SemanticSky

Taking the network up to heaven.

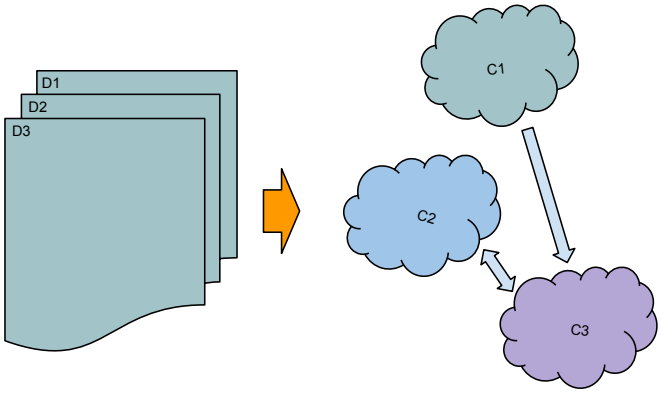
A project report by Pietro Pasotti and Sander Latour.

Contents

Part I
SemanticSky.

1 Making the case for SemanticSky.

(as the previous project has showed,) There are many ways to go if the target is to classify documents by similarity. More precisely, what we need is an algorithm for deriving from a set of documents a graph whose nodes are documents (or documents descriptors, as we shall see) and whose edges represent a similarity relation between pairs of documents.



From the literature, we know that there are several algorithms that can do this, each one of them basing its decision on a particular descriptor/feature of the documents.

As the reader might know, each document has multiple features, and (unless we use some sort of feature selection) each one of them is a plausible candidate for assessing the pairwise similarity of the documents. For example, a document can be described by enumerating the words it is made of; the characters it is made of, but also the capitalized words it contains, its line-wise length, the number and target of hyperlinks or the shape of the coffee stains on its author's desk.

Now suppose we have many different algorithms, each one of which takes into account just one of these features. It is plausible that each algorithm, basing its decisions on different evidence (or maybe just reasoning from it in a different way) will output a different decision for any given pair of documents, or most of them. Then, statistically, if their decisions are

diverse enough, it has been proved[1] that combining their suggestions results in a much more accurate prediction than any of their individual outputs, or than a single algorithm that takes all of the features into account at once to compute its decision.

Furthermore, depending on the kind of algorithm we are considering, its precision will correlate in some way with the contents of the documents we are having it evaluate: documents with a lot of words and no image are more likely to be correctly classified by a word-based algorithm than by an algorithm that only considers attached .jpg images. Thus, we can consider each algorithm as a local expert: in a more or less restricted region of the space of inputs, each algorithm is capable of predictions which, beside being more likely to get it right than random guess[2], are also more likely to be correct than the suggestions of all other algorithms.

Once we have identified these local experts, we may want to combine their suggestions in order to take into account, for each decision we are trying to make, just or mainly the experts in the field the decision is taking place. In other words, we want to train the system in a way such that it will learn (for we don't know it in advance) which algorithm is better at what, and such that having learned this, it will merge the outputs of all the algorithms in a way to maximise the likelihood of having a correct decision given what the system has learned up to that moment.

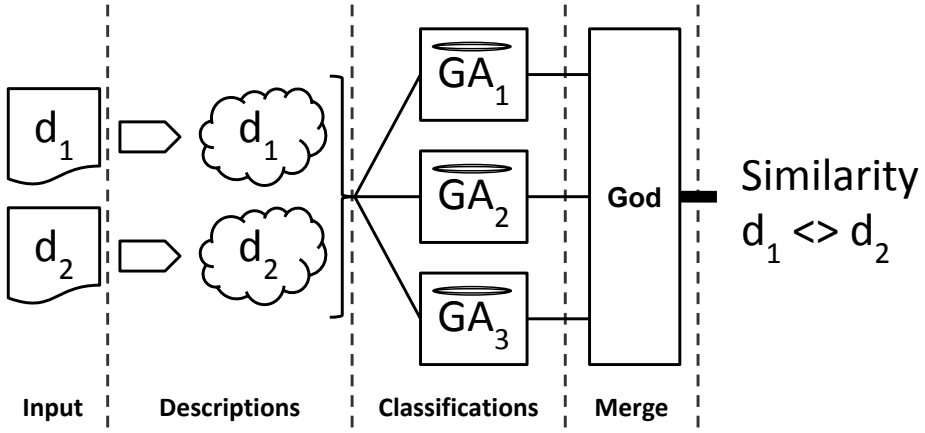
Suppose then that we did not have any intuition or clue about which algorithm is better at what. Then the only way to discover the expertises of the algorithms is to give them feedback: the system has to learn on-line, or be trained in some way, so as that its future decisions are based on the past evidence. In a real-world application of SemanticSky, the feedback can plausibly be expected to come from the users of such a classification system. For the purposes of this project however we came up with a training mechanism to simulate human input, which will be explained in the appropriate section.

In order to manage the feedback mechanism, and to merge the outputs of the various algorithms into a single one, the system needs a supervisor algorithm. Since we were speaking of clouds and skies, it seemed natural to go on with the metaphor and have picture ourselves a round table of guardian angels suggesting their opinions to God himself.

This is then the pipeline of SemanticSky:

[1] For literature: [Ethem Alpaydin 2010 ch.17].

[2] Which is a necessary requirement for the algorithms in general.



We can consider the network of clouds to be an internal representation of the set of documents, which the system uses as basis for its judgements. The set of judgements, on the other hand, can be considered as the belief state of God (or of SemanticSky as a whole). Each document is represented by a cloud, and the similarity of clouds (that hopefully mirrors the similarity of the underlying documents) is assessed by angels which then contribute with different weights to God's decisions.

