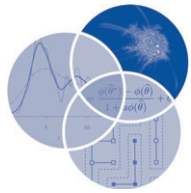
Overview



At the interface of computational linguistics and statistics

Angel Martinez and Wendy Martinez*

Computational linguistics encompasses a broad range of ideas and research areas, and only a brief introduction is possible here. We chose to include areas in computational linguistics where statisticians can contribute, hoping to provide inspiration to the reader. We describe three main aspects of this discipline—formal languages, information retrieval, and machine learning. These support the overarching goal, which is the representation and analysis of meaning from unstructured text. We then provide an example where text analysis has been applied to unstructured text fields in survey records and conclude with some applications and computational resources.

How to cite this article:
WIREs Comput Stat 2015, 7:258–274. doi: 10.1002/wics.1353

Keywords: document classification and clustering; text data mining; machine translation; grammars; latent semantic analysis; language models; natural language processing; part-of-speech tagging; text summarization

INTRODUCTION

Definition of Computational Linguistics

What is computational linguistics? The answer to the question is not easily stated. Even members of the Association of Computational Linguistics (ACL)[1] have difficulty articulating a definition:

> Simply put, computational linguistics is the scientific study of language from a computational perspective. Computational linguists are interested in providing computational models of various kinds of linguistic phenomena.[2]

According to the ACL web site, the work of computational linguists is said to be incorporated in systems for speech recognition, text-to-speech synthesizers, web search engines, and others.[2] In addition, the definition found in the Wikipedia goes further, including the 'area of computational complexity of natural language, the use of automata theory, Turing machines, and the use of logic systems to represent and reason with natural languages'.[3]

*Correspondence to: martinez.wendy@bls.gov

Bureau of Labor Statistics 2 Massachusetts Ave, NE Washington, DC 20212, USA

Conflict of interest: The authors have declared no conflicts of interest for this article.

In his paper on the impact of the statistical revolution on '(computational) linguistics', Johnson[4] does not see much importance of an actual definition for computational linguistics, especially when the state-of-the-art 'draws more heavily on statistics and machine learning than it does on linguistic theory'.[4] In the conclusion section of his paper, Johnson candidly admits wondering 'If by computational linguistics we mean the natural language processing (NLP) engineering applications that typically receive the bulk of attention at today's computational linguistics conferences'.[4]

So, what is computational linguistics? This question is better answered by describing the interdisciplinary work that brings together statistics, linguistics, computer science, machine learning, data mining, artificial intelligence, and much more.[5,6] We hope the reader will come to understand computational linguistics by the description of approaches and applications in this paper.

The Three Legs of Computational Linguistics

Between the late 1950s and early 1960s, two paradigms emerged. First, was the paradigm where language was described symbolically via structures

and rules or grammars. Those with this perspective were called *rationalists*. To rationalists, language structures are innate and are a part of the genetic makeup of human beings. The second paradigm led to a group called the *empiricists*, who accept the rationalist idea that there are innate structures; however, these basic structures work like functions to sift and organize the input from our senses. The empiricists argued that a better language model would be based, not on grammars, but by the application of statistical methods.[7,8]

The rationalists continued to work on language theory and generative grammars (*Formal Language Theory*). Their goal was to create a set of computable symbols for the whole of the English language. This would be the first step in endowing computers with the ability to 'read' and 'speak' English, and in the process advance their main goal of automatic machine translation (MT).

Unfortunately, the rule-based models of language encountered major problems. For instance, the parsing ('reading') of text using grammar rules became too costly—if not impossible—computationally.[9] There were also the additional problems of ambiguity, such as anaphora (reference to a previous entity) and figures of speech. The empiricists, however, found their niche in what was called artificial intelligence. Their holy grail was to endow computers with the capacity to 'understand' natural languages like English, and natural language understanding (NLU) became their basic objective.

The rationalists eventually moderated their goal and applied grammars and symbols to the study of syntax, phonology (the study of speech sounds), morphology (how words are formed in a language), and psycholinguistics.[7] In a similar manner, the empiricists made their goal more modest—from natural language *understanding* to natural language *processing*.[10] Much of the advances attained in computational linguistics have come from the stochastic and data-driven approaches in the area of NLP, specifically information retrieval (*IR and Computational Linguistics*).

One of the reviewers of this paper suggested a three-legged stool model for computational linguistics, where one leg corresponds to researchers focusing on formal language and another to those concentrating on information retrieval (IR) applications. The third leg would be represented by researchers using statistics and machine learning approaches for the analysis of text-based data.

We liked that analogy and decided to expand on it. In our opinion, we can characterize the ultimate goal of applications in computational linguistics as one of extracting the meaning from text (or speech) using machines. For example, MT is concerned with translating text from one language to another in a way that preserves meaning. IR approaches seek to find documents that are relevant to the meaning expressed by the search query, and statisticians might use unsupervised learning (or clustering) techniques to group documents with similar meaning or topic. So, one could say that the seat of the stool is the *machine representation of meaning* (*semantics*), which is supported by all three legs.

Semantic analysis is discussed in *The Challenge: Representation of Meaning* section, where we provide a brief overview of some ways to encode text with the goal of capturing meaning. In *Formal Language Theory* section, we describe the foundation of formal languages and their grammars. The next section (*IR and Computational Linguistics*) delves into some methods related to computational linguistics, as developed by the IR community. *Statistics and Machine Learning in Computational Linguistics* section looks at contributions from the machine learning discipline, and *Example Using Statistical Methods for the Analysis of Text* section provides an example using some of these statistical methods. *Applications* section includes a brief description of applications in computational linguistics where statisticians can contribute. These include MT, part-of-speech tagging, word sense disambiguation, and text summarization.

THE CHALLENGE: REPRESENTATION OF MEANING

As we just mentioned, the issue of semantics is key to those working in computational linguistics. A working definition of semantics could be: *the study of the meaning of utterances—from text or speech*. The study involves the extraction of the meaning from text and the representation of this meaning in machine-friendly form. When adding the tasks of capturing and representing meaning, the subject is then called *semantic analysis*.

Researchers have found that representing semantics of text via computers is a difficult problem. There are challenges in the semantic analysis process, such as

- finding an automatic and efficient way to distill the semantics from text
- creating appropriate representations that preserve meaning
- dealing with the problem of ambiguity in all its forms

Where in a sentence or a phrase is meaning hiding? Is it in the syntax, the words and symbols or a combination of these? The implication that meaning arises from the relationship between syntax and the language's syntactic units (symbols, words) became an important principle of linguistics and computational linguistics. This was called the *principle of compositionality* and is attributed to Gottlob Frege.[11] The principle states that the meaning of an utterance is the sum of the meanings of its composing units (words) and how these units are put together (syntax).[12,13]

Although key to some areas of research in linguistics, compositionality is more of a theoretical model than a readily usable framework for machine implementations. A competing linguistic concept states that the meaning of an utterance is intrinsically linked to its context. This idea is attributed to John R. Firth, whose dictum, 'You shall know a word by the company it keeps',[14] pithily expresses a new way of understanding natural language semantics. However, it was the work of Zellig Harris that set the concept on solid ground with his paper 'Distributional Structure'.[15]

Distributional or Vector Models

Distributional models are the preferred form of representation today, and they have sparked a considerable amount of research. Using the distributional approach to capture phrase and sentential meaning has been found to be hard and not many have tackled the problem. An exception to this is latent semantic indexing (LSI),[16] which is based on the vector space model (VSM).[17,18] This was developed in the IR community; details on LSI are provided in *IR and Computational Linguistics* section.

While the LSI approach focuses on words and their contexts in large corpora, the underlying idea is that an aggregation of all the contexts in which the word appears, or does not appear, sets mutual constraints that determine the degree of similarity of meaning given by words.[19]

The word *latent* in LSI indicates that the results from the representation are evidence of deeper relations.[19] This property is the reason for its success in numerous applications in linguistics and NLP. Some applications include predicting query-document topic similarity, simulation of human choices in multiple-choice tests, predicting text coherence and comprehension, IR, and more.[19–22] LSI shows, among other things, that a distributional approach can address meaning beyond the word level.[23]

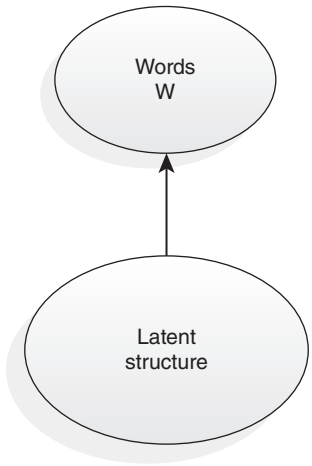


FIGURE 1 | Generative model where a latent structure generates words *w*.

