SAS® Text Analytics for Business Applications

Concept Rules for Information Extraction Models

Teresa Jade
Biljana Belamaric Wilsey
Michael Wallis

The correct bibliographic citation for this manual is as follows: Jade, Teresa, Biljana Belamaric Wilsey, and Michael Wallis. 2019. *SAS® Text Analytics for Business Applications: Concept Rules for Information Extraction Models*. Cary, NC: SAS Institute Inc.

SAS® Text Analytics for Business Applications: Concept Rules for Information Extraction Models

978-1-64295-194-3 (Hard cover)
978-1-63526-664-1 (Hardcopy)
978-1-63526-663-4 (Web PDF)
978-1-63526-661-0 (epub)
978-1-63526-662-7 (mobi)

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

March 2019

Contents

About This Book

What Does This Book Cover?

The independent research firm Forrester recognized SAS as a leader in text analytics. The field of text analytics is rapidly growing, but effective use of models for information extraction (IE) is elusive. This book focuses on tips, best practices, and pitfalls of writing IE rules and creating IE models, with the goal of meeting the needs of a broad audience of users of text analytics products such as SAS Visual Text Analytics, SAS Contextual Extraction, and SAS Enterprise Content Categorization.

The primary goal of the book is to answer application questions, such as the following:

- What criteria were used for developing the predefined concept rules and what type of results are expected to be found in the data? (See chapters 2 and 3.)
- When it is appropriate to use each type of rule, and how do the different rule types interact? (See chapters 4–9 and 11.)
- What pitfalls should be avoided and what best practices should be kept in mind when one is developing IE projects by applying predefined and custom concepts? (See chapters 4–10.)

In answering those questions, this book provides details of concept rule creation using the proprietary SAS syntax called LITI, which stands for *language interpretation for textual information*. Because real-world examples are essential for increased relevance to users, the book presents best practices from seasoned practitioners through realistic use cases and real data as much as possible. The topic selection and discussions were informed by real questions from SAS customers. Important and helpful hints from SAS Text Analytics experts are presented through tool tip boxes.

Generally, more complex topics come after less complex ones, so if you find yourself lost, go back and master some earlier sections; then try the more advanced ones again. As you advance your own skills with IE, you will rely on different sections of the book to assist you. Progressing through the book will be an indicator of successful learning on your part, as will increased success with your IE models.

This book focuses on the underlying functionality of the SAS IE toolkit and not on the visual aspects that may change with software product or version. Although screen captures from current SAS Text Analytics tools are included, please refer to existing documentation to interpret interface components as necessary.

The information in this book builds on SAS Text Analytics documentation by taking a more detail-oriented approach. For example, components of the SAS IE toolkit, such as the basics of rule syntax, lists of predefined concepts, and part-of-speech tags, are described in the documentation for SAS Visual Text Analytics, SAS Contextual Analysis, and SAS Enterprise

Content Categorization. This book picks up where documentation leaves off, so you may find it useful to keep your documentation handy as you proceed, in case you need to refresh your memory of how a product feature works or where to find it in the graphical user interface (GUI). The documentation is accessible from the SAS Support website: https://support.sas.com.

Is This Book for You?

This book is for advanced beginner- or intermediate-level SAS Text Analytics products users who are technically savvy and want to leverage unstructured text analysis by using SAS IE tools. It is written primarily for those text analytics practitioners with some experience in text analytics, so it is assumed that you will have a basic familiarity with SAS Text Analytics products, interfaces, IE rule types, and format, as well as familiarity with finding documentation on how the rules work.

The book is conceptualized as a train-the-trainer technical reference. The topics in the book were chosen to demonstrate how SAS IE can bring value and insights derived from text data. Combined with practical experience and application, this knowledge empowers you to deepen your knowledge of IE methods and the SAS IE toolkit, as well as to become an expert and champion for harnessing text analytics to answer business questions. Additionally, by creating a community of SAS Text Analytics products champions, the book helps educate non-text-oriented practitioners about the rigor of text analytics processes and tools, demonstrating that text analytics is an important part of data science.

What Should You Know about the Examples?

This book includes examples for you to follow to gain hands-on experience with SAS. There are simple and complex examples of each of the LITI rule types and business scenarios combining various rule types into more realistic models. These examples and scenarios span various subject domains, including banking and capital markets, communications, energy and utilities, government, health care, insurance, life sciences, manufacturing, retail, and services.

“Pause and think” boxes have been incorporated throughout the book to provide opportunities for guided practice, review, and application of presented materials with real-life examples. These hands-on sections are a signal for you to pause your reading, try the rules and models yourself in your SAS Text Analytics product, and test your understanding of presented concepts. You can check your output with the output in the book and read the explanations that follow these sections to deepen your understanding.

Software Used to Develop the Book's Content

In the book, you will find examples from each of the SAS Text Analytics products that support creating custom LITI rules: SAS Visual Text Analytics, SAS Contextual Analysis, and SAS Enterprise Content Categorization. All examples can be duplicated in any of the three products, with exceptions noted. The concepts that are created with LITI rules can also be compiled and applied to data sources using Text Analytics DS2 code and Cloud Analytic Services (CAS) actions.

Example Code and Data

You are encouraged to follow along with the examples, run them in your SAS Text Analytics product, and examine the results. Because the text analytics products enable you to create and apply IE models both programmatically and through a GUI, you can also experiment with the examples in the book both ways. The book content includes input documents and rules that you can copy and paste into a taxonomy in your product's GUI to create and apply a model. Alternatively, the programmatic version of the rules and models is provided in the supplemental online materials accompanying the book. This code assumes that you are using SAS Viya and that you have access to the text analytics CAS action sets.

You can access the example code and data for this book by linking to its author pages at https://support.sas.com/authors.

We Want to Hear from You

Do you have questions about a SAS Press book that you are reading? Contact us at saspress@sas.com.

SAS Press books are written by SAS Users for SAS Users. Please visit sas.com/books to sign up to request information on how to become a SAS Press author.

We welcome your participation in the development of new books and your feedback on SAS Press books that you are using. Please visit sas.com/books to sign up to review a book

Learn about new books and exclusive discounts. Sign up for our new books mailing list today at https://support.sas.com/en/books/subscribe-books.html.