Through feedback, the system is able to modify on the fly the weights, thereby modifying the edges of the network of clouds, as if it was adapting his image of the world (his hypothesis, or posteriors) to best fit the evidence we give to it.

2 Clouds: the bricks.

At the moment, clouds are more or less simple wrappers for documents. The central data structure for a cloud is made of what we will call **layers**. What makes it a simple wrapper is the fact that only one layer, the topmost, is currently being filled; but the main idea behind a layered cloud is that it should include hierarchically ordered information: from the most precious (and most hardly one-to-one matchable) information to the second-hand, less polished one.

To show the reason behind this few things would do better than an example: suppose we have a document called 'Rise and Fall of Ziggy Stardust', containing a couple of pages of history of this man called Ziggy. Unless the title is ironic or in any way misleading, which, assume, is not the case, we

already have a few important informations: that this document is about a man called Ziggy Stardust and that a rise and a fall are somehow involved.

SemanticSky currently performs little or no semantic analysis (even though it was originally intended to), so the cloud will not know that it is Ziggy rising and falling, but just that the names 'Ziggy', 'rise', 'fall' are relevant descriptors of the document. And the mere fact of the title being a title, supposing we know how to identify it, tells us that the information stored there is important. This is possible, of course, only because the input data is partly annotated: it's not an uniform body of plain text, but already contains some extra information which we can use. In absence of this, other methods could be used to extract headers, titles and other kind of sources of usually relevant information.

This is what layers are meant to be for: the innermost layer will certainly contain these four words, plus the most relevant others that, with diverse heuristics, we will be able to extract from the body of the document itself.

Currently, the clouds store just about everything in a single layer as the information we have is all first-hand: it comes directly from the document, which is the most trustworthy source of information we have.

The sorts of information a cloud contains in its only layer were extracted in various ways, but many of them were available because the data we used to develop the first application of SemanticSky was partially annotated. Thus we had tags and a few other important metadata associated with documents, such as title, author, and such. This is what the main layer contained:

- names: all Capitalized Sequences Of Words.
- urls: all urls either hidden in hrefs (if the documents' original text is html) or explicitly mentioned in the text.
- words: information about their frequency (tf) *and* their frequency weighted by their inverse document frequency (tf-idf).
- core: a special subset of words which (e.g. due to annotations) we have reasons to believe more important than the other ones; for example those which come from titles or headlines.
- tags: a list of the tags assigned to the item.

2.1 Future work.

The framework can clearly be expanded already at this level, and in our opinion this will be most certainly one of the most promising directions to

go. The layer-based structure is already there, but is not currently used. An immediate expansion of SemanticSky could involve filling the lower-level layers, and finding an appropriate way to use them.

Possible sources for the information to fill these layers with include:

1. The web. The second layer could be filled up, for example, with information drawn from google querying for 'The Rise and Fall of Ziggy Stardust' or some other keywords.

2. Other clouds. At some stage, suppose, the system will settle on the decision that Ziggy's cloud is clearly related with (say) David Bowie's cloud. Then, once the confidence about this fact is higher than a certain threshold, we might let some keywords of either clouds 'filter' into the lower layers of the other cloud.[3]

3. Corpora information. From a corpus search we might discover that 'stardust' is very frequently related with Carl Sagan and stars.

The latter example about stars and Carl Sagan is a clear situation where we want to make sure that this information is taken as second-hand only and is given appropriately less weight than the first-hand one.

3 Agents.

Agents in SemanticSky are basically pipes for human input. In a non-experimental application, they are meant to be associated with accounts of people who have write access to the network of documents, but at a more general level, they are 'whatever human source of information we have', which is used mainly to give feedback to the algorithms and to influence the network themselves, for example by rating similarity of item-pairs, relevance of tags to items, modifying the items, suggesting (removal of) new links or tags.

Ideally, whenever the system will detect that the consensus about some issue is reached, feedback will trigger and backpropagate to all algorithms (and maybe even agents) that had something to say about it, punishing them for their bad suggestions or rewarding them for the good ones.

[3]This might have the side effect of forming loops of self-reinforcing feedbacks, so we will have to make sure that algorithms, when re-evaluating the two clouds, won't take into account *that* information as well.

The most immediate effect of the feedback is to influence (up or down) the trustworthiness of an agent, be it algorithmic or human in nature. Trustworthiness is not absolute, however, but relative (or 'contextual') to some domain or 'area of expertise': a person (an algorithm) can be very good on items regarding people but very bad at spotting connections between glossaries and tags. So, we want to learn his expertises and hold as valuable his suggestions in that field, while (proportionately) ignoring his suggestions on topics which we know he is not keen on.

3.1 Future work.

Detecting when a discussion is 'settled' is not an obvious thing. One could use various 'hotness' metrics or functions of the number of people participating to the discussion, the length of their posts and the time gap between them, or more simply the time since the last update of the content of a page. My guess is that some experimental work is needed to find the optimal technique to evaluate when to trigger the feedback: we don't want to increase or decrease trustworthiness of people when the voting is still open (and changes of mind still possible).

Secondly, at the moment feedback is irreversible. This can and should be questioned sometime in the future.

4 Guardian Angels.

Guardian Angels are currently the core of the algorithmic part of SemanticSky, and the interaction of them, the Agents and God is probably the theoretically most interesting thing going on there.

Basically, a Guardian is nothing but an agent that only examines pairs of items when prompted (by God) and that takes decisions based on a never changing algorithm.

Their strength is of a collective kind: each of them takes decisions based on a rather small part of all the evidence available, and produces a prediction whose only requirement is to be accurate above chance.

