A (PARTIAL) TAXONOMY OF MACHINE LEARNING FEATURES

Michael K. Bergman[1], Coralville, Iowa USA
November 23, 2015
AI3:::Adaptive Information blog

The two most labor-intensive steps in machine learning for natural language are: 1) feature engineering; and 2) labeling of training sets. Supervised machine learning uses these training sets where every point is an input-output pair, mapping an input, which is a feature, to an output, which is the label. The machine learning consists of inferring ("learning") a function that maps between these inputs and outputs with acceptable predictive power. This learned function can then be applied to previously unseen inputs in order to predict the output label. The technique is particularly suited to problems of regression or of classification.

It is not surprising that the two most labor-intensive tasks in machine learning involve determining the suitable inputs (features) and correctly labeling the output training labels. Elsewhere in this series I discuss training sets and labeling in detail. For this current article, we focus on *features*.

"Features" are perhaps the least discussed aspect of machine learning. References are made to how to select them; how to construct, extract or learn them; or how even to overall engineer them. But little guidance is provided as to what features exactly *are*. There really is no listing or inventory for what "features" might even be considered in the various aspects of natural language or text understanding. In part, I think, because we do not have this concrete feel for features, we also don't tend to understand how to maximize and systematize the role of features in support of our learning tasks. This aspect provides a compelling rationale for the advantages of properly constructed knowledge bases in support of artificial intelligence, what we have been terming as KBAI in this series.

So, before we can understand how to best leverage features in our KBAI efforts, we need to first define and name the feature space. That effort, in turn, enables us to provide a bit of an inventory for what features might contribute to natural language or knowledge base learning. We then organize that inventory a bit to point out the structural and conceptual relationships among these features, which enables us to provide a lightweight taxonomy for the space.

Since many of these features have not been named or exposed before, we conclude the article with some discussion about what next-generation learners may gain by working against this structure. Of course, since much of this thinking is incipient, there are certainly forks and deadends in what may unfold ahead, but there also will likely be unforeseen expansions and opportunities as well. A systematic view of machine learning in relation to knowledge and human language features — coupled with large-scale knowledge bases such as Wikipedia and Wikidata — can lead to faster and cheaper learners across a very broad range of NLP tasks [1].

What is a Feature?

A "feature is an individual measurable property of a phenomenon being observed" [2]. It is an input to a machine learner, an explanatory variable, sometimes in the form of a function. Features are sometimes equated to attributes, but this is not strictly true, since a feature may be a combination of other features, or a statistical

[1]Email: mike@mkbergman.com

calculation, or an abstraction of other inputs. In any case, a feature must be expressed as a numeric value (including Boolean) upon which the machine learner can calculate its predictions. Machine learner predictions of the output can only be based on these numeric features, though they can be subject to rules and weights depending on the type of learner.

The importance of features and the fact they may be extracted or constructed from other inputs is emphasized in this quote from Pedro Domingos [3]:

> "*At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used. . . . Often, the raw data is not in a form that is amenable to learning, but you can construct features from it that are. This is typically where most of the effort in a machine learning project goes. It is often also one of the most interesting parts, where intuition, creativity and 'black art' are as important as the technical stuff."*

Many experienced ML researchers make similar reference to the art or black art of features. In broad strokes, a feature may be a surface form, like terms or syntax or structure (such as hierarchy or connections); it may be derived (such as statistical, frequency, weighted or based on the ML model used); it may be semantic (in terms of meanings or relations); or it may be latent, as either something hidden or abstracted from feature layers below it. Unsupervised learning or deep learning features arise from the latent form.

For a given NLP problem domain, features can number into the millions or more. Concept classification, for example, could use features corresponding to all of the unique words or phrases in that domain. Relations between concepts could also be as numerous. To assign a value to such "high-dimensional" features, some form of vector relationship is calculated over, say, all of the terms in the space so that each term can be represented numerically [4]. Because learners may learn over multiple feature types, the potential combinations to be evaluated for the ML learner can literally be astronomical. This combinatorial problem has been known for decades, and has been termed the curse of dimensionality for more than 50 years [5].