Learn more about these authors by visiting their author pages, where you can download free book excerpts, access example code and data, read the latest reviews, get updates, and more:
https://support.sas.com/jade
https://support.sas.com/belamaric-wilsey
https://support.sas.com/wallis

Chapter 1: Fundamentals of Information Extraction with SAS

1.1. Introduction to Information Extraction

At a recent analytics conference, a data analyst approached the SAS Text Analytics booth and asked whether her organization could derive value from unstructured text data. She came to the conference with a solid understanding that there is value in analyzing structured data but was not sure whether the same was true for unstructured text, such as free-form comments, surveys, notes, social media content, news stories, emails, financial reports, adjustor notes, doctor's notes and similar sources.

The answer to this question of deriving value from unstructured text is an unequivocal "yes, it is possible!" This book will show you how *information extraction* (IE) is one way to turn that unstructured text into valuable structured data. You will be able to use the resulting data to improve predictive models, improve categorization models, enrich an index to use in search, or examine patterns in a business reporting tool like SAS Visual Analytics.

This chapter introduces what IE is and when to use it in SAS Text Analytics products. Chapters 2, 3, and 4 give you the knowledge and understanding you need to leverage pre-built sets of rules that are provided in the software "out of the box." You learn how to build your own rules and models in chapters 12–14. Along the way, you will encounter many types of information patterns found in text data across a variety of domains, including health care, manufacturing, banking, insurance, retail, hospitality, marketing, and government. These examples illustrate the value that text data contains and how it can be accessed and leveraged in any SAS Text Analytics product to solve business problems.

1.1.1. History

The practice of extraction of structured information from text grew out of the theories and efforts of several scientists in the early 1970s:

- Roger C. Schank's conceptual dependency theoretical model of parsing natural language texts into formal semantic representations
- R. P. Abelson's conceptual dependency analysis of the structure of belief systems
- Donald A. Norman's representation of knowledge, memory, and retrieval

At this time, the concern was with two-way relationships between actors and actions in sentences (Moens 2006). For example, Company X acquired Company Y; the two companies are in an acquisition relationship. In the mid-1970s, through Marvin Minsky's theoretical work, the focus became frame-based knowledge representation: a *frame* is a data structure with a number of slots that represent knowledge about a set of properties of a stereotyped situation (Moens 2006). For example, for an acquisition, you can add slots like date, valuation, acquiring company, acquired company, and so forth. At the same time, logician Richard Montague and linguist Noam Chomsky were writing about transformational and universal grammars as structures for analyzing formal/artificial and natural languages syntactically and semantically.

By the 1980s, the Defense Advanced Research Projects Agency and the Naval Ocean Systems Center were fueling rapid advances through sponsoring biennial Message Understanding Conferences (MUCs), which included competitions on tasks for automated text analysis and IE (Grishman and Sundheim 1996). The texts ranged from military messages in the first few MUCs to newswire articles and non-English texts in the later ones (Piskorski and Yangarber 2013). The tasks continued the tradition of frames, as they still involved identifying classes of events and filling out slots in templates with event information, although the slots became more complex, nested, and hierarchical as the field advanced (Grishman and Sundheim 1996). In 1995, *named entity recognition* (NER) was introduced as a MUC IE task for the first time (Jiang 2012). NER models extract the names of people, places, and things. In chapter 2, you can learn more about NER and how the SAS Text Analytics products extract information by using techniques for NER.

In 1999, the successful MUC initiative grew into the Automated Content Extraction program, which continued encouraging the development of content extraction technologies for automatic processing of increasingly complex natural language data (Piskorski and Yangarber 2013). In the 21st century, other initiatives, such as the Conference on Computational Natural Language Learning, Text Analysis Conference, and Knowledge Base Population, also adopted the MUC approach to competitions that target complex tasks such as discovering information about entities and incorporating it into knowledge bases (Piskorski and Yangarber 2013; Jurafsky and Martin 2016).

Through the decades, the tasks in the field have grown in complexity in three major areas:

- *Source data*. The data being analyzed has become more complex: from only well-formed, grammatical English text-based documents of a single type (i.e., military reports, news) and document-level tasks, to extraction from various types of sources, well-formed or not (i.e., social media data), across large numbers of documents, in languages other than English, and in non-text-based media (such as images and audio files).
- *Scope of the core tasks*. The core IE tasks have changed from shallow, task-dependent IE to deeper analysis through entity resolution including co-reference (linking multiple references to the same referent), word sense disambiguation (distinguishing multiple meanings of the same word), and predicate-argument structure (linking subjects, objects, and verbs in the same clause).
- *Systems and methods*. The domain-dependent systems with limited applications have expanded to include domain-independent, portable systems based on a combination of rule-based and statistical machine/deep learning methods (supervised, semi-supervised, and unsupervised).

This gradual growth in the complexity of analysis necessitated additional resources for processing and normalization of texts because treating text-based data as a sequence of strings did not leverage enough of the embedded linguistic information. Such resources included tokenization, sentence segmentation, and morphological analysis (Moens 2006).

The SAS Text Analytics products leverage natural language processing (NLP) methods and pair them with a proprietary rule-writing syntax called *language interpretation for textual information* (LITI) to help you extract the information you need from your unstructured text data. This combination, with rule-building tools and support such as automatic rule generation, applies the best of what statistical machine learning has to offer with a rule-based approach for better transparency in extraction.

1.1.2. Evaluation

Another tradition that originally came out of the MUC program is the approach and metrics used for measuring the success of an IE model. In IE, the model targets a span of labeled text. For example, consider the following sentence:

Jane Brown registered for classes on Tuesday.

Possible spans of labeled text in this example include the following:

- "Jane Brown," which has two tokens and could be labeled Person
- "Tuesday," which is one token that could be labeled Date

In general, the most important things to know about a span of text identified by a model are as follows:

1. Is the span of text that was found an accurate representative of the targeted information?
2. Were all the targeted spans of text found in the corpus?

The first of these items is called *precision* and represents how often the results of the model or analysis are right, based on a human-annotated answer key. Precision is the ratio of the number of correctly labeled spans to the total that were labeled in the model. It is a measure of exactness or quality and is typically calculated by this formula:

$$\text{Precision} = \frac{\#\text{Correct spans in model}}{\#\text{Correct spans in model} + \#\text{Incorrect spans in model}}$$

If the model found only "Jane Brown" as Person, then the number of correct spans would be 1 and the number of incorrect spans would be 0, so precision would be 100%. Precision is easy to measure because you need to examine only the output of the model to calculate it.

