2. Relevancy Search

a. Stop Word/Stemming is lossy - keep a backup of the original text as it helps in advanced queries (TT combined)

b. Length of the query field impacts the ES score given to that field matches

   i. Custom boosting field matches
   ii. Using max or other operations should be thought clearly

c. TF - IDF are scaled in ES score (Typically square root)

d. We also learned that scores between fields aren't at all comparable. They exist entirely in their own scoring universes. Each field's TF-IDF are calculated within the universe of that field

e. For search, given all the fudge factors in Lucene scoring and the peculiarities of field statistics, you should never attempt to compare scores between queries without a great deal of deep customization to make them comparable.

f. Luckily, Elasticsearch gives us the ability to disable TF as needed. - TODO Search more - TODO: why, when, and how?

g. Why are you taking a maximum? Why is this strategy used by default? Are there other strategies you could use to combine these scores so that it's not all or nothing between strong title and strong overview matches? You may have solved the Space Jam problem, but what will this max do to other searches? When other searches create different scores, will we be back at square one?

h. Finally, you must ask whether your fieldWeight calculation could be improved. Field Norm can be disabled in the index to avoid bias towards shorter queries

i. Do we truly care about fieldNorms ? Is it important in this use case to bias toward shorter text or longer text? There's also the ever present struggle of the relevance engineer: do the terms themselves represent the right features latent in the text?

j. By using asymmetric analysis techniques, you can encode a notion of specificity into the search application. **Asymmetric analysis**means that the analysis applied at query time is different from the analysis applied at index time. (Just for the term). There is only one token during query time, but indexing synonyms will expand index to contain both tokens.

k. An additive boost stacks the boost on top of the base query. To be effective, the boost must layer on just enough oomph to matter in the final calculation. A 0.01 boost added to a base query score of 4 will hardly matter. A 100.0 boost added to that base query will effectively override the base query.
A multiplicative boosts scales the base query. A simple boost multiplier of 1.2 ensures, regardless of how the base query's score behaves, that boosted documents will get 20% more oomph than unboosted documents.

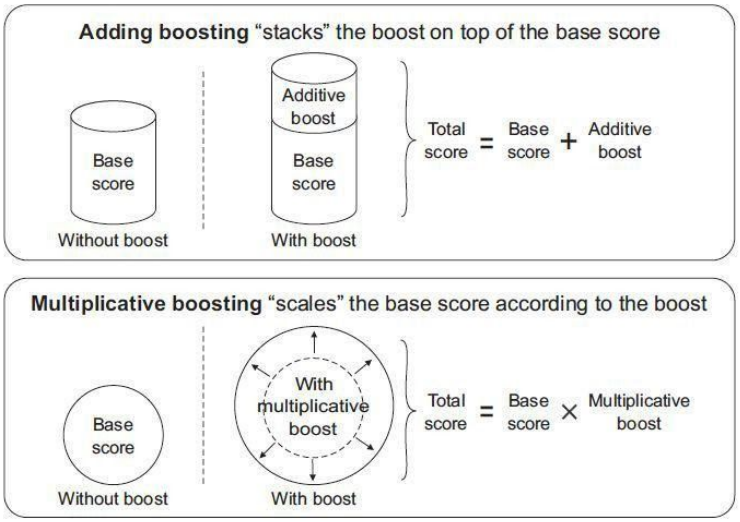

Figure 7.2 Additive boosting layers on extra criteria; multiplicative boosting inflates/ deflates through multiplication.