Of course, just because a feature exists says nothing about whether it is a piece of information that might be useful for ML predictions or not. Features may thus prove to be one of four kinds: 1) strongly relevant; 2) weakly relevant; 3) irrelevant; or 4) redundant [6]. Strongly relevant features should always be considered; weakly relevant may sometimes be combined to improve the overall relevancy. All irrelevant or redundant features should be removed from consideration. Generally, the fewer the features the better, so long as the features used are strongly relevant and orthogonal (that is, they capture different aspects of the prediction space).

A (Partial) Inventory and Taxonomy of Natural Language and KB Features

To make this discussion more tangible, we have assembled a taxonomy of feature types in the context of natural language and knowledge bases. This inventory is drawn from the limited literature on feature engineering and selection in the context of KBAI from the perspectives of ML learning in general [7, 8, 9], ML learning ontologies [10, 11, 12] and knowledge bases [13, 14, 15, 16, 17]. This listing is only partial, but does provide an inventory of more than 200 feature types applicable to natural language.

We have organized this inventory into eight (8) main areas, shown in non-italicized **Bold**, which tend to cluster into these four groupings:

- ***Surface features*** — these are features that one can see within the source documents and knowledge bases. They include **Lexical** items for the terms and phrases in the domain corpus and knowledge base; **Syntactical** items that show the word order or syntax of the domain; **Structural** items that either split the documents and corpus into parts or reflect connections and organizations of the items, such as

hierarchies and graphs; or **Natural Language** items that reflect how the content is expressed in the surface forms of various human languages

- ***Derived features*** — are surface features that have been transformed or derived in some manner, such as the direct **Statistical** items or the **Model-based** ones reflecting the characteristics of the machine learners used
- ***Semantic features*** — these are summarized under the **Semantics** area, and reflect what the various items mean or how they are conceptually related to one another, and
- ***Latent features*** — these features are not observable from the source content. Rather, these are statistically derived abstractions of the features above that are one- to *N*-levels removed from the initial source features. These **Latent** items may either be individual features or entire layers of abstraction removed from the surface layer. These features result from applying unsupervised or deep learning machine learners.

Here is the taxonomy, again organized according to those same eight main areas:

Lexical		
	Corpus	
	Phrases	
		Averages
		Counts
		N-grams
		Weights
	Words	
		Averages
		Counts
		Cut-offs (top *N*)
		Dictionaries
		Named entities
		Stemming
		Stoplists
		Terms
		Weights
Syntactical		
	Anaphora	
	Cases	
	Complements (argument)	
	Co-references	
	Decorations	
	Dependency grammar	
		Head (linguistic)
	Distances	
	Gender	
	Moods	

- Paragraphs
- Parts of speech (POS)
- Patterns
- Plurality
- Phrases
- Sentences
- Tenses
- Word order

Statistical

- Articles
  - Vectors
- Information-theoretic
  - Entropy
  - Mutual information
- Meta-features
  - Correlations
  - Eigenvalues
  - Kurtosis
  - Sample measures
    - Accuracy
    - F-1
      - Precision
      - Relevance
  - Skewness
  - Vectors
  - Weights
- Phrases
  - Document frequencies
  - Frequencies (corpus)
  - Ranks
  - Vectors
- Words
  - Document frequencies
  - Frequencies (corpus)
  - Ranks
  - String similarity
  - Vectors
    - Cosine measures
    - Feature vectors

Structural

- Documents
  - Node types
    - Depth
    - Leaf
- Document parts
  - Abstract
  - Authors
  - Body
  - Captions
  - Dates
  - Headers
  - Images
  - Infoboxes
  - Links
  - Lists
  - Metadata
  - Templates
  - Title
  - Topics
- Captions
- Disambiguation pages
- Discussion pages
  - Authors
  - Body
  - Dates
  - Links
  - Topics
- Formats
- Graphs (and ontologies)
  - Acyclic
  - Concepts
    - Centrality
    - Relatedness
  - Directed
  - Metrics (counts, averages, min/max)
    - Attributes
    - Axioms
    - Children