The second of these items is called *recall* and represents how many of the spans of text representing a targeted entity that exists in the data are actually found by the model. Recall is the ratio of the number of correctly labeled responses to the total that should have been labeled by the model as represented in the answer key. It is a measure of completeness and is typically calculated by this formula:

$$\text{Recall} = \frac{\#\text{Correct spans in model}}{\#\text{Correct spans in key}}$$

In the example at the opening of this section, the number of correct spans in the model was 1 (i.e., only "Jane Brown" was found), but the number of correct spans in the key was 2. Therefore, recall is 50%. The model would have incorrectly missed "Tuesday" as a Date. Recall is more difficult to measure because you need to know all the correct spans in your answer key, so every span in the key must be examined, and all spans to be matched must be annotated.

There are some basic tradeoffs between recall and precision because the most accurate system in terms of precision would extract one thing and, so long as it was right, precision would be 100%, as illustrated by our current basic example. The most accurate system in terms of recall would do the opposite and extract everything, making the recall an automatic 100%. Therefore, when you are evaluating an IE system, reporting a balanced measure of the two can be useful. The harmonic mean of these two measures is called *F-measure* (F_1) and is frequently used for this purpose. It is typically calculated by the following formula, and it can also be modified to favor either recall or precision:

$$F_1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

In terms of these metrics, a good IE model will have a measure of the accuracy that shows a balance between precision and recall for each of the pieces of information it seeks to extract. It is also possible to use these metrics and a smaller annotated sample to estimate the accuracy of a model that is then applied to a larger data set. In other words, if you are planning to build a model to use on a large data set, you do not need to manually annotate the full data set to know the quality of your results.

For more information about setting up measurement for IE projects, see chapter 14.

1.1.3. Information Extraction versus Data Extraction versus Information Retrieval

The phrase “information extraction" is sometimes confused with either data extraction/collection or information retrieval (Piskorski & Yangarber 2013), but they are all different processes. *Data extraction and collection* describes the gathering of data in order to create a corpus or data set. Methods of data extraction include crawling websites, querying or collecting subsets of data from known data sources, and collecting data as it arrives in a single place. The corpus is usually created on the basis of the origin or purpose of the data, but sometimes it might be culled from a larger data collection by the use of keywords or a where-clause. The use of keywords makes the activity seem much like information retrieval, but the goal is to collect *all* items containing the keywords. Recall, not precision, is the focus when you are assessing the success of the collection effort. An example of collection without use of keywords is the collection of all call center notes in a single repository. This process may occur alongside other common processes to collect structured data, as well.

Information retrieval, in contrast, assumes that you already have a data collection or corpus to pull information from. The goal in this case is to align information with a specific information need or question. The result is a set of possible answers in the form of a ranked list, which is not normally intended to be a comprehensive collection of answers or related information. An information retrieval process is successful if at least one document toward the top of the list satisfies the information need. Precision, not recall, is the focus. Keywords and natural language queries are used to interrogate the original data collection.

After a process of data extraction or collection has been completed and a corpus or data set exists, *information extraction* pulls out specific hidden information, facts, or relationships from the data. You can use these facts and relationships as new information, structured data, directly in reports or indirectly in predictive models to answer specific business questions. Both precision and recall are usually in focus and balanced toward the particular use case. The use cases throughout this book illustrate various types of information you can extract as part of this process.

The differences between these terms can be summarized as follows:

- *Data extraction* or *collection* results in a data set or corpus of documents
- *Information retrieval* results in a ranked set of answers to an information question linked to documents
- *Information extraction* results in new structured data variables that can stand alone or be appended to existing data sets

1.1.4. Situations in Which to Use IE for Business Problems

You should use IE when you want to take information from an unstructured or semi-structured text data type to create new structured text data. IE works at the sub-document level, in contrast with techniques, such as categorization, that work at the document or record level. Therefore, the results of IE can further feed into other analyses, like predictive modeling or topic identification, as features for those processes. IE can also be used to create a new database of information. One example is the recording of key information about terrorist attacks that are reported in the news. Such a database can then be used and analyzed through queries and reports about the data.

One good use case for IE is for creating a faceted search system. *Faceted search* allows users to narrow down search results by classifying results by using multiple dimensions, called *facets*, simultaneously. For example, faceted search may be used when analysts try to determine why and where immigrants may perish. The analysts might want to correlate geographical information with information that describes the causes of the deaths in order to determine what actions to take.

Another good example of using IE in predictive models is analysts at a bank who want to determine why customers close their accounts. They have an active churn model that works fairly well at identifying potential churn, but less well at determining what causes the churn. An IE model could be built to identify different bank policies and offerings and then track mentions of each during any customer interaction. If a particular policy could be linked to certain churn behavior, then the policy could be modified to reduce the number of lost customers.

Reporting information found as a result of IE can provide deeper insight into trends and uncover details that were buried in the unstructured data. An example of this is an analysis of call center notes at an appliance manufacturing company. The results of IE show a pattern of customer-initiated calls about repairs and breakdowns of a type of refrigerator, and the results highlight particular problems with the doors. This information shows up as a pattern of increasing calls. Because the content of the calls is being analyzed, the company can return to its design team, which can find and remedy the root problem.

The uses of IE can be complex, as demonstrated by these examples, or relatively simple. A simple use case for IE is sentence extraction. Breaking longer documents down into sentences is one way to address the complexity of the longer documents. It is a good preprocessing step for some types of text analytics. For an example of an IE rule for transforming your documents into sentences, see section 8.3.2.

1.2. The SAS IE Toolkit

The SAS IE toolkit includes the following components:

- NLP foundation for IE
- LITI rule syntax
- Predefined concepts (out-of-the-box NER)
- Taxonomy of components for each model
- Three types of matching algorithms
- Graphical user interface (GUI) for building and testing models to sample data sets and a programmatic interface for building and applying models to large data sets

These parts of the IE toolkit operate together. They also integrate well with the larger SAS product suite including other SAS Text Analytics capabilities—categorization, for example—and SAS Viya products, such as SAS Visual Data Management and Machine Learning, SAS Visual Analytics, and SAS Model Manager.