Follows a short description of all the algorithms currently implemented.

tf and tf_idf_weighting The most standard Guardian Angels, just retrieve the tf / tf-idf value for each word in the bags of words of the documents (also held in the clouds) and returns the cosine of the angle of the two vectorized documents.

naive_name_comparison Based on the 'names' section of the layers of the clouds, which were previously extracted via regex matching, this algorithm just checks how many names the two clouds share, and normalizes it against the set union of the two lists of names. The required kind of matching is very strict: a 'Ziggy Stardust' in a cloud won't match a 'Ziggy, S.' in the other. This could be improved in many ways.

Plus, a name is currently Whatever Is Capitalized, and this is not really so. We are lucky that Starfish does not have pages in German.

Given the little information the algorithm can work on, the 600 nonzero results against 210000 is a reasonable output. Interestingly enough, the algorithm is one of the most precise ones, with a 15% chance that if he detects something, there is actually something.

extended_name_comparison This algorithm, as all other algorithms containing the word 'extended', is based on the social network principle that if A's friends are B's friends, then A and B are likely to be friends themselves.

Let $names(\phi)$ be the set of names contained in the zero layer of the cloud ϕ. Then, for $\phi \in A, B$, we define

$$names(\phi) = \phi.layers[0]['names']$$

$$Neighbours(\phi) = \{a | a \in sky\ if\ names(\phi) \cap names(a) \neq \varnothing\}$$

Finally, we make an extended bag of words by summing ϕ's names with the cores of all of ϕ's neighbours: that is, the most relevant words for all of the clouds which share at least a name with ϕ. Then, we count the overlaps and normalize against the union of the sets.

Not as precise as naive_name_comparison and much more computationally heavy, but gives nonzero results more often.

naive_core_overlap The same as naive_name_comparison, but instead of the 'names' entry of the clouds' layers, takes the core, performing a simple $len(intersection)/len(union)$ computation.

extended_core_overlap let $C(\psi)$ denote the core of (the zero layer of) some cloud ψ. This time, unlike the extended_name_overlap algorithm, the words to extend $C(\psi)$ are going to be drawn not directly from other clouds, but from a global repository of the co-occurrence counts of words in sentences based on the whole corpus.

So, the globally most frequently co-occurring pairs, not weighted by idf (which could of course be an interesting extension), go to extend $C(\psi)$ if at least one of the words in the co-occurrence pair is in the core of ψ.

So, the extended core of psi $E(C(\psi))$ will be, where $coo(a, b) \iff a$ and b co-occur more than twice in the whole corpus, and W is the set of all words occurring in the corpus:

$$E(C(\psi)) = w | w \in W \wedge (w \in C(\psi) \vee \exists b \in C(\psi) : coo(w, b))$$

Finally, the two extended cores are compared in the usual intersection / union way.

tag_similarity_naive This algorithm, as well as the following ones, are made specifically for evaluating the similarity of tag clouds: clouds that wrap tag-type items. This algorithm was made mainly for experimental reasons. Tag clouds can be obtained by filling in the layers with the tag's glossary and derived information, when available.

However the training data we used did not have any rating of the similarity of tags, so we could not assess the evaluation of these two algorithms.

This algorithm, given two clouds, just counts how many pages are tagged with both tags (that is, the tags that the clouds are built around) and normalizes it against the length of the union.

tag_similarity_extended This algorithm is an instance of a higher-level Guardian Angel, or Meta Angel: to work, it needs some previous evaluation. This algorithm, also, is mentioned to evaluate the similarity of two tags, given the clouds they are tagged with.

Basically, it measures the overlap of the sets of clouds marked with the two tags and returns an averaged confidence of the relations of these links.

Suppose we have the following situation, where a, b are tags (i.e. strings such as 'LearningAnalytics','DavidBowie'):

$a \rightarrow$ [cloud1, cloud2]; that is: both cloud1 and cloud2 are tagged with 'a'.

$b \rightarrow$ [cloud1, cloud3]

$c \rightarrow$ [cloud4,cloud5]

Clearly, similarity(cloud1,cloud1) = 1, so tag a and b will be at least 0.5 related since they share half of their clouds; more if cloud2 and cloud3 are related in gods' beliefs. Then suppose cloud1 and cloud4 are believed to be

related by 0.4, and cloud5, cloud2, are related by 0.6, but cloud5 and cloud3 are totally unrelated. Then, tag c will be much closer to tag a than to tag b.

The algorithm just computes over this intuition.

tag_overlap This algorithm, unlike the two previous ones, is a measure of the distance of two nontag-type clouds, and simply measures the intersection/union of the tags the two clouds are labeled with.

coo_dicts_overlap This algorithm has two versions, plus a third one which consists simply in taking the best result from v1 and v2.

v1 This version returns a pair-to-pair correspondence check of co-occurring pairs of words in the two clouds. Such correspondence is very valuable and gives some good results, but overall rare.

v2 This version is more permissive (returns many more nonzero values) and consists in splitting the pairs thereby using single words as term of comparison. (Basically it performs a bag of words overlap/union, building the bags from the co-occurrence dictionary). Implements Grefenstette (1994) algorithm for computing the values.

coo_dicts_neighbour Taken the co-occurring pairs in cloud ϕ, denoted $coo(\phi)$, we split them and augment this bag of words with the most frequent words that often co-occur with them at the whole corpus level. (Similar to extended_core_overlap), but based on a different starting set. Then compares the two extended bags.

coo_dicts_extended_neighbour This algorithm is very complex, takes ages to run and is so permissive that it captures far more noise than signal. Not even worth an explanation.

4.1 Future work.

- Guardian Angels do work, but they also are currently very stupid. Each one of them just performs a little task in a probably naive way: this could be improved. On the other hand, the main idea behind ensembles is precisely to have a large number of very stupid algorithms that collectively

produce a smart decision. Be it their small number or their being too stupid, in the testing application we tried this did not happen so much. To face this situation, we could either increment their number (boosting is an option, or even bagging if we had a larger dataset) or make them smarter.