Some representations based on the compositionality principle also use vectors to represent text documents, but they include syntactic features as part of the representations. In the distributional representation, the words in the vector are no more than a 'bag of words'. However, vectors are at the center of semantic representation modeling for both the compositional and the distributional approaches.

Topic Models

Another interesting approach to semantic representation tries to capture the 'gist' of a sentence or document as a way to predict meaning. This approach combines statistical unsupervised learning (*Statistics and Machine Learning in Computational Linguistics*) and structured representations via a generative model (*Formal Language Theory*). A key result is a generative model to define a probability distribution over a set of words or a document, given a probability distribution over topics.[24] The application of Bayes statistics facilitates the unsupervised learning of a set of topics from a document.

Figure 1 shows the generative model where words are generated by first sampling a latent structure l from a distribution over latent structures $P(l)$, then a sampling of words w conditional on the latent structure $P(w|l)$. The sampling from this distribution defines a joint distribution between the observed data and latent structures[24]:

$$P(w,l) = P(w|l)\,P(l)\,.$$

Using probability theory, we can write

$$P(w,l) = P(w|l)\,P(l) = P(l|w)\,P(w)\,,$$

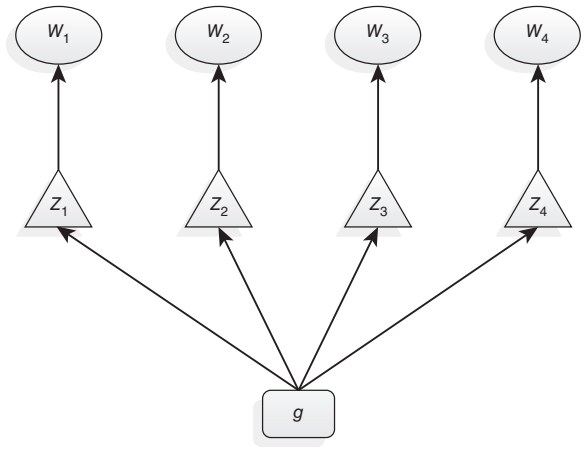


FIGURE 2 | Associating gist with topics.

where

$$P(l|w) = \frac{P(w|l)\,P(l)}{P(w)} = \frac{P(w|l)\,P(l)}{\sum_{l} P(w|l)\,P(l)}. \quad (1)$$

This provides the probability of the latent structure, given the set of words in the document. Griffiths et al.[24] show how these concepts can be used for predicting the next word in a sequence, disambiguation, and gist extraction.

Figure 2 provides a view of a topic model (also a generative model), where each topic is a probability distribution over words. A set of words (a document) is generated by choosing the distribution over T topics reflecting its meaning.[24] A topic z_i can be chosen for each word w_i using the same distribution over words related to the topic. The probability of the ith word is given by

$$P(w_i|g) = \sum_{z_i=1}^{T} P(w_i|z_i)\,P(z_i|g), \quad (2)$$

as defined by the generative process and g being the gist of the document. Equation (2) indicates that the topics specified by $P(w|z)$ are mixed and the weights are given by $P(z|g)$, which varies over documents.[24] Thus, the gist of a document can be represented by a mixture of topics.

Researchers found the topic model to perform measurably better than other approaches like LSI. In addition, it was shown that the topic model addresses three key computational problems: prediction (predicting word w_{i+1} from w_i), disambiguation (inference of a topic by each word from the distribution determined by the gist), and gist extraction (inference of g from w).

If we replace the gist g with a document D, then we get the approach known as probabilistic latent semantic analysis (PLSA). We can use this model to generate a document collection within a probabilistic framework. One result from this approach is the probability of a word w for each of the topics, as given by $P(w|z_i)$. Hofmann[25] showed how the Expectation-Maximization (EM) algorithm[26] can be used to estimate the required probabilities. He also discussed how the latent classes correspond to a topic or cluster, and the top words for each latent class can be used to characterize the topic.

Latent Dirichlet Allocation—LDA

There are no assumptions about the weights with PLSA. Blei et al.[27] extended this model by putting a Dirichlet prior on these mixing proportions and called it *latent Dirichlet allocation* (LDA). This provided a generative probabilistic model for a corpus. The basic assumption is that the documents can be modeled as random mixtures over latent topics, and each topic is characterized by a probability distribution over words.

Blei et al. showed how LDA can be used for document modeling and classification, among other things. Steyvers and Griffiths[28] have a good discussion of probabilistic models, including how they can measure two types of similarities—between documents and between words. Anthes[29] provides a good overview of LDA and other topic models. Interestingly, it includes a quote from Fernando Pereira of Google who discusses how these models are being investigated at Google with the hope that they would provide improved search results. Pereira states that 'While they are intriguing, we haven't yet gotten to the point that we can say, "Yes, this is a practical tool"'.[29]

An interesting research area is the use of topic models to characterize microblogs like Twitter content. Ramage et al.[30] discuss some of the information needed for such an analysis. They also present a partially supervised learning model called labeled LDA to represent the content of the Twitter feed.

The topic-model approaches are not aimed at the deeper meaning of words; there is no effort in achieving an understanding of the world. However, this methodology facilitates the automatic training from corpora via unsupervised learning of the distributions of words and topics, which is essential for a machine implementation.

FORMAL LANGUAGE THEORY

Formal language theory is the mathematical study of languages, some of which are created (computer

languages), and some of which occur naturally (English). A formal language according to Hopcroft and Ullman[31] is '… a set of strings of symbols from some *one* alphabet'. A more detailed definition is found in Revesz,[32] where the author starts by defining a set of symbols (letters) as a finite alphabet V. Finite strings of these symbols are called words and belong to V^*, the vocabulary. This arbitrary set of words is called a language. This definition is applicable to all written natural languages, as well as artificial ones. However, the definition as it stands is not very useful. One thing is missing: *how* to define a particular language.

Grammars and Chomsky's Hierarchy of Language

If the language were to have only a finite set of sentences, a complete list of these would be the definition of the language. However, if the language has an infinite number of sentences, then some set of rules to generate the language needs to exist. For this set of rules to be capable of producing an infinite number of sentences, it has to be self-referencing or iterative. That is, it has to provide for automatic recursion. We call this set of rules—a *grammar*.[33] A grammar does not need to have an infinite set of rules, just a finite set of production rules describing how to use the words in its vocabulary to generate sentences from the language.

The set of production rules serve as guidelines to re-use its own rules in different ways to generate sentences in the language defined by the grammar. Thus, these grammars are known as *generative grammars*. Noam Chomsky developed a classification of generative grammars often called the 'Chomsky Hierarchy of Languages'.[32] There are four types of languages generated by the following grammars:

Language	Grammar
L_0	Unrestricted
L_1	Context-sensitive
L_2	Context-free
L_3	Regular or finite state

The Chomsky hierarchy is given by $L_3 \subset L_2 \subset L_1 \subset L_0$, where all the inclusions are proper.

Context-Free Grammars

The L_2 languages generated by context-free grammars (CFGs) have a wide use in the creation of computer languages, e.g., FORTRAN, C, and others. Natural languages, like English, are also in the category of context-free languages. However, unlike computer languages, natural languages contain anaphora (a reference to a previously discussed entity), polysemy (a word can have several meanings), synonymy (different words have the same meaning), and other sources of ambiguities. Because of this, a grammar for the complete English language has been deemed impossible. The richness of expressions, the time complexity, and the computational load on system capacity add to the problem.[34] Nonetheless, enough CFGs have been produced for well-defined subsets of English.

Noam Chomsky opened the door to the automatic parsing of natural language with generative grammars. Parsing is the process of assigning syntactic types to components of sentences. The automatic parsing of natural languages facilitated the creation of structures for knowledge (or meaning) representation.

Let us look at the definition of generative grammars. A generative grammar G is an ordered four-tuple $G = (V_N, V_T, S, R)$, where V_N is a finite set of grammar symbols called the vocabulary, V_T is the set of terminal symbols, S is the start symbol, and R is a finite set of productions or rules. R contains ordered pairs (P, Q), which are usually written in the form $P \rightarrow Q$.