- Classes
- Depth
- Individuals
- Parents
- Sub-graphs
- Headers
- Content
- Section hierarchy
- Infoboxes
- Attributes
- Missing attributes
- Missing values
- Templates
- Values
- Language versions
- Definitions
- Entities
- Labels
- Links
- Synsets
- Links
- Category
- Incoming
- Linked data
- Outgoing
- See also
- Lists
- Ordered
- Unordered
- Media
- Audio
- Images
- Video
- Metadata
- Authorship
- Dates
- Descriptions
- Formats
- Provenance

- Pagination
- Patterns
  - Dependency patterns
  - Surface patterns
    - Regular expressions
- Revisions
  - Authorship
  - Dates
  - Structure
    - Document parts
      - Captions
      - Headers
      - Infoboxes
      - Links
      - Lists
      - Metadata
      - Templates
      - Titles
  - Versions
- Source forms
  - Advertisements
  - Blog posts
  - Documents
    - Research articles
    - Technical documents
  - Emails
  - Microblogs (tweets)
  - News
  - Technical
  - Web pages
- Templates
- Titles
- Trees
  - Breadth measures
  - Counts
  - Depth measures
- Web pages
  - Advertisements
  - Body

		Footer		
		Header		
		Images		
		Lists		
		Menus		
		Metadata		
		Tables		
Semantics	[most also subject to **Syntactical** and **Statistical** features above)			
	Annotations			
		Alternative labels		
		Notes		
		Preferred labels		
	Associations			
		Association rules		
		Co-occurrences		
		See also		
	Attribute Types			
		Attributes		
			Cardinality	
			Descriptive	
			Qualifiers	
			Quantifiers	
				Many
		Values		
			Datatypes	
			Many	
	Categories			
		Eponymous pages		
	Concepts			
		Definitions		
		Grouped concepts (topics)		
		Hypernyms		
			Hypernym-based feature vectors	
		Hyponyms		
		Meanings		
		Synsets		
			Acronyms	
			Epithets	
			Jargon	

- Misspellings
- Nicknames
- Pseudonyms
- Redirects
- Synonyms

Entity Types

- Entities
- Events
- Locations

General semantic feature vectors

Relation Types

- Binary
- Identity
- Logical conjunctions
  - Conjunctive
  - Disjunctive
- Mereology (part of)
- Relations
  - Domain
  - Range
- Similarity

Roles

Voice

- Active/passive
- Gender
- Mood
- Sentiment
- Style
- Viewpoint (World view)

Natural Languages

Morphology

Nouns

Syntax

Verbs

Word order

Latent

Autoencoders

- Many; dependent on method

Features

		Many; dependent on method
	Hidden	
		Many; dependent on method
	Kernels	
		Many; dependent on method
Model-based		
	Decision tree	
		Tree measures
	Dimensionality	
	Feature characteristics	
		Datatypes
		Max
		Mean
		Min
		Number
		Outliers
		Standard deviation
	Functions	
		Factor graphs
		Functors
		Mappings
	Landmarking	
		Learner accuracy
	Method measures	
		Error rates

Table 1. A (Partial) Taxonomy of Machine Learning Features

This compilation exceeds any prior listings. In fact, most of the feature types shown have never been applied to NLP machine learning tasks. We now turn the discussion to why this is.

Mindset and Knowledge Bases

When one sees the breadth of impressive knowledge discovery tasks utilizing large-scale knowledge bases [18], exemplified by hundreds of research papers regarding NLP tasks utilizing Wikipedia [19], it is but a small stretch to envision a coherent knowledge base leveraging this content for the express purpose of making text-based machine learning systematic and less expensive. Expressed as an objective function, we now have clear guidance for how to re-organize and -express the source content information (Wikipedia, among others) to better support a ML learning factory. The idea of this and how it is driving Structured Dynamics' contracts and research is the *mindset*.

Rather than the singleton efforts to leverage knowledge bases for background knowledge, as has been the case to date, we can re-structure the knowledge source content underneath a coherent knowledge graph. This re-organization makes the entire knowledge structure computable and amenable to machine learning. It also enables the same learning capability to be turned upon itself (see image here), thereby working to improve the coverage and accuracy of the starting KB, all in a virtuous circle. Because of the mindset, we also can now look at the

native structure of the KBs and work to expose still more features, providing still further grist to the next generation ML learners. Fully 50% of the features listed in the inventory in *Table 1* above arise from these unique KB aspects, especially in the areas of **Semantics** and **Structural**, including graph relationships.