- Plus, the more and most diverse the angels' decision algorithm are, the higher the overall performance of the system will be. Currently, as the reader will have noticed, there are bunches of three-to-four algorithms that perform very similar computations (and mostly work on the same evidence). Thus, one immediate way to extend the system will be to implement more guardians, or to add variations to the already existing ones.
- As noticed above, also, the decisions of the algorithms which are currently implemented are based on a rather small subset of all the available information. Some algorithm only takes into account the words' frequency, some other the words' idf-frequency, some other just the 'names' appearing in the text, and so on. Some effort might be put into smarter algorithms able to combine on the fly all these different types of inputs, or to extend the range of inputs available, such as through web crawling[4].
- Another thing which might change is that currently algorithms don't (strictly speaking) learn much. They just go on spitting the same decisions over and over, and they gate them through the mechanism of trustworthiness (which is a way to model the self-confidence in the correctness of the algorithms' own suggestions). An attempt could be made to include amongst the angels some more standard learners, such as neural networks.
- Finally, the Guardians' reach could be much extended by having them evaluate not only pairs of clouds but also single clouds, or larger groups. An algorithm for example might try to find hubs in the network, or islands that might then be addressed by 'bridging' attempts. Suppose for example that there is a rather insulated part of the network that is rather inaccessible from the external world. Not only that is an interesting information by itself, but also we may want to fuel discussion by, for example, suggesting more links to 'external' clouds or by giving incentives to those who propose such links.[5]

[4] A first attempt in this direction has already been made: I constructed an algorithm which would, given a pair of words a, b, retrieve the number of google hits for the query "NEAR(a, b)—NEAR(a, b)", later normalizing it with the number of hits for queries containing just the single words a and b. To do the same with clouds is not as straightforward and will need some more research and might involve, for example, keyword extraction (from other Angels' evaluations, maybe).

[5] A sketch of higher-level Guardian Angels, which I called Meta Angels, is already present but has currently not been tested or used. In this case, the meta angel tries to use evaluations as they come out of the lower-level angels and take them further: given

5 God: the Supervisor.

The Supervisor is a gating machine that mediates between the angels and the final output of the system, doing the mostly bureaucratic work of gathering the judgements of the algorithms and agents involved in some voting and weighting them up or down according to the source's trustworthiness, finally merging them (by averaging) into a conclusive judgement that goes to build a belief state.

Such belief state is the final outcome of the system, and represents a snapshot of the link structure within Starfish according to the system. Ideally, we want such structure to closely match the actual network of documents' own link structure (in the case of an hypertext, for example, the links that physically appear in the pages) that in the testing application we used (a portal) have been hand-hard-wired there by physical persons.

But, as the documents' network is allowed to be a dynamical one, where many discussions are open-ended and there is not always a clear yes/no answer to all questions, we won't pretend Semantic Sky to aim at that goal either. Its highest-rated beliefs should of course have a strong correspondence with the actual structure of Starfish, but the whole body of the iceberg (hidden for example through some thresholding) can be open-ended as much as the underlying discussion (carried out by humans in parallel) is. Of course SemanticSky can also be applied without relevant modifications to a statical bag of documents.

These considerations, among others, led us to the conclusion that recall is not a priority for SemanticSky neither as a suggestions machine nor as a link-spotter in general, making it somewhat closer to a ranker.

The fact that there is a centralised object gating the outputs of several other objects (partly algorithmic, but mostly human) makes SemanticSky close to what goes under the label of 'ensemble methods'.

More specifically, we found the (hem) nearest neighbours of SemanticSky in the literature to be the *mixture of experts* methods and the *stacked generalization model* models.[6]

Also, some intuitions behind SemanticSky can be traced back to Neural Networks and / or Bayesian Network, with respect to which the major difference is that the neurons do not just transform the signal but actually produce it, and are not simple (though fuzzy) logic gates but are complex

two items x, y the meta angel's evaluation is a function of how many neighbours x and y share (weighted, clearly, by how near they are) in the current state of the system.

[6] For terminology and explanations, see for example [Introduction to machine learning, 2nd ed. Ethem Alpaydin, pp. 434-435].

algorithms.

5.1 Future work.

When some agent (or a Guardian) communicates to God a new decision concerning some item x, God's belief state is updated by the classical formula

$$new\_value = previous\_value - [(previous\_value - new\_value) * learning\_rate]$$

and nothing more. Some efforts need to be put in determining whether this is the best strategy available. Through the mechanism of contextualized trustworthiness, angels themselves repress their own decisions based on the feedback they received, but this might not be enough, or there just might be more. Maybe some angel's decisions are just all wrong, or maybe it produces such good decisions that even though his confidence in them with trustworthiness at 1 is still not at all high, we might want to boost artificially his confidence: to hear more than we listen to, so to say. God itself might learn, in other words, some useful way of transforming the output to match the current state of the system, thereby minimising his regrets directly (instead of relying entirely on the angel's own learning mechanisms.).[7]

6 Conclusions

In this section we are going to discuss what we learned while working on SemanticSky. First, we will show some graphical and numerical results, in an attempt to evaluate how the system performs, therewith explaining why the results are what they are and not otherwise.

As we already mentioned in the previous sections, SemanticSky was developed to fit an existing online network of webpages, namely Starfish.[8] This network's data has been downloaded and used to test SemanticSky: clouds were built using each page's data and metadata, and angels were built that were able to crunch that specific data.

The main test we developed consisted in removing all the clouds from the sky and initializing an empty God in the empty sky. Then, we add them back one by one[9] and we ask all the guardians to evaluate the available portion of sky; that is: to assess all similarities they can spot between

[7] This, in the light of appendix 2, can be phrased by saying that God might need to have an equalizer as well.