The following example illustrates a generative grammar. Let a grammar G be defined as follows:

$$G = \left(\{\text{Se, NP, N, Ve}\}, \{\text{Fido, runs}\}, \text{S, R}\right),$$

where Se stands for sentence, NP is a noun phrase, N is a noun, and Ve denotes a verb. We have the following set of rules R:

1. S → Se	3. Se → NP	5. NP → N N	7. N → runs
2. Se → NP Ve	4. NP → N	6. N → Fido	8. Ve → runs

If a parser were to generate the sentence *Fido runs* using the grammar G (or equivalently, will check to see if *Fido runs* is a sentence in the language defined by G), it would check the production rules in the manner shown below. These are called *parse trees*. Two possible parses are shown in Figure 3.

The sentence that was parsed as *runs Fido* belongs to the language according to the grammar, but it does not make sense. This illustrates one problem with CFGs—their inability to handle ambiguities.

The issue of ambiguity led to the development of *probabilistic context-free grammars* (PCFG).[35] Say we have a corpus (or body) of documents where each word has been tagged with the appropriate

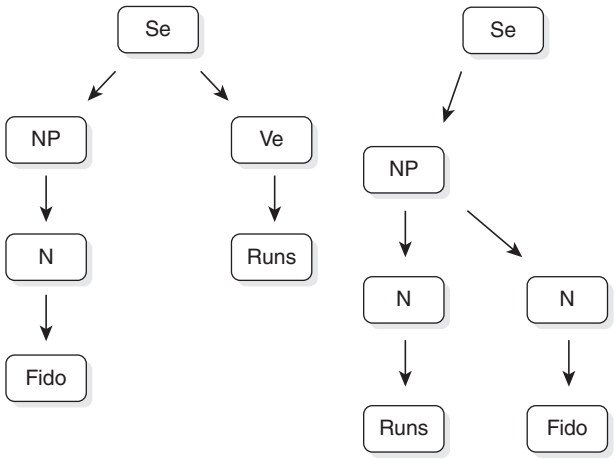


FIGURE 3 | Two possible parses of the sentence *Fido runs*.

1. S → Se (1.0)	3. Se → NP (0.3)	5. NP → N N (0.2)	7. N → runs (0.4)
2. Se → NP Ve (0.7)	4. NP → N (0.8)	6. N → Fido (0.6)	8. Ve → runs (0.8)

part of speech. This can be used to estimate and assign probabilities to the rules in the grammar, which are then used to determine the most likely parsing. For example, our grammar G from above has the following probabilities associated with its rules.

The correct parsing used productions 1, 2, 4, 6, and 8, yielding a joint probability of $1.0 \times 0.7 \times 0.8 \times 0.6 \times 0.8 = 0.2688$. The alternative parsing used rules 1, 3, 4, 7, 4, and 6. This gives a probability of $1.0 \times 0.3 \times 0.8 \times 0.4 \times 0.8 \times 0.6 = 0.0461$, indicating that the correct parsing is more likely.

Although some improvement over CFGs is gained with the PCFGs, the ambiguity problem is not solved. A PCFG just prefers those constructions that are more common in the tagged corpus. Charniak[35] suggests that the approach could be improved by looking at prepositional phrases, using the same probabilistic techniques, and considering the frequency of tags occurring in relation to specific words.

IR AND COMPUTATIONAL LINGUISTICS

We saw that a grammar is a model of a language for the specific purpose of generating or identifying strings from that language. Other approaches are being used to process and represent natural languages for applications such as machine understanding, IR, document classification, and more. Perhaps, the most common one developed in the IR community is the VSM.[17,18]

TABLE 1 | A Word-by-Document Matrix

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
w_1	1	1	1	0	1	0	1	0	1
w_2	0	1	0	1	1	0	1	1	0
w_3	1	1	0	1	1	0	0	0	0
w_4	0	0	1	0	0	0	0	1	1
w_5	0	0	0	0	0	1	1	1	1

The Vector Space Representation

One of the first applications in computational linguistics addressed by the empiricists was to have computers search for information in collections of electronically pre-processed text documents. This is called *information retrieval*. IR methods are the essence of search engines in use today.

Using vectors to capture the words in a document is the basic pre-processing task in IR systems. A collection (or *corpus*) of documents can be represented by a set of vectors by converting each document to a vector of size m, where m is the number of unique words in the lexicon (or vocabulary). The ith element of the vector is a 1 or 0, indicating the presence or absence of the ith word in the document. The *word-by-document matrix* **A** is formed by placing each document vector as a column in the matrix. The matrix is of size $m \times n$, where n is the number of documents in the corpus.

The following example will clarify this idea. Let us say we have a very small collection of nine documents, with a word-by-document matrix shown in Table 1.

We see that document d_1 contains words w_1 and w_3, while document d_2 has words w_1, w_2, and w_3, and so on. Let us say we want to retrieve information by finding all the documents containing w_2 and w_3. First, we create a query vector using the words of interest:

$$\mathbf{q} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \end{bmatrix}^{\mathrm{T}}.$$

Next, we need to compare the query vector against each of the columns (documents) of matrix **A**. A simple and computationally feasible method to compare them is to compute a similarity measure between the query vector and each of the document vectors. A commonly used measure of similarity is based on the cosine of the angle between the query and document vectors. The higher the value, the more

relevant or similar the document is to the query. The cosine similarity measure is given by

$$\cos\theta = \frac{\mathbf{q}^{\mathrm{T}}\mathbf{d}}{\|\mathbf{q}\|\,\|\mathbf{d}\|},$$

where the symbol $\|\bullet\|$ indicates the magnitude of a vector. If more than one relevant document is returned from the query, then a distance threshold could be pre-specified.[36]

The binary representation works well in many applications. More discrimination power can probably be obtained if instead of ones and zeroes, each cell in the word-by-document matrix indicates the frequency of occurrence of the terms w_i in the document d_j.

One could also use term weighting. Several approaches were developed by the IR community to improve the retrieval of documents relevant to the query and to reduce the number of irrelevant documents that are returned. Berry and Browne[36] provide an excellent discussion of term weighting.

They let the element of the word-by-document matrix be represented by

$$a_{ij} = l_{ij} g_i d_j,$$

where l_{ij} is a local weight for the ith word in the jth document, g_i is a global weight for the ith term in the lexicon, and d_j is a normalization factor for the document. Given that the relevance of a document depends on its length, it is more likely that a longer document will be considered relevant. The document normalization factor d_j is used to address this issue and to improve retrieval performance.

Some local weights l_{ij} have been discussed already, i.e., the binary representation or the frequency of word i in document j. Berry and Browne[36] state that term frequencies are satisfactory when the corpus contains general documents (e.g., news articles), while the binary encoding is suitable when the domain of discourse is constrained (i.e., the lexicon is relatively small).

Whether or not to use a global weight g_i depends on the corpus. If it changes often, then one might not want to use it because the weight impacts all of the rows of the word-by-document matrix. A commonly used global weight is the inverse document frequency (IDF) given by

$$g_i = \mathrm{idf_i} = \log\left(\frac{n}{\mathrm{df}_i}\right),$$

where n is the number of documents in the corpus and df_i is the number of documents that contain the ith word. If a word occurs in a lot of documents, then the IDF will be small, and it down-weights the word frequency. In fact, if a word appears in every document, then we have $\mathrm{idf}_i = \log(1) = 0$.

The logarithm is used in the IDF because it dampens or compresses the scale of the values. It also enables us to get a composite document score by adding the IDFs for query words, assuming the occurrence of words are independent. The tf-idf is the term frequency multiplied by the IDF and is used often in IR and statistical text analysis applications.[36,37]

As one might suspect, the word-by-document matrix tends to be sparse (non-zero entries are few). Efficient storage and processing approaches exist to handle these types of data structures, making it possible to handle large corpora. Several refinements have been proposed in the IR community to help with large corpora and to improve retrieval, such as removing stop words (words in the vocabulary that provide little information) and stemming (conflated forms are eliminated).[38]

Latent Semantic Analysis

LSI was developed by Deerwester et al.[39] for automatic indexing of documents to facilitate IR. The goal is to exploit the latent semantic structure to enhance retrieval efficiency and produce better results. It employs the singular-value decomposition (SVD) technique from linear algebra to factor the word-by-document matrix. The same approach is called latent semantic analysis (LSA) in the literature, and LSI and LSA have come to be used synonymously. However, LSI typically refers to the use of this technique for IR, while LSA refers to other applications.

