfN, AfId. These components are separate pieces of data that form a single entity attribute value.

String Attribute: Single value (includes matches against synonyms)
Ti='indexing by latent semantic analysis'
Composite(AA.AuN='sue dumais')

String Attribute: Exact single value (matches only canonical values)
Ti=='indexing by latent semantic analysis'
Composite(AA.AuN=='susan t dumais')

String Attribute: Prefix value
Ti='indexing by latent seman'...
Composite(AA.AuN='sue du'...)

Numeric Attribute: Single value
Y=2010

Numeric Attribute: Range value
Y>2005
Y>=2005
Y<2010
Y<=2010
Y=[2010, 2012) (includes only left boundary value: 2010, 2011)
Y=[2010, 2012] (includes both boundary values: 2010, 2011, 2012)

Numeric Attribute: Prefix value
Y='19'... (any numeric value that starts with 19)

Date Attribute: Single value
D='2010-02-04'

Date Attribute: Range value
D>'2010-02-03'
D=['2010-02-03','2010-02-05']

And/Or Queries:

And(Y=1985, Ti='disordered electronic systems')
Or(Ti='disordered electronic systems', Ti='fault tolerance principles and practice')
And(Or(Y=1985,Y=2008), Ti='disordered electronic systems')

Composite Queries:
To query components of a composite attribute, you need to enclose the part of the query expression that refers to the composite attribute in the Composite() function.

For example, to query for papers by author name, use the following query:

```
Composite(AA.AuN='mike smith')
```

To query for papers by a particular author while the author was at a particular institution, use the following query:

```
Composite(And(AA.AuN='mike smith',AA.AfN='harvard university'))
```

The Composite() function ties the two parts of the composite attribute together. This means that we only get papers where one of the authors is “Mike Smith” while he was at Harvard.

To query for papers by a particular author in affiliations with (other) authors from a particular institution, use the following query:

```
And(Composite(AA.AuN='mike smith'),Composite(AA.AfN='harvard university'))
```

In this version, because Composite() is applied to the author and affiliation individually before And(), we get all papers where one of the authors is “Mike Smith” and one of the authors' affiliations is “Harvard”. This sounds similar to the previous query example, but it's not the same thing.

In general, consider the following example: We have a composite attribute C that has two components A and B. An entity may have multiple values for C. These are our entities:

```
E1: C={A=1, B=1}  C={A=1,B=2}  C={A=2,B=3}
E2: C={A=1, B=3}  C={A=3,B=2}
```

The query

```
Composite(And(C.A=1, C.B=2))
```

matches only entities that have a value for C where the component C.A is 1 and the component C.B is 2. Only E1 matches this query.

The query

```
And(Composite(C.A=1), Composite(C.B=2))
```

matches entities that have a value for C where C.A is 1 and also have a value for C where C.B is 2. Both E1 and E2

match this query.

Please note:

- You cannot reference a part of a composite attribute outside of a Composite() function.
- You cannot reference parts of two different composite attributes inside the same Composite() function.
- You cannot reference an attribute that is not part of a composite attribute inside a Composite() function.