[8] starfish.innovatievooronderwijs.nl

[9] Or, to reduce the running time of the test, a bunch of them per time. In fact, the tests I ran mostly were made with step 5 and 6. This has also the effect of reducing the

the present clouds. Finally, the Knower (a training algorithm that 'knows the truth') judges on their evaluations and triggers feedback, adjusting the trustworthiness of the angels in the various contexts.

6.1 Testing: online monitoring of the system's evaluation.

First I will write down clearly and discuss, where needed, the parameters I used for the test; then show the results.

- Learning rate = 0.4. This is the absorption rate of the feedback: how much of it the recipient is going to take into account. A learning rate of 1 means that the learner will replace its previous trustworthiness with the one suggested by the feedback, a learning rate of 0 means that it will just ignore any feedback.
- Merging rule. This affects how the new suggested value (as computed by the update rule) is merged with the previous one: that is, how the learning rate is used The standard merge is:

$$new\_value = previous\_value - [(previous\_value - new\_value) * learning\_rate]$$

- Punish = False. This means that guardians don't receive negative feedback for links that are not in the knower's database of 'truths'.[10]

To state this more clearly: guardians receive feedback only on items in the knower's database = Starfish's database. The valuation they get for some link, namely, is 1− their current rating of the link.

- Step = 6: clouds were added back to the sky 6 by 6.
- Update rule = median of all feedbacks. The contextual trustworthiness of angels is the *median* of all the feedback he has ever received about links in that area of knowledge (in that context).[11]

---

amount of memory used (by $1/n$th) for n = step. And given that for step 6 it's still $74mb$ of output, this is necessary.

[10] In a real-life application of Semantic Sky this will be a nonexisting issue, but in a test, we have to choose which items we give negative feedback to. In this test I chose not to give negative feedback for all links *not in the network* and mistakenly captured by the guardians as similar. For further discussion and a test with punish True, see the Appendix.

[11] Other available solutions include averaging

$$(sum(allfeedbacks)/len(allfeedbacks))$$

and step-by-step averaging

$$(value = (previousvalue + value)/2)$$

, plus average and median of the last n items.

Follows in a picture, sampled at each loop of the test, the average value of each Angel's beliefs in all true links; that is, all the links that according to our training set (represented by the knower) are truly out there.

We can see how the values increase and (seem to) stabilize for most angels after around 45 iterations. This means that there were 270 clouds (since the step was 6) in the sky at that point. The least stable lines represent the angels which have few nonzero evaluations, and thus receive less feedback.

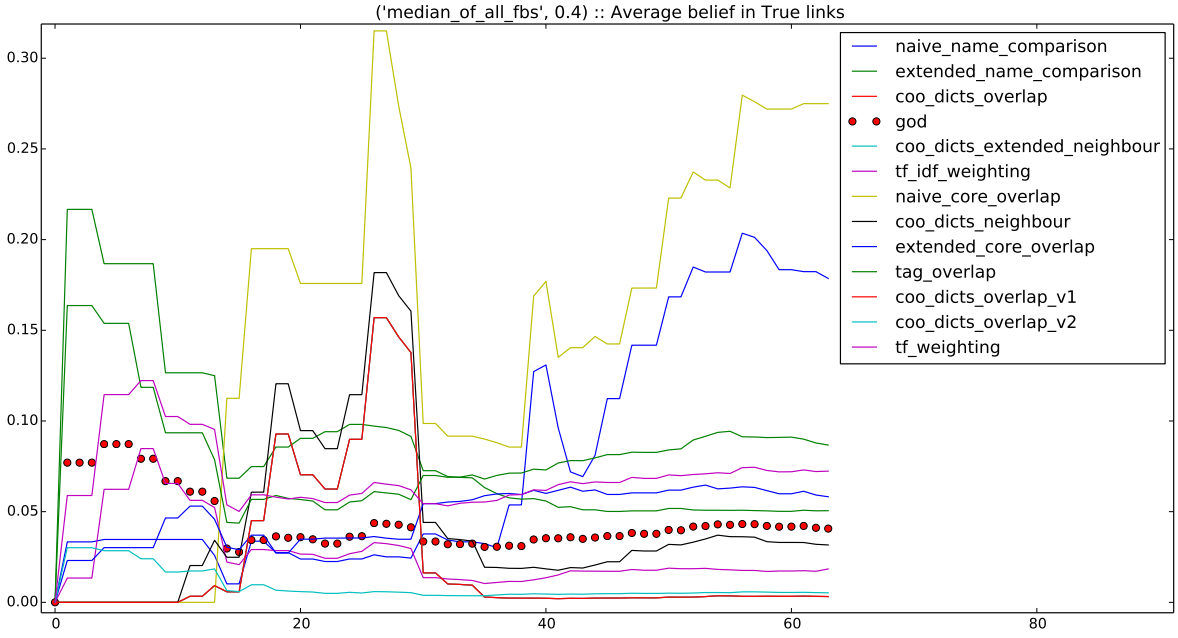


Figure 1: belief in true links

As the test proceeds, the values slowly stabilize around (an average of) 0.05, represented quite accurately by God's line. Note that God's belief set is a weighted average, not a plain one.

Figure 2 shows the progression of average beliefs in false links: that is, links that are currently not in Starfish. The convergence in this case is much faster because there are many more false links than true ones.