The SVD factorization replaces individual words with derived orthogonal factor values. These factor values may be thought of as artificial concepts that represent extracted common semantic components of many different words and documents. Each word or document is then characterized by a vector of weights indicating its strength of association. That is, the meaning of a particular term, query, or document can be expressed by k factor values or equivalently, the location of the vector in the k-space defined by the factors. LSA approximates the original m-dimensional word space by the first k component directions in this space obtained by the SVD. Precision of retrieval is enhanced by this method when compared to other vector space approaches.[40]

In LSA, SVD decomposes the word-by-document matrix $\mathbf{A}$ to form the latent semantic representation,

TABLE 2 | A Small Corpus with a Subset of Words for the Lexicon

Documents in Corpus	Words in Lexicon
How to *Bake Bread* without *Recipes*	bak (e, ing)
The Classic Art of Viennese *Pastry*	recipes
Numerical *Recipes*: The Art of Scientific Computing	bread (s)
Breads, Pastries, Pies and *Cakes*: Quantity *Baking Recipes*	cakes
Pastry: A Book of Best French *Recipes*	pastr (y, ies)
	pies

which takes the following form:

$$\mathbf{A} = \mathbf{TSD}^{\mathrm{T}},$$

where $\mathbf{T}$ is the matrix of left singular vectors; $\mathbf{S}$ is the diagonal matrix of singular values; and $\mathbf{D}$ is the matrix of right singular vectors. By convention, the diagonal elements of $\mathbf{S}$ are ordered in decreasing magnitude. To reduce noise, the highest singular values of $\mathbf{S}$ and their corresponding left and right singular vectors are kept to obtain a matrix $\tilde{\mathbf{A}}$ of rank $k < n$:

$$\tilde{\mathbf{A}} = \mathbf{T}_k\mathbf{S}_k\mathbf{D}_k^{\mathrm{T}},$$

This procedure simultaneously removes noise from the corpus and ensures that semantically related documents are close together in the new space.

We can obtain three types of semantic comparisons: between two documents, two words, and a word and document.[39] Comparison between two words is obtained by the word-to-word inner products contained in $\tilde{\mathbf{A}}\tilde{\mathbf{A}}^{\mathrm{T}}$, while comparison between two documents is attained by the inner products in $\tilde{\mathbf{A}}^{\mathrm{T}}\tilde{\mathbf{A}}$. To compare a word and a document, we look at the ijth element of $\tilde{\mathbf{A}}$.

The following example is from Berry et al.[41] The set of documents contain five book titles (see Table 2). A subset of words from the lexicon has been used to make it easier to explain.

The word-document matrix is shown here:

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

We want to find documents on *baking*, so our query vector is

$$\mathbf{q} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}.$$

The cosine similarities between the query and document vectors are 0.58, 0, 0, 0.41, and 0. Using a cutoff value of 0.5 would return the first document, but would miss the fourth, which is also relevant. If we use a matrix $\tilde{\mathbf{A}}$ obtained from SVD using $k = 3$, then we have cosine similarities of 0.52 and 0.51 for documents 1 and 4, and we would retrieve the additional relevant document.

STATISTICS AND MACHINE LEARNING IN COMPUTATIONAL LINGUISTICS

Using the vector space representation for documents has contributed to the successful introduction of computational statistics methods for document manipulation and analysis. In this section, we will consider a couple of statistical methods used in the area of machine learning.

One of the earliest contributions made by statisticians to the analysis of text was conducted by Mosteller and Wallace.[42] They attempted to solve questions about the authorship of *The Federalist Papers* that includes 77 essays published anonymously in 1787 and 1788. It is generally known that Alexander Hamilton wrote 43, John Jay wrote 5, and James Madison authored 14. The authorship of the remaining 12 is in dispute between Hamilton and Madison.

Mosteller and Wallace used word counts as the variables or features and built a classifier to predict the author of the disputed essays. They chose non-contextual or stop words because these would be invariant to the essay's meaning (or topic) and are indicative of an author's writing style. They had a corpus of documents written by Hamilton and Madison, from which they constructed a naïve Bayes classifier. The classifier was used to assign an author (or class label) to the disputed documents, and they found that Madison wrote all 12 essays. We describe the problem of document classification in more detail next.

Document Classification

Structuring a document collection enhances the effectiveness of IR and analysis tasks. An intuitive way to structure it is by classifying documents as belonging to one of a pre-defined set of categories. We can build a classifier by obtaining a corpus where each document is labeled with a known category (topic, author, etc.), which serves as a training set to build the classifier. Once we have an appropriate document representation and a classifier, each new document can be associated with one of the categories. See Harish et al.[43]

for a brief discussion of different text representations and their use with classifiers.

Methods to create classifiers using a labeled training set are called *supervised learning* techniques. As with any classifier, a key step is to determine what features or variables should be used. If we use the vector space representation, then the features are a function of the word counts in the document. Thus, we potentially have very high-dimensional data implying that we have fewer observations (documents) than variables. We will briefly discuss some of the most common classifiers used on document collections; see Duda et al.[44] and Hastie et al.[45] for more information on these methods and others.

The *naïve Bayes classifier* is obtained by computing the probability that document d_i belongs to class C_j given by $P(C_j|d_i)$ for $j = 1, \ldots, J$, where J is the number of categories.[44,46] This probability is known as the *posterior class probability*. The document is assigned to the class that has the highest posterior probability.

To build the classifier, we have to estimate these class probabilities. From Bayes' rule, we have

$$P\left(C_j|d_i\right) = P\left(C_j|w_{1i}, \ldots, w_{ni}\right) = \frac{P\left(d_i|C_j\right)P\left(C_j\right)}{P\left(d_i\right)}$$
$$= \frac{P\left(w_{1i}, \ldots, w_{ni}|C_j\right)P\left(C_j\right)}{\sum_k P\left(w_{1i}, \ldots, w_{ni}|C_k\right)P\left(C_k\right)},$$

where we are replacing the document d_i with the words it contains.

During the training or learning step, $P(w_{1i}, \ldots, w_{ni}|C_j)$ is estimated for each class using the relative frequencies of these in the training documents labeled with class C_j. The prior probability $P(C_j)$ is estimated as the relative frequency of the documents in each class.

However, for

$$P\left(d_i\right) = \sum_k P\left(w_{1i}, \ldots, w_{ni}|C_k\right)P\left(C_k\right),$$

we need to assume that each word is independent of the others. This allows us to write

$$P\left(w_{1i}, \ldots, w_{ni}|C_k\right) = \prod_{j=1}^{n_i} P\left(w_j|C_k\right),$$

and to calculate the probability of the jth document as

$$P\left(d_i\right) = \sum_k \prod_{j=1}^{n_i} P\left(w_j|C_k\right)P\left(C_k\right).$$

We can estimate the required probabilities using the labeled corpus.

The k-nearest neighbor (k-NN) classifier is well-suited for observations with many features (like documents) because only pair-wise distances are considered in the multidimensional space.[47] The k-NN method also uses probability estimation[48,49] to classify observations. However, we skip the math and just explain it in words. Say, we need to classify document d_i. The k-NN classifier polls the k-nearest neighbors to the document and checks which class has a majority vote among the neighbors. This class label is then assigned to d_i.[49,50]

Other supervised learning methods can be used on document collections. One of these is classification trees and random forests.[51] An advantage of a classification tree is that it can handle data sets with many dimensions and different types of variables, e.g. categorical and continuous. The decision tree classifier divides a variable at the nodes in such a way that the observations in subsequent leaf nodes are as pure as possible with respect to the class labels. The basic approach of random forests is to build an ensemble classifier based on many trees constructed in randomly selected subspaces and take a majority vote of the ensemble as the predicted class.

A less popular method is the use of artificial neural networks (ANNs) to document clustering.[52] ANNs have their origins in the attempt to find a mathematical model for the information processes of biological entities. Beyond this original goal, ANNs have found application in statistical pattern recognition and classification, among many others.[53] The interested reader is referred to Haykin[54] for an introduction to ANNs and to Baeza-Yates and Ribeiro-Neto[55] and Cole[33] for a discussion of how this approach has been used in IR.

In this section, we have considered several classification methods, all of which require observations labeled with their true class as part of a training set. Other classification approaches can be used with text-based data. These include support vector machines[56,57] and logistic regression.[53] There are some excellent references available on classification, in general. The reader is referred to Duda et al.,[58] Bishop,[53] Hastie et al.,[45] and Ripley[52] for more information.

