Computation, Information, and Intelligence (ENGRI/CS/INFO/COGST 172), Spring 2007
2/19/07: **Lecture 13 aid — Information retrieval and indexing basics**

Topics: information retrieval (IR) based on term presence/absence; indices; B-trees

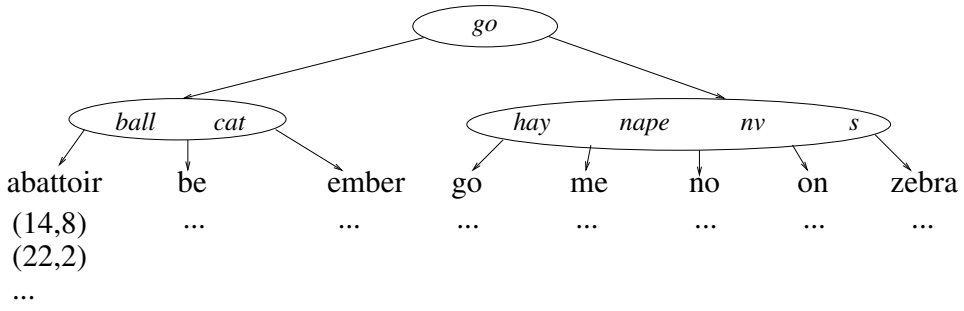
Reminders: Homework Two is due on Wednesday. Remember that although you are free/encouraged to use geometric arguments for the question dealing with the perceptron learning algorithm, you must give explicit numerical computations as well.

I. Standard IR setting We have a *corpus* D consisting of n documents and a *vocabulary* W consisting of m distinct terms. The user expresses their information need via a *query* q.

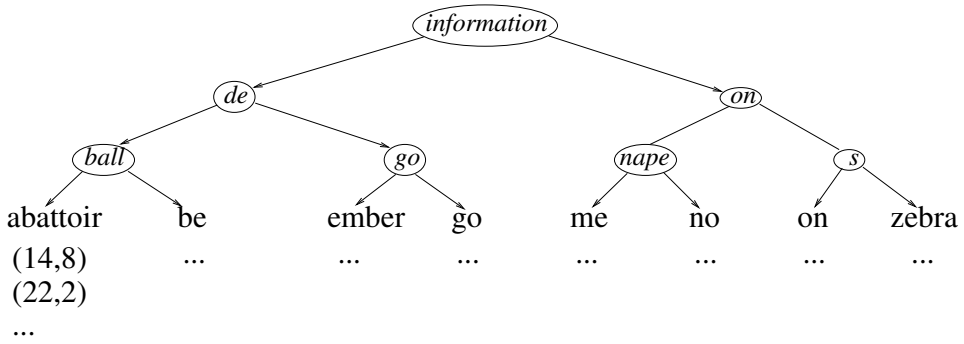
II. Index We can build an *index* that contains all the vocabulary items in sorted order and that indicates, for each term w in W, at least the following information:

- Those documents that contain w, and
- The location(s) of w in each such document.

III. Example B-trees Here is a B-tree (of order 2).



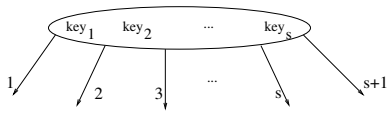
This B-tree (of order 1) is for the same index as the previous B-tree.



(OVER)

IV. B-trees: the formalities (a.k.a. *balanced multiway search trees*) Conceptually, you can think of a B-tree as sitting "on top" of an index, with each leaf corresponding to the information for some single term in the index.[1]

Every B-tree has some *order* (or *minimization factor*) t such that except for the root, each internal node contains between t and $2t$ (inclusive) *keys* in sorted order and has one more child than keys; schematically, this looks like the following, where s is some number such that $t \leq s \leq 2t$:



(The root is an exception; it can have between 1 and $2t$ keys.) Different internal nodes need not contain the same number of keys. Every leaf is required to have the same depth.

The keys give information about the leaves at the bottom of the $s + 1$ *subtrees* "beneath" the node. The i^{th} child "covers" (perhaps indirectly) terms w such that $\text{key}_{i-1} \preceq w \prec \text{key}_i$. The exceptions are the first child, which "covers" terms w such that $w \prec \text{key}_1$, and the $s + 1$th child, which "covers" terms w such that $\text{key}_s \preceq w$.

The depth of an order-t B-tree is at most roughly $\log_t(m)$.

[1] Strictly speaking, when the leaves are data items we have a B^+*-tree* rather than a B-tree, but we won't make this distinction.