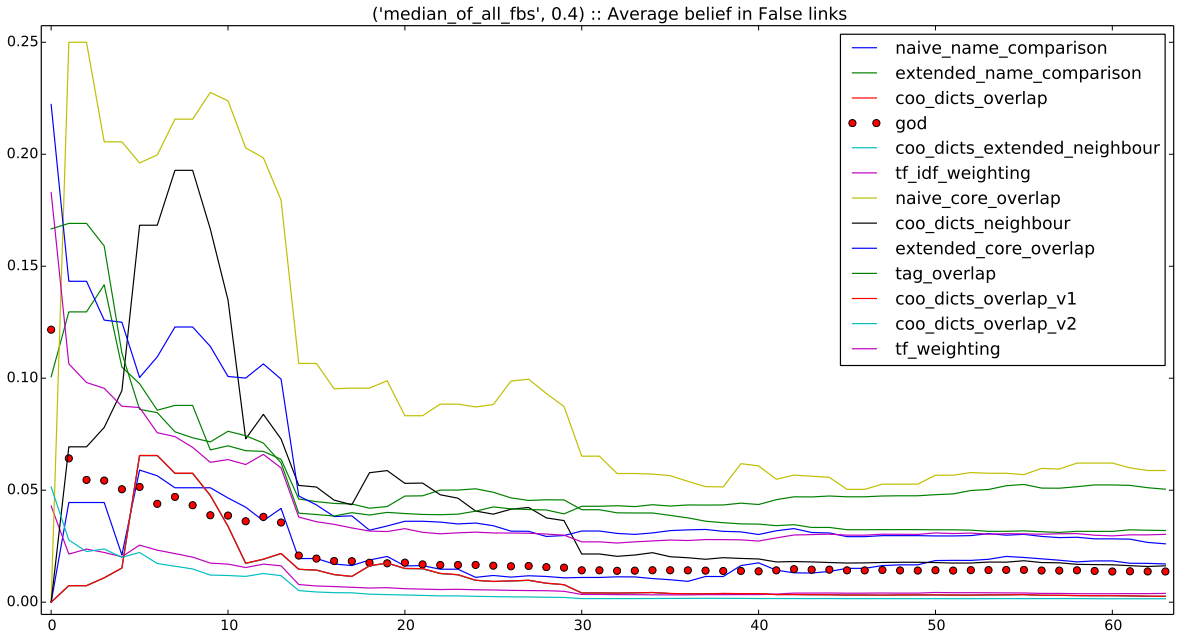


Figure 2: belief in false links

Finally, these are the (non-cumulative) regrets of all angels and God itself.

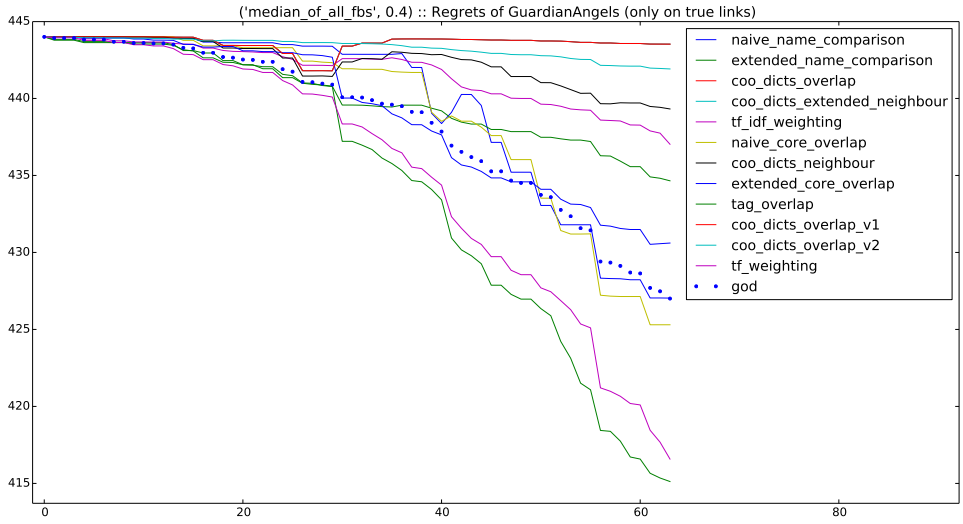


Figure 3: regrets on true links

Regret (for angel A) was computed as follows: $sum((1 - A.belief\_in(x))$ for x in all_truths), where all_truths is a list of all links actually existing in the database (that is: the Knower's knowledge).

The angels whose regrets don't decrease are the tag-similarity angels (hidden for clarity's sake in the next plots). Currently lacking Starfish any data about how similar two tags are, no link of that type (tag-to-tag) is available in Starfish, and all the feedback will be negative.

One nice thing to notice is that all the algorithms' regret increment is stably decreasing: there is no faulty learner. Although, the regrets computed on all links (and not only on true ones) reveals a different picture:

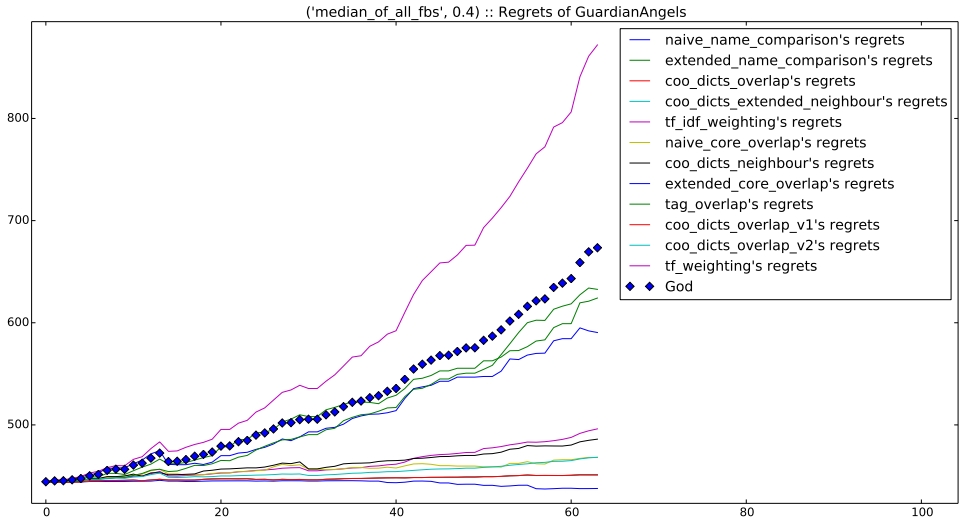


Figure 4: regrets on all links

Here one can see how for some algorithms (the least smart) regrets are increasing almost exponentially (in parallel with the increase of available permutations of clouds as the sky increases in size), whereas the regrets of some other algorithms are even decreasing. God seems to be driven a bit too high, not following the best ones, but the worse.[12]