Document Clustering

Clustering is the process of organizing a set of data into groups such that observations—or documents—within a group are similar to each other and not to those in a different group. Clustering is also known as *unsupervised learning* because typically one does not have observations with known class labels.

With documents, we are interested in grouping them so those in a group are semantically similar. In the remainder of this section, we briefly mention several approaches that can be used for document clustering. Many clustering methods can be found in various communities, such as statistics, machine learning, data mining, and computer science.

In our opinion, *k*-means clustering is probably the method used most often and is suitable for high-dimensional document representations. Thus, one does not have to reduce the dimensionality before clustering. *K*-means partitions the data into *k* groups such that the within-group sum-of-squares is minimized. A basic method for *k*-means clustering is a two-step procedure: (1) assign each observation to its closest group (or cluster center) and (2) update the cluster centers using the assigned observations. These steps are repeated until there are no changes in cluster membership or the centers do not change.

Several issues should be noted about this method. First, the user must specify the number of clusters *k*, which can be problematic. Second, the results depend on an initial starting point. Thus, it is a good idea—especially with document clustering—to explore the groups in the data by varying the value for *k* and the initial starting point, then examining the resulting clusters for insights.

Other clustering methods exist and have been applied to document analysis. Some examples are hierarchical clustering,[40] model-based clustering (MBC),[59] spectral clustering,[60] probabilistic latent semantic indexing (PLSI),[61] and nonnegative matrix factorization (NMF).[62] The interested reader is referred to Manning and Schutze[8] for more information on clustering methods applied to documents or Everitt et al.[48] for clustering in general.

The agglomerative approach is used most often for hierarchical clustering, where each observation starts out in its own group. The two closest groups are merged at each step, until we have just one group. The resulting clusters depend on the distance that is used and the type of linkage, which determines how one measures the closeness of groups. Agglomerative clustering is suitable for high-dimensional data because it only requires the interpoint distances, so one does not have to reduce the dimensionality of the data before the analysis.

The MBC method assumes that the cluster structure can be modeled as a finite sum of weighted multivariate Gaussians. Each term in the finite mixture corresponds to a cluster. The clustering process then involves the estimation of the parameters in the mixture. Thus, it depends on the dimensionality of the data and is not feasible with very high dimensions. We illustrate the use of MBC for text analysis in the next section.

EXAMPLE USING STATISTICAL METHODS FOR THE ANALYSIS OF TEXT

Here, we provide a brief example of analyzing unstructured text data in survey records.[63] The data consist of accident narratives from Occupational Safety and Health Administration (OSHA) records.[64] We downloaded reports for November and December 2011 with the goal to cluster the records based on the abstract narratives. This would demonstrate that useful information might be found in the text fields.

We first pre-process the text by removing all special characters and converting to lower case. Non-informative words based on a general list are removed. The final data set contains 358 documents ($n = 358$), and the lexicon has 3841 words. We create a vector for each document (or observation) containing raw word frequencies.

We used MBC[59] to group the documents. The MBC approach estimates a finite mixture density function from the data, and each term in the mixture corresponds to a cluster. To get the clusters, we have to estimate the parameters of the finite mixture—the weights, *p*-dimensional means, and the $p \times p$ covariance matrices. MBC implicitly assumes that the data can be modeled as a finite mixture of Gaussians, and this should be tested. Looking at the scatterplot matrix in Figure 4, we see that the assumption in this case is a reasonable one. Some attractive aspects of MBC over other alternatives like *k*-means or agglomerative clustering are listed here: it provides an estimate of the number of groups; it does not require a distance or linkage, which can affect the result; and it allows for a flexible clustering structure (e.g., a mixture of spheroidal and ellipsoidal clusters).

The number of parameters to estimate in MBC is dependent on the dimensionality *p* or the number of words, in this case. Thus, we need to reduce the dimensionality before proceeding. So, the next step is to reduce the dimensionality from 3841 to look for common themes based on a particular clustering approach. We apply ISOMAP (isometric feature mapping)[65] to reduce the dimensionality using estimates of the geodesic distance between points as inputs to classical multidimensional scaling. It should be noted that reducing the dimensionality is not always required. For example, one does not need to reduce the dimensionality for *k*-means or agglomerative clustering.

Dimensionality is reduced to four based on a scree plot, and the 358 documents are now represented

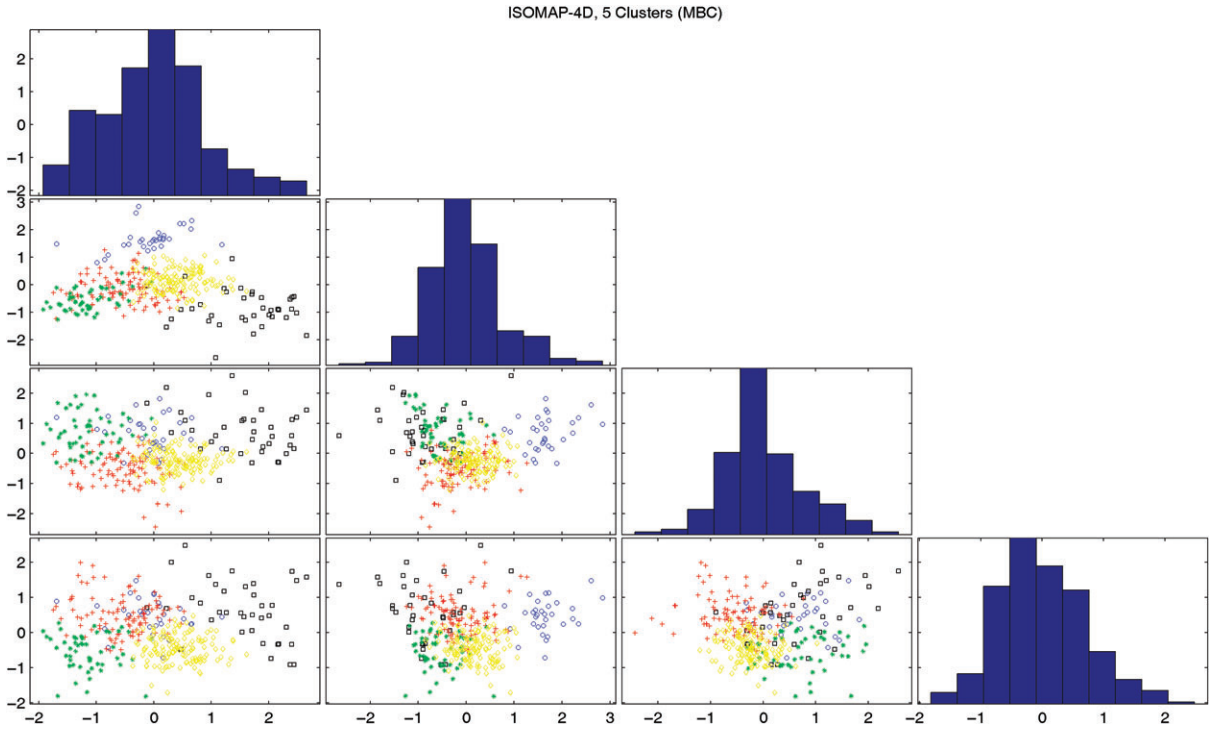


FIGURE 4 | This is a scatter plot matrix of the documents encoded in the four ISOMAP dimensions. The colors and symbols indicate cluster membership found using MBC.

in the first four ISOMAP dimensions. We use these transformed data as inputs to MBC.

The results of MBC indicate the presence of five groups; the next step is to determine whether the clusters have some common meaning or structure. We first look at a scatterplot matrix of the data with the groups indicated by different colors and plotting symbols. This is shown in Figure 4, where we see that the clusters seem reasonable and indicate the presence of some meaningful structures.

Next, we can try and determine topics by finding the high-frequency words in each cluster. Table 3 shows the ten highest frequency words in each of the five clusters. The red-highlighted words appear to characterize the groups. For example, cluster 1 has to do with vehicle accidents; group 2 involves cutting accidents; and clusters 3 and 4 comprise falls.

This is just a preliminary analysis of the text, and much more can be done. For example, we did not use any stemming or weights. It is interesting to note in Table 3 that the word *employee* occurs with the highest frequency in each cluster. This is an indication that we should revise our stop word list to include domain-specific words or use weights.

Later work with these records showed how to analyze the text for data editing and verification.[66] In particular, we used the accident narratives and their associated event code to build a classification tree. This classifier was used to examine misclassified records for editing and consistency. We also used the procedure to provide a better classification of miscoded records and to suggest specific codes originally classified as 'Other'.