Many, if not most, of these new feature possibilities may prove redundant or only somewhat relevant. Not all features may ever prove useful, though some not generally used in many broader learners, such as case, may be effectively employed for named entity or specialty extractions, such as for copyrights or unique IDs or data types. Because many of these KB features cover quite orthogonal aspects of the source knowledge bases, the likelihood of finding new, strongly relevant features is high. Further, except for the **Latent** and **Model-based** areas, each of these feature types may be used singly or in combination to create coherent slices for both positive and negative training sets, helping to reduce the effort for labor-intensive labeling as well. By extension, these capabilities can also be applied to more effectively bootstrap the creation of gold standards, useful when parameters are being tested for the various machine learners.

Though the literature most often points to classification as the primary use of knowledge bases as background knowledge supporting machine learners, in fact many NLP tasks may leverage KBs. Here is but a brief listing of application areas for KBAI:

- Entity recognizers
- Relation extractors
- Classifiers
- Q & A systems
- Knowledge base mappings
- Ontology development
- Entity dictionaries
- Data conversion and mapping
- Master data management
- Specialty extractors

Table 2. NLP Applications for Machine Learners Using KBs

Surely other applications will emerge as this more systematic KBAI approach to machine learning evolves over the coming years.

Feature Engineering is an Essential Component

As noted, this richness of feature types leads to the combinatorial problem of too many features. Feature engineering is important both to help find the features of strongest relevance while reducing the feature space dimensionality in order to speed the ML learning times.

Initial feature engineering tasks should be to transform input data, regularize them if need be, and to create numeric vectors for new ones. These are basically preparation tasks to convert the source or target data to forms amenable to machine learning. This staging now enables us to discover the most relevant (“strong”) features for the given ML method under investigation.

In a KB context, specific learning tasks as outlined in *Table 2* are often highly patterned. The most effective features for training, say, an entity recognizer, will only involve a limited number of strongly relevant feature types. Moreover, the relevant feature types applicable to a given entity type should mostly apply to other entity types, even though the specific weights and individually important features will differ. This patterned aspect means that once a given ML learner is trained for a given entity type, its relevant feature types should be approximately applicable to other related entity types. The lengthy process of initial feature selection can be reduced as training proceeds for similar types. It appears that combinations of feature types, specific ML learners and methods to create training sets and gold standards may be discovered for entire classes of learning tasks. These combinations can be discovered, tested and repeated for new specific tasks within a given application cluster.

Probably the most time-consuming and demanding aspect of these patterned approaches resides in *feature selection* and *feature extraction*.

Feature selection is the process of finding a subset of the available feature types that provide the highest predictive value while not overfitting [20]. Feature selection is typically split into three main approaches [6, 21, 22]:

- *Filter* — select the *N* most promising features based on a ranking from some form of proxy measure, like mutual information or the Pearson correlation coefficient, which provides a measure of the information gain from using a given feature type
- *Wrapper* — wherein feature subsets are tested through a greedy search heuristic that either starts with an empty set and adds features (forward selection) keeping the "strongest" ones, or starts with a full set and gradually removes the "weakest" ones (backward selection); the wrapper approach may be computationally expensive, or
- *Embedded* — wherein feature selection is a part of model construction.

For high-dimensional features, such as terms and term vectors, one may apply stoplists or cut-offs (only considering the top *N* most frequent terms, for example) to reduce dimensionality. Part of the "art" portion resides in knowing which feature candidates may warrant formal selection or not; this learning can be codified and reused for similar applications. Extractions and some unsupervised learning tests may also be applied at this point in order to discover additional "strong" features.

Feature extraction transforms the data in the high-dimensional space to a space of fewer dimensions. Functions create new features in the form of **Latent** variables, which are not directly observable. Also, because these are statistically derived values, many input features are reduced to the synthetic measure, which naturally causes a reduction in dimensionality. Advantages from a reduction in dimensionality include:

1. Often a better feature set (resulting in better predictions) [23]
2. Faster computation and smaller storage
3. Reduction in collinearity due to reduction in weakly interacting inputs
4. Easier graphing and visualization.