1.2.1. NLP Foundation for IE

The first component in the SAS IE toolkit, NLP, involves computational and linguistic approaches to enabling computers to “understand” human language. Computers process character-by-character or byte-by-byte and have no conceptualization of “word,” “sentence,” “verb,” or the like. NLP provides methods that help the computer model the structure and information encoded in human language.

Some of the foundational methods of NLP include tokenization, sentence breaking, part-of-speech (POS) tagging, lemmatization or stemming, misspelling detection, and grammatical parsing. These foundational NLP processes often feed information into higher-level processing types, such as machine translation, speech-to-text processing, IE, and categorization. The SAS Text Analytics products carry out many of these foundational NLP analyses behind the scenes and make the results available as part of the IE toolkit. Toolkit users do not directly see or participate in the NLP foundation but benefit in various ways, which are described in the next few sections.

Tokenization

One of the basic operations in NLP and a critical task for effective IE is *tokenization*. *Tokenization* refers to the process of analyzing alphanumeric characters, spaces, punctuation and special characters to determine where to draw boundaries between them. The pieces of text that are separated by those boundaries are called *tokens*.

Different text processing systems may approach tokenization differently. Some tasks may require that tokens be as short as possible, whereas others may produce better results if tokens are longer. Furthermore, natural languages have different conventions for certain characters such as white space and punctuation. For example, Chinese does not have white spaces between words, Korean sometimes has white spaces between words, and English usually has white spaces between words. These conventions play an important role in tokenization. Even if focusing only on English text, different tokenization approaches may produce different results.

Consider the following example sentence:

> Starting Dec. 21st, Mrs. Bates-Goodman won’t lead the co-op any more.

Pause and think: Can you identify some words that could potentially be tokenized two or more different ways in the sentence above?

You may have identified some of the following possible differences in tokenization in the sentence:

- "Dec." could be 1 or 2 tokens: /Dec./ or /Dec/./
- "21st" could be 1 or 2 tokens: /21st/ or /21/st/
- "Dec. 21st" could possibly be 1 token if dates are important: /Dec. 21st/
- "Mrs." could be 1 or 2 tokens: /Mrs./ or /Mrs/./
- "Bates-Goodman" could be 1 or 3 tokens: /Bates-Goodman/ or /Bates/-/Goodman/
- "Mrs. Bates-Goodman" could possibly be 1 token if person names are important /Mrs. Bates-Goodman/
- "won't" could be 1, 2, or 3 tokens: /won't/, /won/'t/, or /won/'/t/ or even be turned into /will/not/
- "co-op" could be 1 or 3 tokens: /co-op/ or /co/-/op/

Furthermore, some systems may tokenize proper names like "Bates-Goodman" differently from words that may be found in a dictionary and contain a hyphen, such as "co-op." In other words, when you are tokenizing text, there are many decisions that must be made in order to present the most meaningful set of tokens possible to aid downstream analysis. For more information about how complex the tokenization of periods can be, see Belamaric Wilsey and Jade (2015).

The default SAS Text Analytics tokenization approach embodies one of these advanced systems that tries to get these decisions right. The tokens are optimized to represent semantic meaning. Therefore, if a character is a part of a series of characters that means something, then the goal is to make all of the series into a single token rather than keeping them as separate pieces of meaningless text. This approach is effective for enabling better POS tagging, which will be described in more detail in the next section.

Since at least 2016, the English language analysis tools in SAS have followed this approach of tokenization based on meaningful units. In order to limit the combinations, the SAS method of NLP follows two rules about putting together pieces with internal white space. First, there are no tokens with white space created during tokenization, so you can use special tags (described in the subsection "Part-of-speech Tagging" below), such as ":url" or ":time," and they will match tokens without white space only. Second, the only tokens containing internal white space come from a process known as *multiword identification*, a process whereby meaningful terms that have multiple pieces, but a single meaning and POS, are combined as a single compound token. For example, SAS NLP will analyze "high school" as a single token based on an entry in the multiword dictionary.

In English and many other languages, there is a process of word formation called *compounding*, which combines two separate words together to create a new expression with a different meaning than that comprised by the two words used together. It is common for this process to start with the two words used as a pair of words with a normal space between them, for example, "bubble wrap." Later, as users of the multiword become accustomed to the new meaning, the pieces may be hyphenated or even written as a single word, for example, "play-date," "suitcase," "nickname," or even "before." Analyzing these terms as a single token when they are still space-separated, but have a single meaning, improves POS tagging and topic identification.

Tokens are important for the SAS IE toolkit, because a token defines the unit over which an IE model will operate. The model can recognize and operate over a single token or a series of multiple tokens, but it will not easily recognize partial tokens, such as only “ing” in word endings. This tokenization limitation actually saves a lot of work, because the models can be based on semantically meaningful units rather than being cleaned up piece by piece before finally targeting the meaningful pieces.

If you are accustomed to modeling using only a regular expression approach to processing text data, you may find that this token-based approach to models seems to limit your options at first. However, if you shift your focus and strategy to target those larger tokens, you will likely find that you end up with a smarter and more easily maintained model in the long run. If that is not the case for your data, then you can still turn to the regular expression syntax in SAS code in procedures, such as the PRXCHANGE procedure, to identify partial-token matches.

Other Boundaries

Another type of division of the text that is provided as a part of the NLP foundation for IE is *sentence tokenization* or *sentence segmentation*. In this process, the data is broken up into sentence-level pieces, taking into account cues including punctuation, newline characters and other white space, and abbreviations in each language. All SAS Text Analytics products detect sentence boundaries and feed this information forward into the IE and categorization processes.

Some SAS Text Analytics products will also detect simple paragraph boundaries and pass that information into both IE and categorization. Additionally, detection of clause boundaries for IE is a planned feature on the development roadmap in order to enable even more refined IE models.

Part-of-Speech Tagging

Once the tokens, the units of analysis, have been determined in the NLP foundation for IE, it is useful to understand how they fit into the sentence from a grammatical viewpoint. For this task, a set of grammatical labels is applied that determine each token’s POS. These labels, such as “noun,” “verb,” “adjective,” “adverb,” and so on are called *POS tags*, and they are fully documented in your product documentation. Assigning these labels to tokens is called *tagging*. There are also a few special tags that can be applied to tokens, which include the following: “:sep,” “:digit,” “:url,” “:time,” and “:date.” These tags, explained in Table 1.1, are created for specific types of tokens that are not labeled with grammatical tags.

