Article
Benjamin De Boe · Jun 7, 2016 7m read

# iKnow Feature Overview - Sentiment Analysis

Sentiment Analysis is a thriving research area in the broader context of big data, with many small as well as large vendors offering solutions extracting sentiment scores from free text. As sentiment is highly dependent on the subject a piece of text is about (financial news vs tweets about the latest computer game), most of these solutions are targeted at specific markets and/or focus on a given type of source data, such as social media content. Also, given that the vocabulary used to express sentiment evolves over time, especially on social media with younger or mixed audiences, sentiment analysis software often needs to be updated or re-trained to stay up-to-date.

Starting with version 2015.2, InterSystems ' iKnow technology supports identifying sentiment as an attribute in the same way as the negation attribute, which was introduced in 2013.1. The attribute approach allows users to not just highlight the mere usage of positive or negative sentiment terms, but also which parts of a sentence (path) the sentiment applies to, based on linguistic rules. This " path expansion " adds accuracy and therefore significant value over the simplified approach used by many other software packages which is to look at word positions or simply presence within the sentence to identify what the sentiment applies to.

In order to avoid hardcoding a sentiment terminology dictionary (implying the challenges described earlier) and to ensure maximum flexibility, InterSystems ' iKnow provides developers with the option to feed their own domain- and/or application-specific dictionary of sentiment terminology to the iKnow engine. The engine will then perform path expansion of the sentiment attribute when a term is encountered in a sentence, telling you to which entities the sentiment applies. This mechanism is provided as an extension to the **User Dictionary**, which is the primary mechanism to feed additional information to the iKnow engine, complementing the default iKnow language models. The next few sections will provided more detailed information about using these mechanisms in practice.

## Setting up a User Dictionary with Sentiment Markers

The **Using iKnow** guide describes how to set up a User Dictionary (%iKnow.UserDictionary) and include it in a %iKnow.Configuration object.

Starting in 2015.2, the %iKnow.UserDictionary interface introduces new, specific methods to add terms carrying positive or negative sentiment:

- AddPositiveSentimentTerm()
- AddNegativeSentimentTerm()

The same mechanism can be used to add custom negation markers using a different method:

- AddNegationTerm()

Using the above methods will ensure the User Dictionary contains the appropriate information for the engine to identify the terms in input text and perform path expansion appropriately.

**Note:** these attributes are supported for the English, German, Russian and Ukrainian language models in 2015.2, and for the remaining languages (except Japanese) in 2015.3.

```
USER> set UD = ##class(%iKnow.UserDictionary).%New("MyUD")
```

```
USER> write UD.%Save()
1

USER> write UD.AddPositiveSentimentTerm("happy")
1

USER> write UD.AddNegativeSentimentTerm("awful")
1

USER> set Config = ##class(%iKnow.Configuration).%New("Cfg")

USER> set Config.UserDictionary = "MyUD"

USER> write Config.%Save()
1

USER> set txt = "I'm happy to wear these shoes, but the socks are awful."

USER> write $system.iKnow.IndexString("","x",txt,"Cfg",,.src)
1

USER> write ##class(%iKnow.Queries.PathAPI).GetAttributes( .att, 0, 1, src)
1

USER> zwrite att
att(1)=$lb(5,"sentpositive",2,3,"")
att(2)=$lb(6,"sentnegative",6,3,"")

USER> do ##class(%iKnow.Source.Loader).DeleteVirtualSource(0, src)
```

## Current limitations

### Normalized format

Currently terms have to be supplied **in lowercase**, as the iKnow engine will match them to the normalized tokens in your sentences. Note that this is different from the case-sensitive rewrite rules, which can also be specified through the User Dictionary.

### Overlap with language model

As of 2015.2, there are two limitations regarding the use of these user-defined labels, due to the way how they technically complement the built-in language model ' s internal representation.

1. If the user-defined entry itself does not appear in the language model, but is part of a string that *is* part of the language model, it will not be picked up. For example, if your UD contains the word " anxious " , its label will not be picked up because the term " anxious about " is part of the iKnow language model and the simple term " anxious " is not.
2. If the user-defined entry is a multi-word term, it will only be picked up if that term is already part of the language model itself. For example, a user-defined label for the term " anxious for " would not be picked up, but would for " anxious about " .

These two limitations have been removed in 2016.3, as the User Dictionary now takes precedence over the regular language model.

