Final Prolog Project

I decided to use the data of more than one article. While loading the data I run into problems because of tags like 'W103' that represent different words in different articles. I tried to append the corresponding article tag to every predicate while asserting but I could not figure out how to do it. Since I did not want to lose too much time I wrote a Python program (get_prolog_data.py) that appends the article to every predicate and writes all the data into one file (prolog_data.pl). I used 20 random articles.

Get Subject, Object, Verb or Passive Phrase

I played around with the data on chunks and dependent phrases and decided to write several predicates that return the grammatical functions of a sentence, such as subject or object. **full_svo\8** gives the full phrases for Subject, Verb and Object. Subj, Verb, Obj are variables for the keyword of each phrase (head). Fullsbj, Fullverb, Fullobj represent the whole phrase. I included the search by keyword to enable more general queries like full_svo(Subj,'causes','growth',Sent,Art,Fullsbj,Fullverb,Fullobj) that allow to search for any kind of growth. For full phrases only use **action\3**.

```
?- full_svo(Subj, Verb, Obj, Sent, Art, Fullsbj, Fullverb, Fullobj).
Subj = variant,
Verb = Fullverb, Fullverb = causes,
Obj = growth,
Sent = 'S15',
Art = 'Art3',
Fullsbj = 'the B variant',
Fullobj = 'an accelerated-rate growth' ;
```

The predicates **full_subject_verb\6**, **full_verb_object\6** and **full_verb_agens\6** work very similarly. The corresponding phrase for every keyword is computed by **depfull\7** and other predicates that compute the longest Noun Group or Verb Group for a keyword in a given sentence. **subj_verb\4** and **prep_obj\4** return only the keyword, not the full phrases.
I implemented a predicate that searches for passive constructions. passive\8 uses the data from **full_verb_agens\6**, **full_verb_subj\6** and **prep_obj\4** to find sentences written in passive voice.

```
?- passive(Theme,Verb,Agens,Sent,Art,Fulltheme,Fullverb,Fullagens).
Theme = strain,
Verb = obtained,
Agens = lab,
Sent = 'S41',
Art = 'Art1',
Fulltheme = 'The mutant strain',
Fullverb = 'was obtained',
Fullagens = 'the lab'
```

Using these predicates you can find relations between A and B, no matter if they are formulated in active or passive voice. **activates_full\4** gives full phrase solutions for the question: What activates what? Using the keyword search (**act\2**) one can even implement a predicate that looks recursively for activation relations. If A activates B and B activates C, then A activates C (**activates\2**). Unfortunately, in my data no indirect relations were found. I wrote similar predicates (without indirect relations) for other verbs: **constructs\2**, **binds\2**, **elevates\2**, **cleaves\2** and **amplifies\2**. All of these predicates can by accessed by operators.
For example: ?- What amplifies 'COX16'.
The more general predicate **affects\3** allows us to use any verb. Example: ?- (X,Y,'obtain').

TF-IDF

After all these linguistic features, I decided to do something more mathematical. I wrote a predicate that calculates the TF-IDF score, which represents the importance of a word in a document. **most_rep_word\2** returns the lemma with the highest TF-IDF score for a specific article. **printMostRepWords\2** prints the N most representative words for an article.

The score **tfidf\3** is computed step by step with predicates **tf\3**, **df\3** and **idf\2**. A sorted list of all scores and lemmas is given by **tfidfList\2**. Many predicates concerning the number of articles and the number of words in sentences and articles were used to compute the TF-IDF score. They seem very self-explanatory so I will not explain them in detail.

Study focus and Results

I wanted to use the Article Section tags that come with every sentence. **introduction\2** and **results\2** return a list of all sentences that are tagged with 'Introduction' or 'Results', 'Discussion' or 'Results and discussion'. **print_study_focus\1** writes all sentences from the introduction of a given article that contain the word 'study' (as in 'In this study, we will analyse...'). **print_results_focus\1** does the same for the keyword 'shown' (as in 'We have shown that...'). Of course, these keywords can and should be expanded to gain a more advanced summaries for every article.

Compounds and Nominal Conjunctions

Unfortunately, compounds bound by a hyphen are split while parsing so they end up in different Noun Groups. To extend the phrases with their full compounds I created **compound\2**. Example:

```
?- compound('B',Compound).
Compound = 'SVH-B'.
```

Nominal conjunctions are also not together in the same Noun Group. **nom_conj\4** finds noun phrases bound by a connector. I tried to use these predicates to extend the phrases but I did not find the time to finish it.

Play Arounds

is_of\2 returns constructions that contain a prepositional phrase with the preposition ‘of’. An operator allows queries such as:

```
?- 'the lab' is_of X.
X = 'Prof . K . K' ;
X = 'Prof . Grossman' ;
```

is_a\2 returns a 'definition' as found in sentences with 'to be'. Sentences with negations are not considered.

```
?- 'CDKN1A' is_a X.
X = 'a well-known downstream target gene'.
```

is_protein\1 tells us if something is a protein or not.

```
?- 'Spo0A' is_protein.
true.
```

This predicate is based on the 'PROT' tag in the term predicate of the input data. It seems to be very vague. Most of the PROTs are only protein-related and not real proteins themselves, for example 'cell death'. That's why this predicate is not very reliable and should not be used.

I will not discuss the helping functions since they are self-explanatory.

Examples

I prepared another file (write_examples.pl) that contains a predicate **writeExamples\0**. It creates a file example_results.txt and writes examples for the TF-IDF score, the relations of A and B and an example for an introductional phrase of an article.