One can also notice that god's line is not the simple average of the other lines, but is a weighted average: it will be influenced (negatively) proportionally to the regrets of the angels. Ideally, an high-regrets angel should have low average weights, and thus influence less god's beliefs (and finally, god's regrets). The fact that this is not really happening here might

[12]This probably being due to the fact that high regrets means high confidence in the wrong things: and high confidence means high impact on God's belief state.

mean that the low-regrets angels also have low average weights, which is in fact the case.

A few extra facts :

The same test, with different parameters (learning speed 0.4, update rule = *average* of all feedback) yields comparable results (though perhaps a bit less nice):

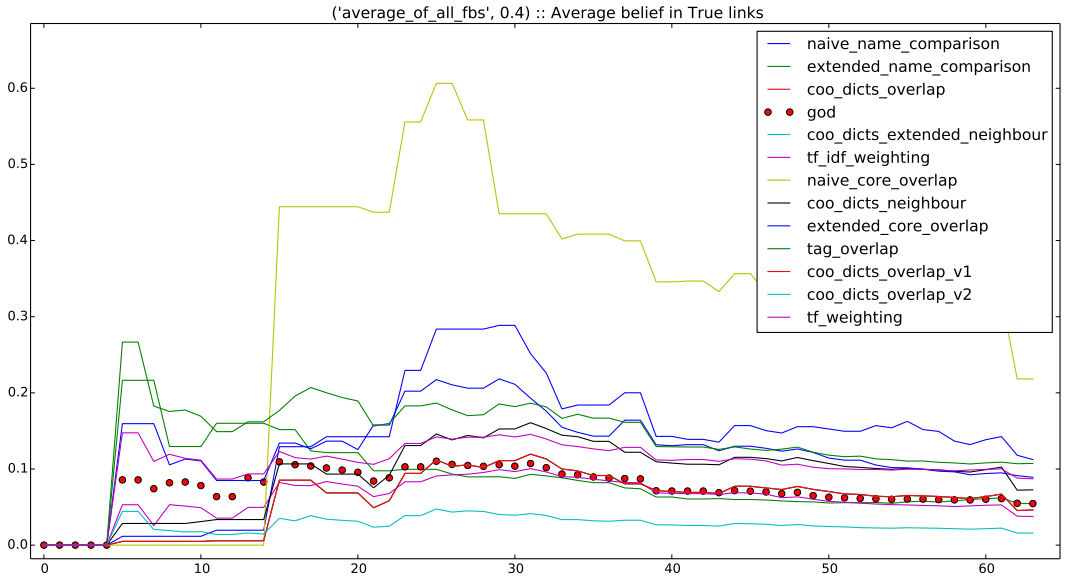


Figure 5: true links beliefs

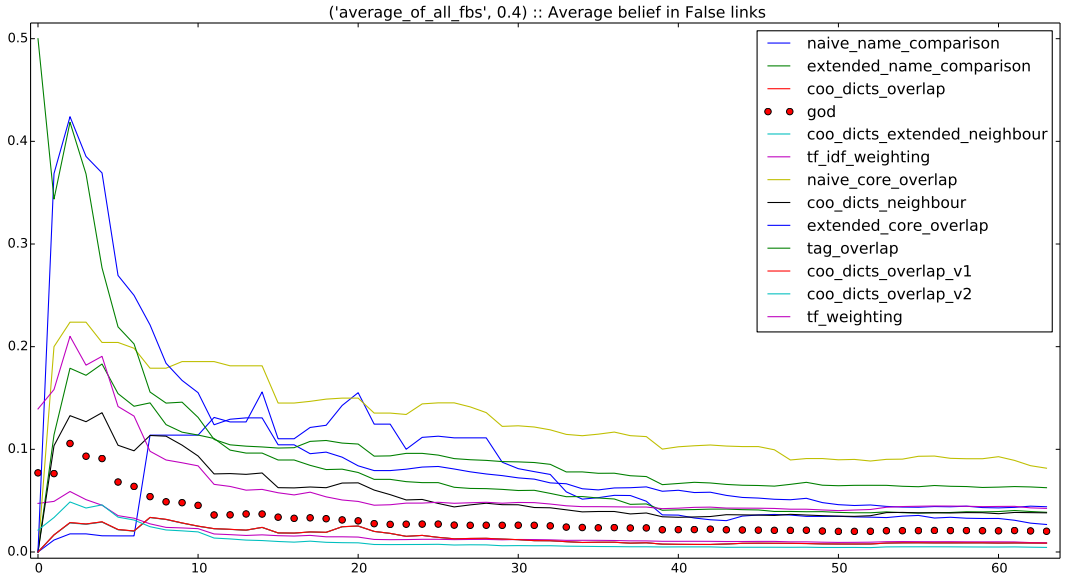


Figure 6: false links beliefs

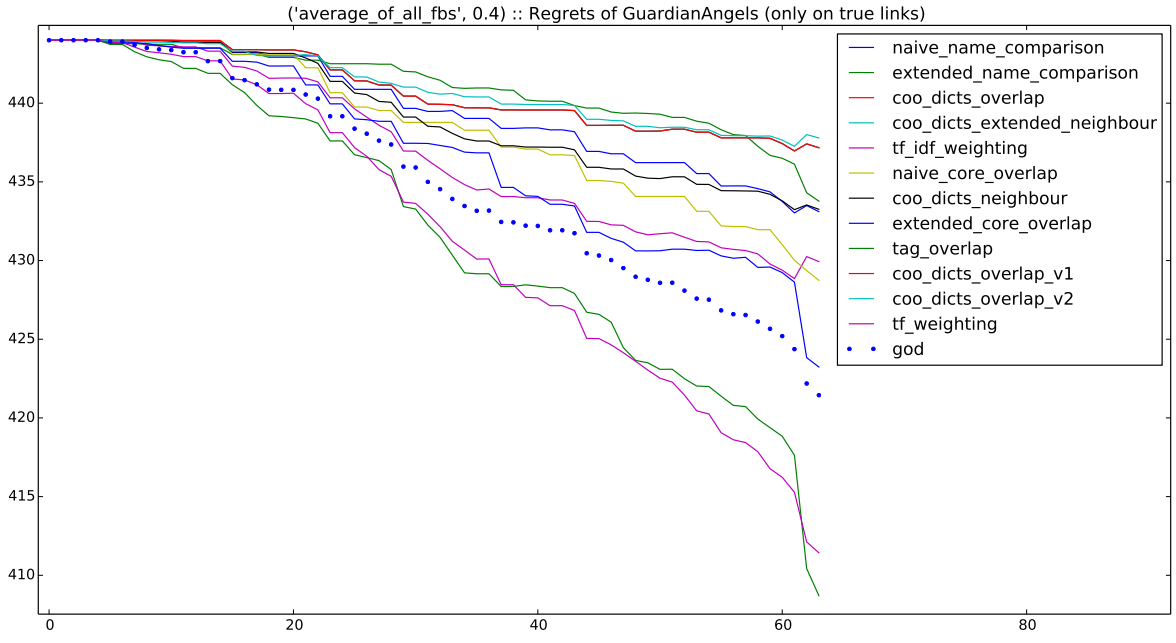


Figure 7: regrets on true links