Table 1.1. Special Tags and Description

Special Tag	Description
:sep	Applied to any single punctuation character that stands alone as a token
:digit	Applied to any single or multiple numeric token; number
:url	Applied to URLs, email addresses, and computer path; digital location
:time	Applied to any string of characters without spaces that depicts a time
:date	Applied to any string of characters without spaces that depicts a date

Knowing a token's tag adds tools to your IE toolkit that enable you to refer to and capture tokens that appear in the same grammatical patterns in a sentence. For illustration, consider the following phrases: "a counteractive measure," "an understandable result," and "the predictable outcome."

Pause and think: How could you use tagging to extract these phrases in your IE model?

Because the phrases all follow the same POS pattern of a determiner followed by an adjective and noun, an IE rule that references those POS tags in a sequence will extract all three phrases, as well as any additional ones that follow the same pattern in the text. Leveraging POS tags makes IE rules more efficient and versatile.

Parenting

In addition to tagging, two other NLP processes that happen behind-the-scenes in SAS Text Analytics products help to group related tokens together into sets: identification of inflectional variation of terms (*lemmatization*) and misspelling detection. Inflectional variants are those words that come from a *lemma*, the base form of a word, and remain in the same basic POS family. For example, English verb paradigms can contain multiple forms:

- The base form, also called the infinitive, as in "be"
- The first person present tense "am"
- The second person present tense "are"
- The third person present tense "is"
- The first person past tense "was"

In the SAS IE toolkit, you can access these "sets" of words directly through a single form, called the *parent term*. See section 1.5.3 for more details about parenting.

Misspelling detection is the second process that adds word forms to the set of child terms under a parent. When users choose to turn on this feature, misspellings are automatically detected and added to the "sets" of words grouped under a parent term.

Hybrid System

The NLP processing that takes place to produce tokens, lemmas, POS tags, misspellings, and the like uses a combination of dictionaries, human-authored rules, and machine learning approaches. In other words, like most real-world NLP systems, it is a hybrid system. SAS linguists are continually working to improve and modernize the approaches used in the SAS NLP foundation. Therefore, an upgrade or move to a newer SAS Text Analytics product will likely result in differences in how this processing occurs or the results you may see on specific data. It is advised that you recheck any models that you migrate from system to system so that you can adjust your models, if needed, to align with the newer outputs.

It is important to note that, even though the quality of the results of SAS NLP is increasing over time, the specific results you may observe on a particular data set may vary in quality. Particularly, if you are using very noisy or ungrammatical data, the results may not always look like what you would expect them to. For example, POS tagging assumes *sentential* data, which is data containing sentences with punctuation. Therefore, examining POS tagging output on non-sentential data will often not provide expected results, because context is a critical part of the POS tagging analysis.

The SAS linguists strive to ensure that the NLP foundation works well on data from the common domain, as well as across all the domains of SAS customers, including health care, energy, banking, manufacturing, and transportation. Also, the analysis must work well on sentential text from a variety of document types, such as emails, technical reports, abstracts, tweets, blogs, call center notes, SEC filings, and contracts.

Because of the variety of language and linguistic expression, correctly processing all of these types of data from all the domains is an unusual challenge. The typical NLP research paper usually reports on a specific domain and frequently also addresses a single document type. SAS linguists have a higher standard and measure results against standard data collections used in research for each language, as well as against data that SAS customers have provided for testing purposes. If you have data that you want the SAS systems to process well, you are encouraged to provide SAS with a sample of the data for testing purposes. All of the supported languages would benefit from additional customer data for testing. You can contact the authors or SAS Technical Support to begin this process.

1.2.2. LITI Rule Syntax

The SAS IE toolkit leverages the hybrid systems in the NLP foundation, but centers on a rule-based approach for the IE component. This type of IE approach consists of collections of rules for extraction and policies to determine the interactions between those rule collections. The rules in the SAS IE toolkit leverage a proprietary programming language called *LITI*. Policies include procedures for arranging taxonomies and resolving match conflicts.

LITI is a proprietary programming language used to create models that can extract particular pieces of text that are relevant for various types of informational purposes. The LITI language organizes sets of rules into groups called *concepts*. Each group of rules can be referenced as a set in other rules through the name of the concept. This approach enables models to work like a well-designed building with foundational pieces that no one sees directly, such as electrical wiring and plumbing, as well as functional pieces that visitors to the building would readily identify, such as doors, elevators, and windows.

Each rule written in the LITI syntax is a command to look for particular characteristics and patterns in the textual data and return targeted strings of text whenever the specified conditions are met in the text data. You can use LITI to look for regular expressions, simple or complex strings, strings in particular contexts, items from a class (like a POS class such as “verb”), and items in particular relationships based on proximity and context. LITI syntax enables modeling of rules through different rule types, combinations of rule types, and operators, including Boolean and proximity operators.

The LITI syntax is flexible and scalable. One aspect of LITI that contributes to these attributes is the variety of rule types that are available. Many other IE engines take advantage of regular expression rules. In addition to this capability, LITI supports eight other rule types, which give you the ability to extract strings with or without specifying context and with or without extracting the context around those strings. In addition, the rules for fact matches allow you to specify and extract relationships between two or more matches in a given context. Finally, the LITI syntax enables you to take advantage of Boolean and proximity operators, such as AND, OR, SENT and others, to restrict extracted matches. The benefit of this set of rule types is that the user can target exactly the type of match needed efficiently, without using more processing than is required for that type of extraction.

The different types of rules and operators, as well as LITI rule syntax, are discussed in detail in chapters 5–11. For users who do not want to write their own rules, the next section discusses automatically generated rules, as well as pre-built rules.

1.2.3. Predefined Concepts

LITI rules can be written by a human or automatically generated. Automatically generated rules have been included in the SAS IE toolkit since 2018 and are available through the SAS Visual Text Analytics GUI starting in product releases in 2019. In addition, commonly used concepts for each supported language are predefined and provided out-of-the-box as part of the toolkit. All fully supported languages in SAS Text Analytics products support at least seven SAS-proprietary predefined concepts, including the following: Person Names, Location Names, Organization Names, Dates, Times, Currency Amounts, and Percentage Amounts. Additional predefined concepts, such as Measures or third-party open-source predefined concepts, may also be available in certain products. Some examples are provided in Table 1.2.

