Available Article

Author's final: This draft is prior to submission for publication, and the subsequent edits in the published version. If quoting or citing, please refer to the proper citation of the published version below to check accuracy and pagination.

Cite as: Bergman, M. K. Appendix C: KBpedia Feature Possibilities. in *A Knowledge Representation Practionary: Guidelines Based on Charles Sanders Peirce* (ed. Bergman, M. K.) 421-434 (Springer International Publishing, 2018). doi:10.1007/978-3-319-98092-8_20

Official site: https://link.springer.com/book/10.1007/978-3-319-98092-8

Full-text: http://www.mkbergman.com/publications/akrp/appendix-c.pdf

Abstract: The two most labor-intensive steps in machine learning for natural language are 1) feature engineering, and 2) labeling of training sets. A systematic view of machine learning relating 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 comprehensive range of NLP tasks. Machine learners can only predict output based on numeric features, to which we must convert text or other representation, though they can be subject to rules and weights depending on the type of learner. For natural language, 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. I provide an organized inventory of more than 200 feature types applicable to natural language. They include lexical, syntactical, structural and other items that reflect how we express the content in the surface forms of various human languages.

APPENDIX C:

KBPEDIA FEATURE POSSIBILITIES

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 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 adequate predictive power. We can apply this learned function to previously unseen inputs to predict the output label. The technique is particularly suited to problems of regression or of classification. Yet, despite the integral role of *features* in the machine learning process, we often overlook their importance compared to labels and algorithms.

Before we can understand how best to leverage features in our *knowledge-based artificial intelligence* (KBAI) efforts, we need first to define and name the feature space. Separately, we also need to study what exists on how to select, construct, extract or engineer these features. Armed with this background, we can now assemble an inventory of what features might contribute to natural language or knowledge base learning.

We have followed these steps to produce a listing of possible KBpedia features.[1] We have organized 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 others have not named or exposed many of these features before, we conclude this appendix with some discussion about what next-generation learners may gain by working against this structure. Of course, since much of this thinking is incipient, forks and dead ends may unfold, but there also will likely be unforeseen expansions and opportunities as well. A systematic view of machine learning and its 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 comprehensive range of NLP tasks.

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. Some equate features with attributes, but this is not strictly accurate, since a feature may be a combination of other features, or a statistical calculation, or an abstraction of other inputs (some would say it could be about anything!). In any case, we must express a feature as a numeric value (including Boolean as 0 and 1) 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.

Pedro Domingos emphasizes the importance of features and the fact they may be extracted or constructed from other inputs:[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 a similar reference to the art or black art of features. In broad strokes in the context of natural language, a feature may be: a surface form, like terms or syntax or structure (such as hierarchy or connections); derived (such as statistical, frequency, weighted or based on the ML model used); semantic (in terms of meanings or relations); or latent, either as 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. We calculate some form of vector relationship over, say, all of the terms in the space so that we may assign a numerical value to 'high-dimensional' features.[4] Because learners may learn about multiple feature types, the potential combinations for the ML learner can 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 useful for ML predictions. Features may thus be one of four kinds: 1) strongly relevant, 2) weakly relevant, 3) irrelevant, or 4) redundant.[6] We should favor strongly relevant features; we may sometimes combine weakly relevant to improve the overall relevancy. We should remove all irrelevant or redundant features from consideration. Often, 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) to one another.

A (Partial) Inventory of Natural Language and KB Features

To make this discussion tangible, we have assembled a taxonomy of feature types in the context of natural language and knowledge bases. I drew this inventory 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] and knowledge bases.[12] [13] [14] [15] [16] 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 in *Table C-1*, shown in *italics*, 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 we express the content in the surface forms of various human languages;
- Derived features — are surface features that we transform or derive 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 in 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. Instead, 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.

We may nucleate features and training sets based on the syntax, morphology, semantics (meaning of the data) or relationships (connections) of the source data in the knowledge base. Continuous testing and the application of machine learning to the system itself creates virtuous feedback where the accuracy of the overall system is constantly and incrementally improved.

The compiled taxonomy listing of features in *Table C-1* exceeds any prior listings. In fact, most of the feature types we show have yet to participate in NLP machine learning tasks. We organize our taxonomy according to the same eight main areas, shown under the shaded entries, noted above:

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 (Worldview)	
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 C-1: A (Partial) Taxonomy of Machine Learning Features

Fully 50% of the features listed in the inventory in *Table C-1* above arise from 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 or perhaps not at all. Not all features may ever prove useful. Some, such as *Case*, may be effectively employed for named entity or specialty extractions, applicable to copyrights or unique IDs or data types, but may prove of little use in other areas.

Still, because many of these KB features cover 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, we can use these capabilities to more effectively bootstrap the creation of gold standards, useful when we are testing parameters.

The *Statistical* and *Meta-features* sections of *Table C-1* are first derivatives of the base structure. The few listed here are examples of how we may include such measures in the feature pool, and they all are common ones. The point is that we may use derivatives and embeddings from other features in the table as legitimate features in their own right.

Though the literature most often points to classification as the primary use of knowledge bases as background knowledge supporting machine learners, in fact, many natural language processing (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 C-2: NLP Applications for Machine Learners Using KBs

See also *Table 4-1*. Undoubtedly other applications will emerge as this more systematic KBAI approach to machine learning evolves over the coming years.