APPLICATIONS

This section introduces some areas in computational linguistics where statisticians can contribute. These include MT, part-of-speech tagging, word sense disambiguation, and summarization.

Machine Translation

The pioneer of speech recognition work Frederick Jelinek is quoted as saying in 1988: 'Every time I fire a linguist, my translation improves'.[67] However, he states in 2004 that this statement was due to the lack of progress in automatic speech recognition and NLP using linguistic approaches in the 1970s and 1980s.[68] In the early years, linguists conducting research in MT tried to develop linguistic models that would allow a computer to understand a sentence in a source language and produce a sentence meaning the same thing in another language. The task was found to be very difficult because of different grammatical rules,

TABLE 3 | Highest Frequency Words in Clusters from MBC

Cluster 1, 31 Docs	Cluster 2, 37 Docs	Cluster 3, 102 Docs	Cluster 4, 51 Docs	Cluster 5, 137 Docs
employee	employee	employee	employee	employee
truck	machine	december	ladder	november
trailer	finger	approximately	approximately	december
approximately	saw	feet	roof	forklift
november	left	working	fell	approximately
driver	hand	fell	working	hospital
cable	approximately	november	employees	back
pole	blade	employer	november	right
december	december	tree	december	left
struck	cutting	accident	ft	working

dialects, and synonymy (equivalent meanings). Those impediments inspired researchers to change from a semantic-centered approach to one based on statistics.

Newer approaches arising in the 1980s were data-driven or corpus-based methods.[69] These benefited from the statistical techniques already used in speech recognition. One popular approach used an example-based technique that required databanks of parallel bilingual texts, where phrases in the databanks have been aligned by statistical or rule-based methods.[70]

Translation models received a push by Brown et al.[71] with the Fundamental Theorem of MT,

$$T^* = \arg\max_{T\in \mathrm{T}} p(T|S) = \arg\max_{T\in \mathrm{T}} p(S|T)\,p(T),$$

which decomposes the model into a language model $p(T)$ and a translation model $p(S|T)$. T is the target translation, and S is the source language. In recent research efforts, a log-linear approach is used and $p(T|S)$ is modeled directly as

$$P(T|S) = \frac{1}{\mathrm{Z}(S,T)} \exp \sum \lambda_i \phi_i (S,T),$$

where ϕ_i are a set of feature functions computed over the translation; Z is the normalizing constant; and λ_i are features, such as language model, phrase unigram probability, and model scores (see Brown, et al.[71]) These features are functions of the source and target sequences in the translation units and their context. State-of-the-art systems now employ millions of binary indicator features trained via machine learning methods.[70]

An alternative is the alignment template (AT) approach. This is the one used most often in phrase-based MT systems today and considered to be statistical MT systems. Another popular one, the syntax-based approach, is less dependent on statistical methods and more reliant on parsing. It could be implemented by parsing the source language text, or the target language text, or both texts. The parsers used are statistical parsers trained for each language. One advantage of the syntax-based systems is the capability of learning to reorder the words in preparation for the translation process. This can be achieved by collecting parse-tree fragments with respect to the alignments. However, error in the parsers and in the part-of-speech taggers brings additional burden to the whole process.[70]

MT in general and statistical MT in particular continues to be the subject of a large number of papers. Of special interest is finding new methods to bring more information into translation systems. For example, parsers and taggers (like name-entity taggers) for various languages are now available. Another area of interest being actively pursued is the creation of methods to automate the collection of translation models from parallel corpora with more robust parameter estimation.[70,71]

Part-of-Speech Tagging

We saw an example of ambiguity in *Formal Language Theory* section, which happens because the same word can be classified into different syntactic categories or parts of speech. For example, a word can be used as a noun or verb, as we see below with the word *fly*.

> Sometimes, I just want to *fly* away.
>
> That *fly* is bothering me.

Labeling words with their syntactic category is called *part-of-speech tagging* and is essential to parsing languages. The assignment of tags can be automated after a tagger has been trained with labeled corpora.[72]

We can use a tagged corpus to learn the probabilities of tag sequences. The probability of tag t^k coming after t^j is given by

$$P\left(t^k|t^j\right) = \frac{C\left(t^k, t^j\right)}{C\left(t^j\right)},$$

where $C(t^k, t^j)$ is the number of times t^j is followed by t^k, and $C(t^j)$ is the number of times tag t^j appears in the training corpus.

The best tagging $t_{1,n}$ for a sentence $s_{1.n}$ is the most-probable sequence:

$$\arg\max_{t_{1,n}} P\left(t_{1,n}|s_{1,n}\right) = \arg\max_{t_{1,n}} P\left(s_{1,n}|t_{1,n}\right) P\left(t_{1,n}\right).$$

Assuming words are independent and the identity of a word depends only on its tag, then we can estimate the probabilities from the training corpus as

$$P\left(s_{1,n}|t_{1,n}\right) P\left(t_{1,n}\right) = \prod_{i=1}^{n} P\left(s_i|t_i\right) P\left(t_i|t_{i-1}\right),$$

with $P(t_1|t_0) = 1$.

Scmid[73] proposed a probabilistic tagging method that estimates probabilities using decision trees. Ratnaparkhi[74] developed a statistical model based on maximum entropy that combines diverse contextual information, where distributional assumptions on the training data are unnecessary. Brants[75] describes another statistical approach that seems to work better than the entropy method. It is called Trigrams'n'Tags and is based on Markov models with mechanisms for handling unknown words.

Word Sense Disambiguation

Words can have different meanings (polysemy), which is a significant problem when analyzing text. For example, the word *company* has a different meaning in these sentences.[10]

> Captain Nolan leads a *company* of soldiers.
>
> We are looking forward to having *company* tonight.
>
> IBM is a large global *company*.

We could use part-of-speech tagging in the previous example (*Part-of-Speech Tagging*) to disambiguate the word *fly*; in one case, it is a noun and the other a verb. This is not the case with the three sentences above. The word *company* is a noun in all of them, so part-of-speech tagging will not help much.

The goal of word disambiguation is to decide which of the possible meanings should be attached to a word. Most of the methods for word sense disambiguation fall into three categories: supervised, unsupervised, and dictionary-based. The first two are described here.

Supervised disambiguation methods start with a corpus where any ambiguous word has a semantic label attached to it. Manning and Schutze[8] describe two approaches for learning. One is based on Bayesian classification[76] using word counts only. The second approach employs information theory and exploits one informative feature selected from many possibilities. Unsupervised disambiguation does not use a labeled corpus, so it can be employed in cases where there are no lexical resources, e.g., in online IR systems. Schutze[77] used a procedure called context-group discrimination, which is based on the EM method for clustering and discrimination.

Another interesting technique to discover word senses was proposed by Pantel and Lin.[78] Their method uses the immediate context of the word and assumes that words found in similar contexts have a similar sense. They employ a clustering-by-committee (CBC) algorithm to separate features that are most similar from those that are marginally similar. Each word is represented by a feature vector representing a context in which the word occurs.

Text Summarization

The goal of text summarization is to create a condensed version of a document or corpus. The summarization should include the important topics. Finding these meaningful portions of text is the main challenge in this application.

There are different types of summaries that have been addressed in computational linguistics, such as

- Informative summaries as a shortened version
- Topic-oriented summaries with respect to the reader
- Generic summaries from the author's point-of-view

Radev et al.[79] provide an overview, which serves as an introduction to a special issue of *Computational Linguistics* devoted to text summarization. At that time, research focused on systems to extract, merge, and edit phrases to produce a summary. A more-recent survey by Madnani and Dorr[80] discusses data-driven paraphrase generation methods.

Li et al.[81] present a framework based on conditional random fields to summarize reviews. Wong et al.[82] identify the important sentences by using

extrinsic aspects, content-conveying words, events, and relevance. They combine both supervised and unsupervised learning approaches.

Li et al.[83] describe an approach for event-based summarization that locates and organizes sentences in a summary according to the events described in the text. They derive the inter- and intra-event relevance using internal association, semantic relatedness, distributional similarity, and named-entity clustering. The inclusion of a sentence is determined by a PageRank[84] algorithm to determine its significance.

CONCLUSION

We have presented a brief (not broad enough, some may say) view of the area of computational linguistics and how the discipline intersects with statistics. The emphasis has been mainly on computational issues as these relate to NLP. This slanted view reflects the current interest in the computational linguistics community on NLP issues vice linguistic theoretical ones.