Table 1.2. Example Matches for Predefined Concepts

Predefined Concept	Example Match (in Bold)
Person Name	**Mayor Roland Ries** told XYZ television the gunman got inside a security zone to stage the terrorist attack in the French city of Strasbourg on Tuesday at 10am.
Organization Name	Mayor Roland Ries told **XYZ television** the gunman got inside a security zone to stage the terrorist attack in the French city of Strasbourg on Tuesday at 10am.
Location Name	Mayor Roland Ries told XYZ television the gunman got inside a security zone to stage the terrorist attack in the French city of **Strasbourg** on Tuesday at 10am.
Date	Mayor Roland Ries told XYZ television the gunman got inside a security zone to stage the terrorist attack in the French city of Strasbourg on **Tuesday** at 10am.

Predefined Concept	Example Match (in Bold)
Time	Mayor Roland Ries told XYZ television the gunman got inside a security zone to stage the terrorist attack in the French city of Strasbourg on Tuesday at **10am**.
Currency Amount	Stocks fell 3.5% and investors lost around **$1.1 trillion**.
Percentage Amount	Stocks fell **3.5%** and investors lost around $1.1 trillion.

These predefined concepts enable the immediate identification of information that many models need to find. They are built with general data sets and therefore have some limitations in terms of accuracy in specialized data sets. However, you can expand and constrain them through custom, user-written rules that refine their behavior in accordance with the specific characteristics of the data to which they will be applied. In other words, predefined concepts are a springboard for immediate text analysis and a solid foundation on which to build even more effective concepts for specific analytic needs.

SAS proprietary predefined concepts are introduced in chapter 2. The definition and results for each concept are described in detail in chapters 3 and 4.

1.2.4. Taxonomy of Concepts

Another piece of the SAS IE toolkit is the *taxonomy* of concepts. In the design stages of a model, each of the concepts, which are groups of LITI rules, is organized into a hierarchy, or taxonomy. This taxonomy serves as a blueprint for the model being built. It lays out the pieces and enables a visual depiction of their relation to one another. This feature is flexible and allows the model to be set up in a logical way for testing and maintenance of the particular rule sets used in the model.

For example, you will probably want to set up any reasonably complex model with two types of concepts. One type of concept will be used to produce your explicit results—to generate your extracted information data. This type of concept is a *functional* piece of the model, because it produces results that can be observed. Other pieces of the model could be constructed solely to support the output of these functional components, like the foundation of a building supports all the pieces that one can see and use directly. These supporting concepts can be called *foundational* or *helper* concepts. They have rules and produce results, but those results only feed into other concepts rather than producing resulting data from the model.

Taxonomy design and setting up projects is discussed in more detail in chapters 12–14. Best practices for identifying the concepts to use are in section 11.3, and how to set up more complex models is covered in chapters 12 and 13.

1.2.5. Algorithms for Matching

Another feature available as a part of the SAS IE toolkit is choice of matching algorithm: “all matches,” “longest match,” or “best match.” When a LITI rule is evaluated, the software can assess whether the same span of text has been matched by another concept or rule. Two of the three available matching algorithms use this information to filter out or select among possible matches: the “best match” algorithm and the “longest match” algorithm. In some SAS Text Analytics products, the default algorithm is the third one, “all matches,” which returns all the matches after applying the model.

The selection of a matching algorithm is one way you can control the output of your model—for example, to reduce duplicate matches. This choice may determine which matches are returned under the circumstances of overlap or duplication. More details about the three available matching algorithms and how to use them is found in section 13.4.1. The chapters that introduce rules and all the examples up to chapter 12 use the “all matches” algorithm.

1.2.6. Interfaces for Building and Applying Models

Building and applying an IE model can be achieved in two different ways in SAS Text Analytics products: through a GUI or programmatically.

Each SAS Text Analytics product includes a GUI for rule building and testing, which is an iterative process. In the GUI, a data set can be examined, rules can be written, and results of applying the model can be evaluated. Therefore, the GUI environments have been optimized to support this process of model creation and iterative evaluation and improvement.

In each product, the GUI shows the model taxonomy, enables changes to the taxonomy, associates the rule set with the concept in the taxonomy, and shows matches from testing associated with each concept. Tests can be conducted on snippets of text or across all documents loaded into the GUI, and results can then be examined and analyzed. In some SAS Text Analytics products, the metrics of recall and precision are also presented within the GUI context.

If you have created a model in the GUI, it can automatically generate code to apply the model to a new data set. This code is called *score code*, and the process of applying it is often referred to as *scoring*. It enables efficient application of the model to a large amount of data. The score code can be based on DS2 or Cloud Analytics Services (CAS) actions, depending on the product that generates it. Starting with products released in 2019, models can also be exported and applied in an analytic store (ASTORE) format. Score code can be modified, if needed, and applied to a new set of documents or placed in production. One frequent modification to the model parameters in score code is changing the matching algorithm to control the output, as described in section 13.4.

As an alternative to the GUI approach, you can also build and apply IE models programmatically. The process for building a model programmatically is explained in section 5.3.4. You can apply the programmatically built model with the SAS IE procedures or actions, using a SAS programming interface or SAS Studio, as described in section 5.3.5.

This book will focus primarily on the approach to rule building used in the GUIs, but not focus on a particular GUI environment, because the goal is to present information that can be used across all of the GUIs available in SAS Text Analytics products and versions. Where screenshots are shown, the product used will be mentioned, and any relevant differences across products will also be highlighted whenever possible. However, the basics of building and applying rules programmatically are also covered, and the code samples in the supplemental materials online follow this approach (for how to locate these materials, see

About This Book). Therefore, you can quickly reproduce the results presented in the book by simply running the provided code.

1.3. Reasons for Using SAS IE

SAS IE has many benefits over other types of IE approaches. Because it is part of a set of text analytics tools, as well as being integrated into the larger SAS product family, you have the flexibility to combine it with other tools and products, such as statistical or predictive algorithms and machine learning. SAS IE models can be integrated directly into SAS categorization models, used to make more useful topics, or applied independently to create structured data that can be used to populate databases, inform predictive models, or feed into reports such as those available in SAS Visual Analytics.