Feature Engineering for Practical Limits

This richness of feature types leads to the combinatorial problem of too many features. Feature engineering is the way both to help find the features of strongest relevance while reducing the feature space dimensionality 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 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 C-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 individual features (attributes and other type relations) will differ. This patterned aspect means that once we train a given ML learner for a given entity type, its relevant feature types should be approximately applicable to other related entity types. We can reduce the lengthy process of initial feature selection as training proceeds for similar types. It appears we may discover combinations of feature types, specific ML learners, and methods to create training sets and gold standards for entire classes of learning tasks.

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.[17] Researchers typically split feature selection into three main approaches:[6] [18] [19]

- *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* — test feature subsets 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* — include feature selection as a part of model construction.

For high-dimensional features, such as terms and term vectors, we 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. One may also apply some unsupervised learning tests at this point 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 of a reduction in dimensionality include:

1. Often a better feature set (resulting in better predictions);[20]
2. Faster computation and smaller storage;
3. Reduction in collinearity due to a reduction in weakly interacting inputs; and
4. Easier graphing and visualization.

On the other hand, the latent features are abstractions, and so not easily understood as the literal. Deep learning generates multiple layers of these latent features as the system learns.

Of course, we may also combine the predictions from multiple ML methods, which then also raises the questions of ensemble scoring. We may also self-learn (that is, meta-learn) more systematic approaches to ML 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 have 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. The purposeful use of knowledge bases and structuring them is one way we can make progress. Still, progress also requires some answers to some fundamental 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; and
- A better understanding of how to select 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 the codification of leading recipes; and
- Feature transformations — the same for transformations, up to and including vector creation.

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 bespoke artificial intelligence applications of natural language for specific domains. KBAI places primary importance on *features*. An abundance of opportunity exists in this area, and an abundance of work required, but little systematization.

The good news is we can build platforms that manage and grow the knowledge bases and knowledge graphs supporting machine learning, as we discussed in *Parts III* and *IV*. We can apply machine learners 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 primary purpose of KBpedia.

Still, in the real world, engagements always demand unique scope and unique use cases. We should engineer our platforms to enable ready access, extensions, configurations, and learners. It is vital to structure our source knowledge bases such that slices and modules can be specified, and all surface attributes may be selected and queried. Mapping to the external schema is also essential. Background knowledge from a coherent knowledge base is the most efficient way to fuel this.

Appendix Notes

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. Bishop, C., *Pattern Recognition and Machine Learning*, Berlin: Springer, 2006.
3. Domingos, P., "A Few Useful Things to Know About Machine Learning.," *Communications of the ACM*, vol. 55, 2012, pp. 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. Bellman, R. E., *Dynamic Programming*, Rand Corporation, Princeton University Press, 1957.
6. Guyon, I., and Elisseeff, A., "An Introduction to Feature Extraction," *Feature extraction*, 2006, pp. 1–25.
7. Haussler, D., *Convolution Kernels on Discrete Structures*, Technical report, Department of Computer Science, University of California at Santa Cruz, 1999.
8. Reif, M., Shafait, F., Goldstein, M., Breuel, T., and Dengel, A., "Automatic Classifier Selection for Non-Experts," *Pattern Analysis and Applications*, vol. 17, 2014, pp. 83–96.
9. Tang, J., Alelyani, S., and Liu, and H., "Feature Selection for Classification: A Review.," *Data Classification: Algorithms and Applications*, 2014, p. 37.
10. Hilario, M., Nguyen, P., Do, H., Woznica, A., and Kalousis, and A., "Ontology-based Meta-mining of Knowledge Discovery Workflows," *Meta-Learning in Computational Intelligence*, Heidelberg: Springer Berlin, 2011, pp. 273–315.
11. Panov, P., Soldatova, L., and Džeroski, S., "Ontology of Core Data Mining Entities," *Data Mining and Knowledge Discovery*, vol. 28, Sep. 2014, pp. 1222–1265.
12. Anastacio, I., Martins, B., and Calado, P., "Supervised Learning for Linking Named Entities to Knowledge Base Entries," *Proceedings of the Text Analysis Conference (TAC2011)*, 2011.
13. Cheng, W., Kasneci, G., Graepel, T., Stern, D., and Herbrich, R., "Automated Feature Generation from Structured Knowledge," *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, ACM, 2011, pp. 1395–1404.
14. Huang, L., Milne, D., Frank, E., and Witten, I. H., "Learning a Concept-Based Document Similarity Measure," *Journal of the Association for Information Science and Technology*, vol. 63, 2012, pp. 1593–1608.
15. Medelyan, O., Milne, D., Legg, C., and Witten, I. H., "Mining Meaning from Wikipedia," *International Journal of Human-Computer Studies*, vol. 67, 2009, pp. 716–754.
16. Shen, H., Chen, M., Bunescu, R., and Mihalcea, R., "Wikipedia Taxonomic Relation Extraction using Wikipedia Distant Supervision," *Ann Arbor*, vol. 1001, p. 48109.
17. 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.
18. John, G. H., Kohavi, R., and Pfleger, and K., "Irrelevant Features and the Subset Selection Problem," *Machine Learning: Proceedings of the Eleventh International Conference*, 1994, pp. 121–129.
19. Žabokrtský, Z., "Feature Engineering in Machine Learning," 2015.
20. Hinton, G. E., "Deep Belief Networks," *Scholarpedia*, vol. 4, 2009, p. 5947.