l. You also need to concern yourself with the coordinating factor (coord)—that strong bias toward documents that match all the clauses (here, both the boost and base queries). This can be disabled with the disable_coord option if you don't want this bias. You might want to do this if the boost query is more "extra credit" than "required."
m. Recall, TF Å~ IDF has a strong bias toward shorter fields through field normalization
n. Simply tweaking a boost weight may postpone a problem for another search.
o. Step away from TF - IDF to get to yes/no scoring.
p. You need to get inside users' heads, modeling their expectations by using math.
q. When scoring doesn't matter, and you only want to show or not show a set of search results, filtering can be a simpler solution.
r. Instead of disabling tokenization, you'll use a technique referred to as sentinel tokens . Sentinel tokens represent important boundary features in the text (perhaps a sentence, paragraph, or begin/end point).
s. In some cases, you need a way to ignore the TF × IDF scoring. To do this, you can wrap your query in a constant_score query. This lets you hardcode the resulting query's score to a constant (the boost value).

t. elasticsearch's field_value_factor function lets you take a field's value and use it directly. Optionally, you can apply a few simple modifiers and functions.
u. Whatever the case may be, if the search application modifies the search on behalf of the user, make sure to inform the user about how the search has been changed so that the user won't become disoriented with results that don't match expectations.
v. Prevent overfitting your search by stepping back and asking yourself, "Is it worth it?"
w. Despite holding the title relevance engineer, you likely don't know what users deem relevant
x. Good feedback goes beyond data. It's based on a broader, collaborative organization that can interpret both search correctness and user behavioral data together. Only then can data be a factor in providing you good information on how users perceive search. Instead of a data-driven culture being the starting point, it's in fact the most mature form of a collaborative, user-focused culture.
y. An expert user was once a member of the user population and now works at the organization. If you're building search for a gardening site that caters to professional horticulturalists, then perhaps marketing, QA, or other groups in your organization staff horticulturalists. Expert users are a gold mine for search feedback. They can directly simulate the appropriate negative reaction that users will experience with irrelevant search. Yet unlike most users, they're deeply invested in your success. They're much more likely to give you detailed, constructive feedback.
z. Therefore, why not take the thrashing search terms and add them to the concept field for the document that finally satisfies the user's information needs? This way, the next time someone follows the same path as our thrashing user, they will more likely find what they need in their first set of search results.
aa. Collocation extraction is a common technique for identifying statistically significant phrases. The text of the documents is split into n-grams (commonly bigrams). Then statistical analysis is used to determine which n-grams occur frequently enough to be considered statistically significant.

bb.

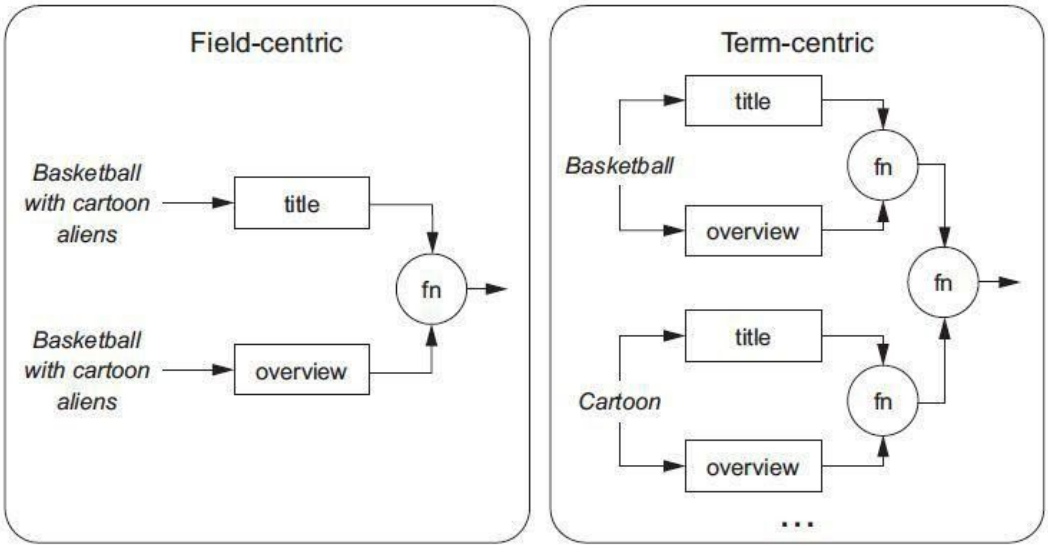

Figure 5.4 Field-centric versus term-centric search showing the fields `title` and `overview`. Field-centric searches each field in isolation; term-centric searches each field term by term.

cc.