As we have illustrated in this paper, methods from probability and statistics are being used successfully in a wide range of applications, providing many areas of fruitful research for statisticians. Two of the main professional societies in computational linguistics are the Association for Computational Linguistics (ACL) and the Special Interest Group on IR (SIGIR), both of which have conference and journal venues. Research in engineering applications can be found in several IEEE publications and conferences, such as the International Conference on Acoustics, Speech and Signal Processing (ICASSP), or the International Conference on Data Mining (ICDM). There are the Neural Information Processing Systems Foundation (NIPS) and the Knowledge Discovery and Data Mining (KDD) conferences for those focusing on machine learning approaches. Papers discussing the analysis of text-based data are becoming more prevalent in mainstream statistical conferences such as the Joint Statistical Meetings (JSM) and the Conference on Statistical Practice (CSP). Readers can consult past conferences for these groups to get a sense of further topics and applications in computational linguistics.

We conclude this paper by providing some useful references and resources. There are several open-source software options. The R computing environment[85] has a NLP task view[86] providing information about modules for analyzing text, such as the *tm* package.[87] Python[88] is another open-source system that has add-ons for dealing with text data. Examples are the Natural Language Toolkit[89] that uses statistical methods and Gensim[90] for topic modeling. See the book by Bird[91] for NLP in Python. There is an Apache OpenNLP[92] library that has machine learning approaches for common computational linguistic tasks.

Jurafsky and Martin[7] wrote a good introductory book on computational linguistics. An excellent resource on NLP from a statistical viewpoint is by Manning and Schutze.[8] Abney[93] describes semisupervised learning approaches for computational linguistics, and Baayen[94] gives a statistical introduction to the analysis of text data using R. A handbook on NLP was edited by Indurkhya and Damerau.[70] Solka[95] provides a brief introduction to text mining from a statistical perspective, and Berry[96] surveys this topic focusing on the areas of clustering, classification, and IR.

Statisticians entering this field likely have text-based data that they want to analyze. However, it is always good to have other resources, and we provide two of them here. The Natural Language Toolkit mentioned above has a list of corpora that can be used for text analysis. The Linguistic Data Consortium[97] collects and distributes data for research in processing natural languages.

There are many more problems in computational linguistics where statisticians might contribute than what we describe in this paper. Our hope is that interested readers will consult past conference proceedings and journals for ideas and become a part of this fascinating research area.

REFERENCES

1. http://www.aclweb.org/ (Accessed May 2, 2015).
2. Association for Computational Linguistics. What is computational linguistics? Available at: http://www.aclweb.org/archive/misc/what.html (Accessed February 2005).
3. Computational Linguistics. *Wikipedia*. Available at: http://en.wikipedia.org/wiki/Computational_Linguistics (Accessed November 17, 2010).
4. Johnson M. How the statistical revolution changes (computational) linguistics. In: *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics*, Athens, Greece, March, 2009, 3–11.
5. Hearst MA. Untangling text data mining. In: *Proceedings of ACL '99: The 37th Annual Meeting of the Association for Computational Linguistics*, University of Maryland, June, 1999.

6. Maitra R. A statistical perspective on data mining. Available at: http://www.public.iastate.edu/~maitra/papers/datamining.pdf
7. Jurafsky D, Martin JH. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2nd ed. Upper Saddle River, NJ: Prentice Hall; 2008.
8. Manning CD, Schutze H. *Foundations of Statistical Natural Language Processing*. Cambridge, MA: The MIT Press; 2000.
9. Krahmer E. What computational linguists can learn from psychologists (and vice versa). *Comput Linguist* 2010, 36:285–294.
10. Martinez AR. Natural language processing. *WIREs Comput Stat* 2010, 2:352–357.
11. Frege G. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Halle a. S.: Louis Nebert; 1879. Translation: Concept Script, a formal language of pure thought modelled upon that of arithmetic, by S. Bauer-Mengelberg in Jean Van Heijenoort, ed. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*. Harvard University Press; 1967.
12. Partee B. Lexical semantics and compositionality. In: Gleitman L, Liberman M, eds. *Invitation to Cognitive Science, Part I: Language*. Cambridge, MA: The MIT Press; 1995.
13. Mitchell J. Composition in distributional models of semantics. PhD Dissertation, *School of Informatics, University of Edinburgh*, 2011.
14. Honeybone P, Firth JR. In: Chapman S, Routlede P, eds. *Key Thinkers in Linguistics and the Philosophy of Language*. Edinburgh: Edinburgh University Press; 2005, 80–86.
15. Harris ZS. Distributional structure. *Word* 1954, 10:146–162.
16. Landauer RK, Dumais ST. A solution to Plato's problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychol Rev* 1997, 104:211–240.
17. Salton G, Wong A, Yang CS. A vector space model for automatic indexing. *Commun ACM* 1975, 18:613–620.
18. Salton G. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Reading, MA: Addison-Wesley; 1989.
19. Landauer TK, Foltz PW, Lahan D. An introduction to latent semantic analysis. In: *Discourse Processes*, vol. 25. Mahwah, NJ: Lawrence Erlbaum Associates, Inc; 1998, 259–284.
20. Coccare N, Jurafsky D. Toward better integration of semantic predictors in statistical language modeling. In: *Proceedings of the 5th International Conference on Spoken Language Processing*, Sydney, Australia, 1998, 2403–2406.
21. Bellegarda JR. Exploiting latent semantic information in statistical language modeling. *Proc IEEE* 2000, 88:1276–1296.
22. Jessup ER, Martin JH. Taking a new look at the latent semantic analysis approach to information retrieval. In: Berry MW, ed. *Computational Information Retrieval*. Philadelphia, PA: SIAM; 2000, 121–144.
23. Washtell J. Compositional expectation: a purely distributional model of compositional semantics. In: *Proceedings of the Ninth International Conference on Computational Semantics*, Oxford, UK, 2011.
24. Griffiths TL, Steyvers MS, Tenenbaum JB. Topics in semantic representation. *Psychol Rev* 2007, 114:211–244.
25. Hofmann T. Probabilistic latent semantic indexing. In: *Proceedings of the 22nd International SIGIT Conference on Research and Development in Information Retrieval*, Berkeley, CA, 1999. Available at: http://cs.brown.edu/~th/papers/Hofmann-SIGIR99.pdf (Accessed December 2014).
26. Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. *J R Stat Soc B* 1977, 39:1–38.
27. Blei DM, Ng AY, Jordan MI. Latent Dirichlet allocation. *J Mach Learn Res* 2003, 3:993–1022.
28. Steyvers M, Griffiths TL. Probabilistic topic models. In: Landauer T, McNamara D, Dennis S, Kintsch W, eds. *Handbook of Latent Semantic Analysis*. New York, NY: Psychology Press; 2007. Available at: http://psiexp.ss.uci.edu/research/papers/SteyversGriffithsLSABookFormatted.pdf.
29. Anthes G. Topic models vs unstructured data. *Commun ACM* 2010, 53:16–18.
30. Ramage D, Dumais ST, Liebling DJ. Characterizing microblogs with topic models. In: *Proceedings of ICWSM*, Washington, DC, 2010. Available at: http://nlp.stanford.edu/pubs/twitter-icwsm10.pdf (Accessed December 2014).
31. Hopcroft JE, Ullman JD. *Introduction to Automata Theory*. Reading, MA: Addison-Wesley Publishing Company; 1979.
32. Revesz GE. *Introduction to Formal Languages*. New York: McGraw-Hill Book Company; 1983.
33. Cole R. *Survey of the State of the Art in Human Language Technology*. New York, NY: Cambridge University Press; 1997.
34. Lewis HR, Papadimitriou CH. *Elements of the Theory of Computation*. Upper Saddle River, NJ: Prentice Hall; 1981.
35. Charniak E. *Statistical Language Learning*. Cambridge, MA: The MIT Press; 1996.
36. Berry MW, Browne M. *Understanding Search Engines: Mathematical Modeling and Text Retrieval*. Philadelphia, PA: SIAM; 1999.