## Using the Sentiment Attributes in iKnow

Sentiment Analysis information can be leveraged in the same way as other attributes (such as negation) through the following methods:

- %iKnow.Queries.EntityAPI : GetOccurrenceAttributes() and IsAttributed()
- %iKnow.Queries.PathAPI : GetAttributes()
- %iKnow.Queries.SourceAPI : GetAttributes()
- %iKnow.Queries.SentenceAPI : GetAttributes() and GetHighlighted()

Especially the last one is useful as it takes care of all aspects of highlighting entities and words affected by the attributes. This method is also used in the "Indexing Results" sample UI to render sentiment data.

The following sample code could be added to the previous examples, right before calling the DeleteVirtualSource() method:

```
USER> set H("ATTRIBUTE", 5) = "<span style='color:Green'>"

USER> set H("ATTRIBUTE", 6) = "<span style='color:Red'>"

USER> set H("ATTRIBUTEWORDS", 5) = "<u>"

USER> set H("ATTRIBUTEWORDS", 6) = "<u>"

USER> w ##class(%iKnow.Queries.SentenceAPI).GetHighlighted(0, 1, .H, src)

I&#39;m <span style='color:Green'><u>happy</u></span> <span style='color:Green'>to we
ar</span> these <span style='color:Green'>shoes</span>, but the <span style='color:Re
d'>socks</span> <span style='color:Red'>are</span> <span style='color:Red'><u>awful</
u></span>.
```

Note that 2015.2 also introduces a shorthand method $system.iKnow.Highlight() allowing you feed text and highlighting schema directly, with no need to create and drop a (virtual) source.

**Note**: in the above example, the numbers 5 and 6 represent the attribute IDs for positive and negative sentiment respectively, as available through the macros $$$IKATTSENPOSITIVE and $$$IKATTSENNEGATIVE in %IKPublic.inc.

## Sentiment Analysis in iFind

Sentiment Analysis can also be used to enrich the text search experience in iFind. In order to use a User Dictionary for an iFind index, set the USERDICTIONARY index parameter to the name of a %iKnow.UserDictionary object with sentiment information created as described above. This will ensure the User Dictionary is used by the iKnow engine when indexing records in the table the index is registered for. Note that you will need a Semantic or Analytic index with the IFINDATTRIBUTES parameter set to 1 to save and leverage attribute information.

```
Class Sentiment.MyClass Extends %Persistent {

Property MyText As %String(MAXLEN=20000);

Index MyIndex On (MyText) As %iFind.Index.Semantic(
         IFINDATTRIBUTES=1, USERDICTIONARY="MyDict");
```

```
}
```

This will then generate classes projecting relevant word (MyClassMyIndexWordPos ) and attribute (MyClassMyIndexAttributePos ) information in the SentimentMyClass package. They can be used to resolve advanced queries like the following, which retrieves sentiment information for each word position:

```
SELECT
     wp.RecId,
     wp.Position,
     WordId->Value,

     (SELECT TOP 1
         CASE ABS(AttributeId)
              WHEN 5 THEN 'positive'
              WHEN 6 THEN 'negative'
         END
     FROM
       Sentiment_MyClass.MyClass_MyIndex_AttributePos ap
     WHERE
         ap.RecId = wp.RecId
         AND ap.Position = wp.Position
     ORDER BY
         AttributeId DESC
     ) sentiment

FROM
  Sentiment_MyClass.MyClass_MyIndex_WordPos wp
ORDER BY
     wp.RecId, wp.Position
```

Note that in the above query, you ' ll see double rows for some word positions, where the original word contained punctuation. iFind retains both the original, punctuated form and the stripped form of each word to enable refining your search when needed.

Also note that we ' re using ABS(AttributeId) in the above query. iFind will store the explicit word-level sentiment (or other attribute) markers with a positive attribute ID, whereas the positions getting their attribute through path expansion will be annotated with the negative version of the corresponding attribute ID. You can change the CASE statement in the above query to reflect this:

```
...
   CASE AttributeId
         WHEN 5 THEN 'positive (explicit)'
         WHEN 6 THEN 'negative (explicit)'
         WHEN -5 THEN 'positive (implied)'
         WHEN -6 THEN 'negative (implied)'
   END
...
```

This is similar to the difference between the " ATTRIBUTEWORDS " and " ATTRIBUTE " highlighting options we saw earlier. The actual attribute ID numbers are listed in %IKPublic.inc, as we saw earlier too.

#iKnow