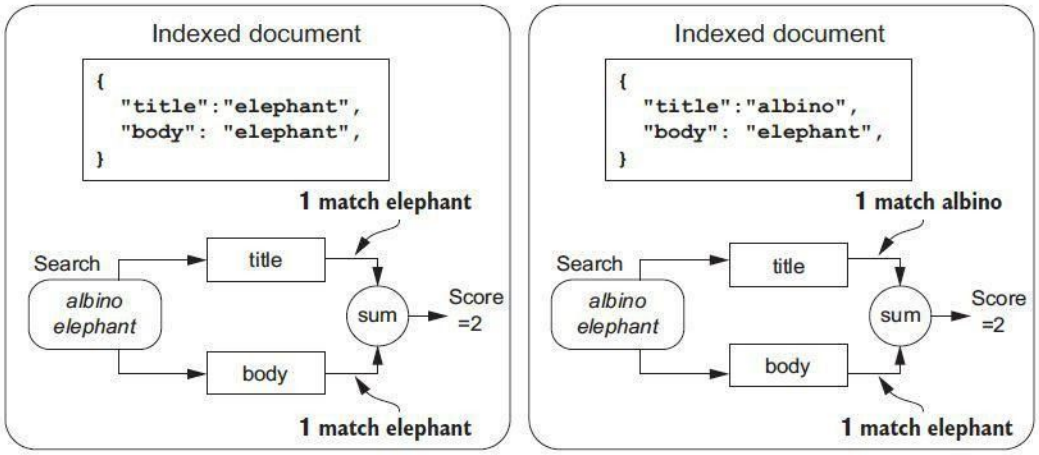

Figure 6.3 Field-centric search can fail to account for cases in which multiple search terms match.

dd. Second, and perhaps more important, TF × IDF scoring is biased heavily against what the searcher is likely searching for. Remember, TF × IDF scores bias heavily toward rare terms (IDF correlates with rareness). But the user is more likely to be searching for a mundane, commonplace item. If you go to the grocery store and ask for coffee, you're more likely to be happy with being brought to the coffee aisle, where coffee is plentiful. You wouldn't be happy if brought to the ice-cream aisle, where a few tubs of coffee ice cream await you.

ee. The boost is just a multiple, chosen through experimentation, to ensure the desired lopsidedness. If you searched a catalog of academic articles, would you

focus on whether matches occurred in specific sections of an article? Would you prioritize a match in an article's introduction section over a match in its conclusion section, for example? What if instead of being broken up this way, the documents were subdivided into searchable fields by page (page 1, page 2, and so forth)?

How would you prioritize each page? The point is that your database could arbitrarily break up article text any number of ways! But are those subdivisions appropriate for search? Do they map to your users' mental models of the content?

ff. Signal discordance is the disconnect between the signals generated from fine-grained, specific fields present in a search engine (as derived from a source data model) and a user's far more general mental model of the content.

gg. So in reality, a term-centric query-parser enforces identical search terms for each field. We call this property field synchronicity, the capability to query multiple fields with identical search terms—or put a different way, the restriction that they must be searched in the same way.

hh. In contrast, a cross_fields search is dynamic, addressing signal discordance at query time. It does this by becoming a dismax-style query parser on steroids. The ranking function of cross_fields remains identical to the query parser approach, with one important modification: the cross_fields query temporarily modifies the search term's document frequency, field by field, before searching. Much like the dismax query parser, cross_fields continues to suffer from field synchronicity. If fields don't share the same analyzer, cross_fields returns an error.

ii. By having one wide-net base score and carefully selected discriminating amplifiers, you're more likely to arrive at a place that at the very least satisfies the user.

jj.