37. Dumais ST. Improving the retrieval of information from external sources. *Behav Res Methods Instrum Comput* 1991, 23:229–236.
38. Porter MF. Algorithm for suffix stripping. *Program* 2006, 40:211–218.
39. Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R. Indexing by latent semantic analysis. *J Am Soc Inf Sci* 1990, 41:391–407.
40. Hand D, Mannila H, Smyth P. *Principles of Data Mining*. Cambridge, MA: The MIT Press; 2001.
41. Berry MW, Drmac Z, Jessup ER. Matrices, vector spaces, and information retrieval. *SIAM Rev* 1999, 41:335–362.
42. Mosteller F, Wallace DL. Inference in an authorship problem: a comparative study of discrimination methods applied to the authorship of the disputed Federalist papers. *J Am Stat Assoc* 1963, 58:275–309.
43. Harish BS, Guru DS, Manjunath S. Representation and classification of text documents: a brief review. *Int J Comput Appl* 2010, RTIPPR:110–119.
44. Duda RO, Hart PE. *Pattern Classification and Scene Analysis*. New York: John Wiley & Sons; 1973.
45. Hastie T, Tibshirani R. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. New York, NY: Springer-Verlag; 2011.
46. Hotho A, Nurenberger A, Paab G. A brief survey of text mining. *LDV-GLDB J Comput Linguist Lang Technol* 2005, 20: 19–62.
47. Martinez AR. A framework for the representation of semantics. PhD Thesis, *George Mason University*, Fairfax, VA, 2002.
48. Everitt BS, Landau S, Leese M, Stahl D. *Cluster Analysis*. New York: John Wiley & Sons; 2011.
49. Webb A. *Statistical Pattern Recognition*. 2nd ed. Oxford: Oxford University Press; 2002.
50. Cover TM, Hart PE. Nearest neighbor pattern classification. *IEEE Trans Inf Theory* 1967, 13:21–27.
51. Breiman L. Random Forests. *Mach Learn* 2001, 45:5–32.
52. Ripley B. *Pattern Recognition and Neural Networks*. New York, NY: Cambridge University Press; 1996.
53. Bishop CM. *Pattern Recognition and Machine Learning*. New York, NY: Springer; 2006.
54. Haykin S. *Neural Networks: A Comprehensive Foundation*. New York: Macmillan College Publishing Company; 1994.
55. Baeza-Yates R, Ribeiro-Neto B. *Modern Information Retrieval*. New York: ACM Press; 1999.
56. Vapnik VN. *The Nature of Statistical Learning Theory*. New York, NY: Springer-Verlag; 1998.
57. Hearst MA. Support vector machines. *IEEE Intell Syst* 1998, July/August:18–28. Available at: http://pages.cs.wisc.edu/~jerryzhu/cs540/handouts/hearst98-SVMtutorial.pdf (Accessed December 2014).
58. Duda RO, Hart PE, Stork DG. *Pattern Classification*. 2nd ed. New York: John Wiley & Sons; 2001.
59. Fraley C, Raftery AE. How many clusters? Which clustering method? Answers via model-based cluster analysis. *Comput J* 1998, 41:578–588.
60. Ng AY, Jordan MI, Weiss Y. On spectral clustering: analysis and an algorithm. *Adv Neural Inf Process Syst* 2002, 14:849–856.
61. Hofmann R. Probabilistic latent semantic indexing. In: *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval*, Berkeley, CA, 15–19 August, 1999, 50–57.
62. Xu W, Liu G, Gong Y. Document clustering based on non-negative matrix factorization. In: *Proceedings of SIGIR'03*, Toronto, Canada, 2003, 267–273.
63. Martinez WL, Measure A. Statistical analysis of text in survey records. In: *Proceedings of the Federal Committee on Statistical Methodology Research Conference, FCSM*, Washington, DC, 2013. Available at: https://fcsm.sites.usa.gov/files/2014/05/C3_Martinez_2013FCSM.pdf (Accessed December 2014).
64. http://ogesdw.dol.gov/views/data_catalogs.php
65. Tenenbaum JB, de Silva V, Langford JC. A global geometric framework for nonlinear dimensionality reduction. *Science* 2000, 290:2319–2323.
66. Martinez WL. Text analysis tools for editing and verification. In: *UNECE Work Session on Statistical Data Editing*, Paris, France, April, 2014. Available at: http://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/ge.44/2014/mtg1/Topic_1_USA_Martinez.pdf (Accessed December 2014).
67. Kakaes K. Google, Yahoo! BabelFish use math principles to translate documents online. *The Washington Post*, February 21, 2011. Available at: http://www.washingtonpost.com/wp-dyn/content/article/2011/02/21/AR2011022102191.html
68. http://www.lrec-conf.org/lrec2004/doc/jelinek.pdf
69. Hutchins JW. Machine translation over fifty years. *Histoire Epistemologie Lang* 2001, 23:7–31.
70. Indurkhya N, Damerau FJ. *Handbook of Natural Language Processing*. 2nd ed. Boca Raton, FL: CRC Press; 2010.
71. Brown PF, Pietra VJD, Pietra SAD, Mercer RL. The mathematics of statistical machine translation: parameter estimation. *Comput Linguist* 1993, 19:263–311.
72. Martinez AR. Part-of-speech tagging. *WIREs Comput Stat* 2012, 4:107–113.
73. Scmid H. Probabilistic part-of-speech tagging using decision trees. In: *International Conference on New Methods in Language Processing,* Manchester, UK, 1994.
74. Ratnaparkhi A. A maximum entropy model for part-of-speech tagging. In: *Proceedings of the Conference on Empirical methods in Natural Language Processing (EMNLP)*, Philadelphia, PA, 1996.

75. Brants R. TnT—a statistical part-of-speech tagger. In: *Proceedings of the 6th Conference on Applied Natural Language Processing*, Seattle, WA, 2000.
76. Gale WA, Church KW, Yarowsky D. A method for disambiguating word senses in a corpus. *Comput Humanit* 1992, 26:415–439.
77. Schutze H. Automatic word sense discrimination. *Comput Linguist* 1998, 24:91–124.
78. Pantel P, Lin D. Discovering word senses from text. In: *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Edmonton, Canada, 2002, 613–619.
79. Radev DR, Hovy E, McKeown K. Introduction to the special issue on summarization. *Comput Linguist* 2002, 28:399–408.
80. Madnani N, Dorr BJ. Generating phrasal and sentential paraphrases: a survey of data-driven methods. *Comput Linguist* 2010, 36:341–387.
81. Li F, Han C, Huang M, Zhu X, Xia Y, Zhang S, Yu H. Structure-aware review mining and summarization. In: *Proceedings of the 23rd International Conference on Computational Linguistics*, Beijing, China, 2010, 653–661. Available at: http://aclweb.org/anthology/C10-1074 (Accessed December 2014).
82. Wong K, Wu M, Li W. Extractive summarization using supervised and semi-supervised learning. In: *Proceedings of the 22nd International Conference on Computational Linguistics*, Manchester, UK, 2008, 985–992. Available at: http://www.aclweb.org/anthology/C08-1124 (Accessed December 2014).
83. Li W, Wu M, Lu Q, Xu W, Yuan C. Extractive summarization using inter- and intra-event relevance. In: *Proceedings of the 21st International Conference on Computational Linguistics*, Sydney, Australia, 2006, 369–376.
84. Page L, Brin S, Motwani R, Winograd T. The PageRank citation ranking: bring order to the web. Technical Report, Stanford University, 1998.
85. http://cran.r-project.org/
86. http://cran.r-project.org/web/views/NaturalLanguageProcessing.html
87. Feinerer I, Hornik K, Meyer D. Text mining infrastructure in R. *J Stat Softw* 2008, 25:1–54. Available at: http://www.jstatsoft.org/v25/i05/ (Accessed December 2014).
88. https://www.python.org/ (Accessed December 2014).
89. http://www.nltk.org/
90. https://radimrehurek.com/gensim/
91. Bird S, Klein E, Loper E. *Natural Language Processing with Python*. Sebastopol, CA: O'Reilly Media; 2009.
92. http://opennlp.apache.org/index.html
93. Abney S. *Semisupervised Learning for Computational Linguistics*. Boca Raton, FL: Chapman & Hall/CRC; 2008.
94. Baayen RH. *Analyzing Linguistic Ata: A Practical Introduction to Statistics Using R*. New York: Cambridge University Press; 2008.
95. Solka J. Text data mining: theory and methods. *Stat Surv* 2008, 2:94–112. Available at: http://projecteuclid.org/euclid.ssu/1216238228 (Accessed December 2014).
96. Berry MW, ed. *Survey of Text Mining: Clustering, Classification, and Retrieval*. New York: Springer-Verlag; 2004.
97. https://www.ldc.upenn.edu/