In addition to being a part of a larger ecosystem of analytics, another benefit of SAS IE is that you can create models right away by creating rules and concepts without requiring large amounts of pre-annotated data for training a model. Furthermore, SAS Text Analytics products provide a set of out-of-the-box concepts that you can leverage immediately, and the GUI enables customization of those rules by adding or removing matches.

If you have annotated data or can create it, you have the option of applying an automated process to generate LITI rules in SAS Text Analytics (starting in 2018 in the programmatic interface and in 2019 in the GUI). You can later edit and cull these rules. Check user documentation for more information about how the algorithms for rule generation work. When building rules, just as with any other model, the amount of knowledge and care given to the creation of the model directly impacts the quality and the usefulness of the model. For this reason, it is not recommended that you rely solely on automated rule generation, but instead combine this method with human understanding, tuning, and testing of the rules to meet your specific goals.

Because rule writing can take time and effort, scientific papers on IE often cite these requirements as drawbacks (Jiang 2012, p.17). However, with the right tools and expertise for building rules, this approach has been proven to be more accurate, robust, and interpretable than typical machine-learning-only or statistical models in a variety of use cases (Chiticariu et al. 2013; Liu et al. 2013; Small and Medsker 2013; Woodie 2018). Other benefits of rule-based systems for IE include readability, maintainability, expressivity, and transparency, as well as the direct transfer of domain and linguistic knowledge into rules (Waltl et al. 2018).

Typical machine-learning models also have drawbacks, such as lack of transparency, limited interpretability, and inability to leverage in-house human expertise. Furthermore, these models require a large amount of manually annotated data for training, development, and testing purposes. In many cases, this data does not come for free, or if it does, then it does not reflect the same type of textual content that may be targeted with the model. In those situtations of mismatch between training and target data, the accuracy of the model is lowered even further.

In reality, both defining rules for a complex model and curating annotated (pretagged) data are big jobs. The former simply gives you more control over your model and helps you build up the latter as you go. Therefore, if you want a model that you will reuse repeatedly and perfect over time to be as accurate as possible, then using predefined and custom rules that you define for your specific business needs may be the best path for you.

Another benefit of the SAS IE toolkit is that the NLP work, such as tokenization, stemming, lemmatization, and POS tagging, is done behind the scenes, without users having to build any of those features into their models directly. Users access some of the resulting information inside the syntax of LITI. For example, inflectional variants can be referenced without specifying each variant in rules. A newer feature in SAS Text Analytics products released since 2017 is called *Textual Elements*. Under each parent term, groups of terms, such as misspellings and synonyms, together with inflectional variants, are recognized as a set, which makes easy use of the full set of variants in rules.

As already mentioned, for ease of building, testing, and updating models, as well as taxonomy construction and maintenance, each product that supports the SAS IE toolkit, such as SAS Text Analytics, SAS Contextual Analysis, and SAS Enterprise Content Categorization, includes a GUI. Features, such as syntax highlighting and testing at both the model level and the specific concept level with both test text and documents, facilitate easier experimentation with and maintenance of rules, concepts, and models. In addition to the benefits already mentioned, the GUI also supports componential design. You can abstract pieces of the model for reuse, easier testing, and maintenance through the use of groupings or sets of rules bound together in a single node of the taxonomy or hierarchy.

LITI itself is flexible and scalable: For beginning users and simple tasks, it can be a simple tool; in the hands of an advanced user, it can also be a complex tool for accomplishing complex tasks. For example, a beginning user can easily model a list of keywords to find all of the instances of those words in the data or easily model and find a sequence of elements in text, while a more advanced user can model facts, events, aliases for names, coreference, or grammatical relationships, such as subject-verb-object (SVO) triples.

LITI is powerful but efficient. Rules in LITI are typically short enough to be represented on a single line, and they are human-readable and maintainable. The syntax includes a set of rule types that helps you to target the rule to the type of matching behavior needed, while using only the processing resources required. In this fashion, you have direct control over optimizing the time it takes a model to process data. Furthermore, the set of Boolean and proximity operators included in LITI are designed specifically to capture and extract both very simple and very complex information and relationships from text. LITI also contains additional modifiers and features to allow you to use POS tags to represent tokens, identify other types of placeholder tokens in patterns, mark the exact pieces of text you want returned, and find related matches within the same document.

Because of these benefits, SAS IE can ultimately be used for building precise and powerful models that represent human knowledge and language patterns. These models can be applied to new data and problems either in full or by repurposing pieces of one model in a new one.

The rest of this book focuses on using predefined and custom concepts in isolation and in various types of combinations, but also points out ways where you may be able to leverage existing lists, automatically generated categorization rules, or automatically generated LITI rules to speed your development process. SAS IE, in other words, can help you to solve many business problems in isolation and this book focuses on helping you understand and practice doing so. Once you have gained proficiency in using LITI rules, you will see the many ways you can integrate SAS IE models into the larger context of your analytics operations, as well.

1.4. When You Should Use Other Approaches instead of SAS IE

SAS IE is a powerful way to model patterns in text, to extract information, and to relate different pieces of information to each other. But other tools can also do some of these tasks, and knowing when to select IE instead of another tool is an important part of designing your analytic approach to a problem. The optimal choice will not always be clear, and sometimes experimentation may be required. You may also decide to use SAS IE instead of another option because you are comfortable with the syntax or rule type choices or simply because you like the flexibility of being able to extract or relate information whenever you want to.

Although there are no overarching rules to text analytics, some analytic scenarios lend themselves to one approach over another, and this section will describe some of those situations and mention some of the alternate options to get you started with making these decisions. The most important factor for deciding whether to use IE is what information you want to get as a result of your analysis.

Tip: Pay attention to the output you need from the analysis, because this is a good criterion for deciding which IE approach to leverage.