On the other hand, the latent features are abstractions, and so not easily understood as the literal.

In deep learning, multiple layers of these latent features are generated as the system learns. But latent passes may also be combined with observable features, which is one way that evaluations of what a document means can be applied across multiple input forms of the content.

Of course, it is also possible to combine the predictions from multiple ML methods, which then also raises the questions of ensemble scoring. Surely we will also see these more systematic approaches to machine learning themselves be subject to self-learning (that is, metalearning), such that the overall learning process can proceed in a more automated way.

Considerations for a Feature Science

In supervised learning, it is clear that more time and attention has been given to the labeling of the data, what the desired output of the model should be. Much less time and attention has been devoted to features, the input side of the equation. As a result, much needs to be done. The purposeful use of knowledge bases and structuring them properly is one of the ways progress will be made.

But progress also requires some answers to some basic questions. A scientific approach to the feature space would likely need to consider, among other objectives:

- Full understanding of surface, derived and latent features
- Relating various use cases and problems to specific machine learners and classes of learners

- Relating specific machine learners to the usefulness of particular features (see also hyperparameter optimization and model selection)
- Improved methods for feature engineering and construction
- Improved methods for feature selection
- A better understanding of how to select and match supervised and unsupervised ML.

Some tools and utilities would also help to promote this progress. Some of these capabilities include:

- Feature inventories — how to create and document taxonomies of feature types
- Feature generation — methods for codification of leading recipes
- Feature transformations — the same for transformations, up to and including vector creation
- Feature validation — ways to test feature sets in standard ways.

Role of a Platform

The object of these efforts is to systematize how knowledge bases, combined with machine learners, can speed the deployment and lower the cost of creating tailored artificial intelligence applications of natural language for specific domains. This installment in our KBAI series has focused on the role and importance of *features*. There is an abundance of opportunity in this area, and an abundance of work required, but little systematization.

The good news is that platforms are possible that can build, manage, and grow the knowledge bases and knowledge graphs supporting machine learning. Machine learners can be applied in a pipeline manner to these KBs, including orchestrating the data flows in generating and testing features, running and testing learners, creating positive and negative training sets, and establishing gold standards. The heart of the platform must be an appropriately structured knowledge base organized according to a coherent knowledge graph; this is the present focus of Structured Dynamics' efforts.

In the real world, engagements always demand unique scope and unique use cases. Platforms should be engineered that enable ready access, extensions, configurations, and learners. It is important to structure the KBs such that slices and modules can be specified, and all surface attributes may be selected and queried. Mapping to external schema is also essential. Background knowledge from a coherent knowledge base is the way to fuel this.

Acknowledgements

This article was originally posted on the *AI3::Adaptive Information* Web site at http://www.mkbergman.com/1905/a-partial-taxonomy-of-machine-learning-features/. This version has been edited and reformatted slightly for PDF distribution. We thank Cognonto Corporation for making this content freely available.

---

[1] Features apply to any form of machine learning, including for things like image, speech and pattern recognition. However, this article is limited to the context of natural language, unstructured data and knowledge bases.

[2] See the feature entry from Wikipedia, which itself is based upon Christoper Bishop, 2006. *Pattern Recognition and Machine Learning*. Berlin: Springer. ISBN 0-387-31073-8.

[3] Pedro Domingos, 2012. "A Few Useful Things to Know About Machine Learning," *Communications of the ACM* 55, no. 10 (2012): 78-87.

[4] For example, in the term or phrase space, the vectors might be constructed from counts, frequencies, cosine relationships between representative documents, distance functions between terms, etc.

[5] Richard Ernest Bellman, 1957. *Dynamic Programming*, Rand Corporation, Princeton University Press, ISBN 978-0-691-07951-6, as republished as Richard Ernest Bellman, 2003. *Dynamic Programming*, Courier Dover Publications, ISBN 978-0-486-42809-3.

[6] Isabell Guyon and André Elisseeff, 2006. "An Introduction to Feature Extraction," in Guyon, Isabelle, Steve Gunn, Masoud Nikravesh, and Lofti A. Zadeh, eds. *Feature Extraction: Foundations and Applications*, pp. 1-25. Springer Berlin Heidelberg, 2006.

[7] Haussler, David, 1999. *Convolution Kernels on Discrete Structures*, Vol. 646. Technical Report UCSC-CRL-99-10, Department of Computer Science, University of California at Santa Cruz, 38 pp., July 8, 1999.

[8] Reif, Matthias, Faisal Shafait, Markus Goldstein, Thomas Breuel, and Andreas Dengel, 2014. "Automatic Classifier Selection for Non-experts," *Pattern Analysis and Applications* 17, no. 1 (2014): 83-96.

[9] Tang, Jiliang, Salem Alelyani, and Huan Liu, 2014. "Feature Selection for Classification: A Review." *Data Classification: Algorithms and Applications* (2014): 37

[10] Melanie Hilario, Phong Nguyen, Huyen Do, Adam Woznica, and Alexandros Kalousis, 2011 "Ontology-based Meta-mining of Knowledge Discovery Workflows," in *Meta-Learning in Computational Intelligence*, pp. 273-315. Springer Berlin Heidelberg, 2011.

[11] Panče Panov, Larisa Soldatova, and Sašo Džeroski, 2014. "Ontology of Core Data Mining Entities," *Data Mining and Knowledge Discovery* 28, no. 5-6 (2014): 1222-1265.

[12] See the general KBAI category entries on M.K. Bergman, *AI3:::Adaptive Information* blog, various dates.

[13] Ivo Anastacio, Bruno Martins and Pavel Calado, 2011. "Supervised Learning for Linking Named Entities to Knowledge Base Entries," in *Proceedings of the Text Analysis Conference (TAC2011)*.

[14] Weiwei Cheng, Gjergji Kasneci, Thore Graepel, David Stern, and Ralf Herbrich, 2011. "Automated Feature Generation from Structured Knowledge," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pp. 1395-1404. ACM, 2011.

[15] Lan Huang, David Milne, Eibe Frank, and Ian H. Witten, 2012. "Learning a Concept-based Document Similarity Measure." *Journal of the American Society for Information Science and Technology* 63, no. 8 (2012): 1593-1608.

[16] Olena Medelyan, Catherine Legg, David Milne and Ian H. Witten, 2008. *Mining Meaning from Wikipedia*, Working Paper Series ISSN 1177-777X, Department of Computer Science, The University of Waikato (New Zealand), September 2008, 82 pp.

[17] Hui Shen, Mika Chen, Razvan Bunescu, and Rada Mihalcea, 2012. "Wikipedia Taxonomic Relation Extraction using Wikipedia Distant Supervision," *Ann Arbor* 1001: 48109.

[18] Conventional knowledge bases have also been supplemented with massive-scale statistical bases, most often created from major search engine indexes; see the section on 'Statistical Corpora' in M.K. Bergman, 2014. "Knowledge-based Artificial Intelligence," *AI3:::Adaptive Information* blog, November 14, 2014.

[19] See M.K. Bergman, "SWEETpedia," listing of Wikipedia research articles, on *AI3:::Adaptive Information* blog, January 25, 2010. The listing as of its last update included 246 articles; also, see Wikipedia's own "Wikipedia in Academic Studies."

[20] Overfitting is where a statistical model, such as a machine learner, describes random error or noise instead of the underlying relationship. It is particularly a problem in high-dimensional spaces, a common outcome of employing too many features.

[21] George H. John, Ron Kohavi, and Karl Pfleger, 1994. "Irrelevant features and the subset selection problem." In *Machine Learning: Proceedings of the Eleventh International Conference*, pp. 121-129. 1994.

[22] See especially slide #11 in Zdeněk Žabokrtský, 2015. "Feature Engineering in Machine Learning," *Machine Learning Methods* course, Institute of Formal and Applied Linguistics, Charles University in Prague, Czech Republic.

[23] If constructed properly, deep learning models can be effective feature extractors over high-dimensional data; see Geoffrey E. Hinton, 2009. "Deep Belief Networks," *Scholarpedia* 4 (5): 5947, which references an earlier paper, Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation* 18, no. 7 (2006): 1527-1554.