What if you do not know your text data very well and you already have a predictive model? You want to see whether there is anything valuable in the unstructured text data that could provide some lift for your model. In this case, it may be possible to reach your goals by using either topic modeling or clustering. Topics and clusters create new structured data, like IE does, but because they are unsupervised machine learning tasks, you have no direct control over the data they create. They identify inherent patterns in the data, which may or may not be related to your interests directly. Therefore, it is more difficult to diagnose the cause of the lift of the predictive model. However, the benefit of these approaches is that they are self-driven and do not require subject matter expertise or a predefined target. Either of these strategies may provide sufficient lift for your predictions. Later, if you want even more lift, you might try adding an IE model to this process; see Albright et al. (2013) for a description of this process.

Another scenario that may not lend itself to IE is when you have a set of documents that you want to allocate to various buckets or categories. You have criteria for placing these documents into each category, or you have data that can be used to train such a model. In these circumstances, using SAS categorization will likely be more efficient than IE, because categorization identifies each document as “in” as opposed to “out” of each category that you define. You can also derive categories from topic models. With this approach, there is no overhead for extraction of additional information from the documents. In addition, categorization uses only one type of rule, which may be simpler for some users. However, that rule type is still powerful, and authoring good rules requires some practice. You can categorize documents as a part of your IE model, so if you want to extract data *and* put documents into categories, you should use SAS IE instead or, even better, use a combination of extraction and categorization. The SAS Text Analytics products enable the categorization model to reference rules in the IE model directly. Combining category and concept models is also a good approach if your categorization rules are getting very large and difficult to maintain.

If you want to detect sentiment or attitudes in your data, you should examine the capabilities of SAS Sentiment Analysis. It has great features for automatic, custom, or hybrid methods of creating models. However, it is also possible to leverage SAS IE or SAS categorization for this purpose. SAS IE may be the way to go if you want more detail about the exact words or phrases used to describe the positive or negative position the writer is expressing or to identify the particular feature that is being noted as positive or negative. Otherwise, a categorization approach for document-level sentiment would likely be more efficient.

If you are trying to simply normalize text or do some very mechanical operation to transform text data, then it may be possible to use SAS regular expressions or a language such as Python. SAS IE is probably not the best choice for such situations. However, once you pass the threshold of starting to care about meaning, then SAS IE becomes a relevant option again. Meaning is easier to model with the kinds of tools SAS IE supplies than in regular expressions, which focus on a character-by-character analysis. Types of normalization or transformation that rely on linguistic information, such as sentence boundaries, are easy and quick to do through SAS IE, as well.

If you are trying to do your own foundational NLP, and you want to learn how to find tokens, break sentences, tag words, or the like, then the SAS IE system will not expose those types of tasks. The SAS foundational NLP is mostly hidden from the users to enable users to focus on models for IE or other methods of analysis. That said, you can use SAS IE to expose sentences, generate *n*-grams, or find sets of related items, even triples like SVO.

1.5. Important Terms in the Book

Before continuing, take some time to familiarize yourself with terminology that is important for following the ideas presented in this book. This section explains terms used at SAS, as well as common terms used in the IE field.

1.5.1. Strings versus Tokens

In text processing, *string* means a defined set of sequential characters that may include punctuation but not a new line character. One example of a string is “123 Main Street Apt. 3B.”

Note that the term *string* is not the same as *token*. In NLP systems, a *token* is the basic unit of analysis. In SAS, tokens are intended to represent a meaningful unit, identified by taking alphanumeric characters, spaces, punctuation, and special characters, and determining where to draw boundaries between them. Returning to the example just given, the string contains the following tokens:

- 123
- Main
- Street
- Apt.
- 3B
- .

Usually, a token will not contain an internal space, unless it is defined in the system as a *multiword*. A multiword is a set of two or more words that is defined as a token because the combination of the two words means something different than would be expected by the simple meaning of each word, and it functions as a single grammatical unit. An example of a multiword is “high school.”

1.5.2. Named Entities and Predefined Concepts

The phrase *named entities* historically refers to one or more words or numeric expressions in sequence that name a single individual or specify an instance of a type in the real world (or an imaginary world). Some of the most common named entities are names of persons, locations, and organizations, as well as currency amounts, dates, times, and percentages. You can learn more about named entities in chapter 2.

The SAS Text Analytics software provides out-of-the-box rule sets for extracting named entities from text. These rule sets are called *predefined concepts*. In this book, the phrase “named entities” describes the generic idea or historical work, and the phrase “predefined concepts” refers to the SAS implementation of this idea. For more about each of the predefined concepts that SAS provides, see chapters 3 and 4.

1.5.3. Parent Forms and Other Variants

In many dictionaries, a single form of a word represents multiple useful words. For example, if you look up the noun “apple,” then you will find the singular form “apple,” but not the plural form “apples.” The form of the word you find in a dictionary is known as the *lemma* or *base form* of the word. Most lemmas have *inflectional variants*, like plural forms for nouns or past tense forms for verbs, that will not be found in the dictionary. These inflectionally related words are considered to be a set in linguistics and in NLP.

Another type of a word set is a proper or accepted spelling as opposed to *spelling variants*. Spelling variants may include incorrect spellings and typos, as well as variants across dialects such as British or American English. Finally, in some linguistic analyses, it is useful to create broader sets of words that may be considered *synonyms* of each other or have some other semantic similarity.

The SAS Text Analytics products can recognize each of these three sets of related word forms and can group them together. The products enable access to the sets through different mechanisms, and you should consult your product documentation for details. In all cases, when showing such sets of forms, the forms are called *terms*. The lemma, properly spelled form, or a user-specified form is called the *parent term* and the other members of the set are the *child terms*.

1.5.4. Found Text and Extracted Match

To be clear when discussing the fundamentals of writing and applying LITI rules, this book uses two sets of terms: *found text* and *extracted match*. LITI rules contain specifications for matching a span of text, which is the *found text*. In some examples, the entire matched span needs to be extracted, but in others it does not. Therefore, a different set of terms is needed for the part that is extracted as the output or result, the *extracted match*. For example, a rule could match the string "Beatles band members" (found text) but return as output only the band name, "Beatles" (extracted match), when it is followed by the string "band members."

1.6. Suggested Reading

To gain more background in the field, consult the following sources in the References list at the end of the book:

- Albright et al. (2013)
- Belamaric Wilsey and Jade (2015)
- Piskorski and Yangarber (2013)
- Sabo (2015)
- Sabo (2017)
- Sarawagi (2007)
- Small and Medsker (2014)