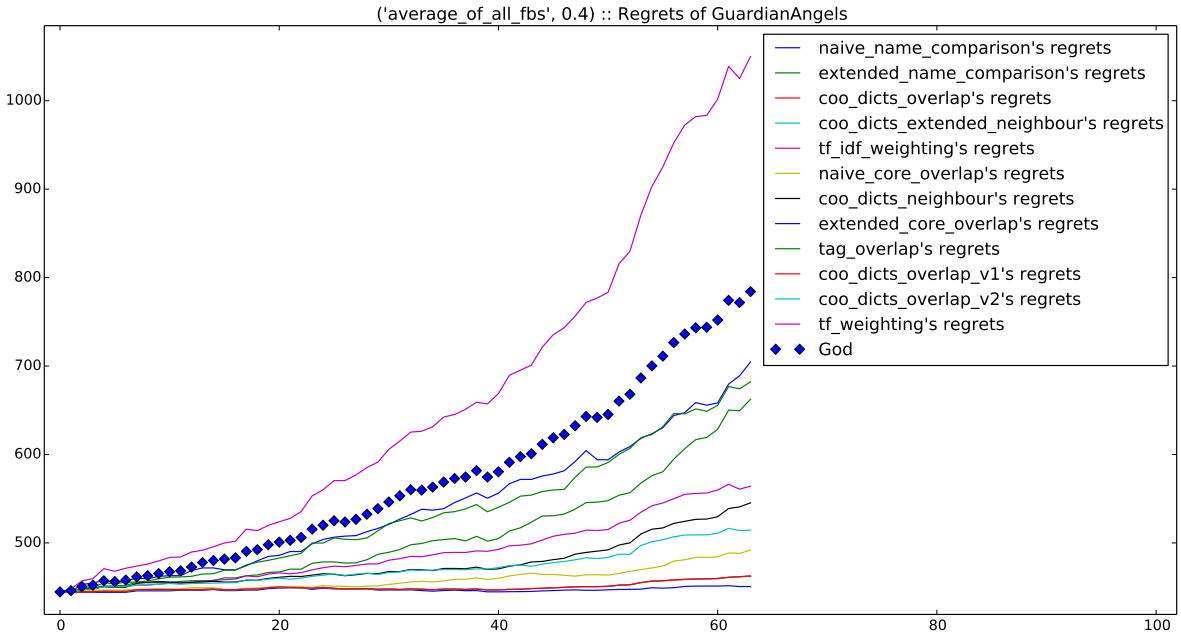


Figure 8: regrets on all links

6.1.1 Further considerations.

Summing up, tests of this kind were ran with the following parameters:

- Median, learning rates 0.2, 0.4(2), 0.5, 0.8 and 1.
- Average, learning rates 0.4(2), 0.8 and 1.
- Step by step average, learning rates 0.2 and 0.8.
- Median and average, learning rate 0.2, punish = True[13].

From these tests clearly emerged that regret on true links (figures 3 and 6) stably decreased, but average belief in true links also decreases or in general stays as low as 0.05.

Also, since clouds were added in batches of 5 or 6, and regret on true links depends on the amount of true links available given the current cloud

[13]This means that feedback was given not on known truths but also on unknown pairs (which, we remind, are assumed to have similarity 0'). More on this in the appendix.

set, it's unclear whether the (final) regrets would have been lower, had the weights permanently all been set to 1.

This doubt is easily dispelled, and the answer follows in a graph.

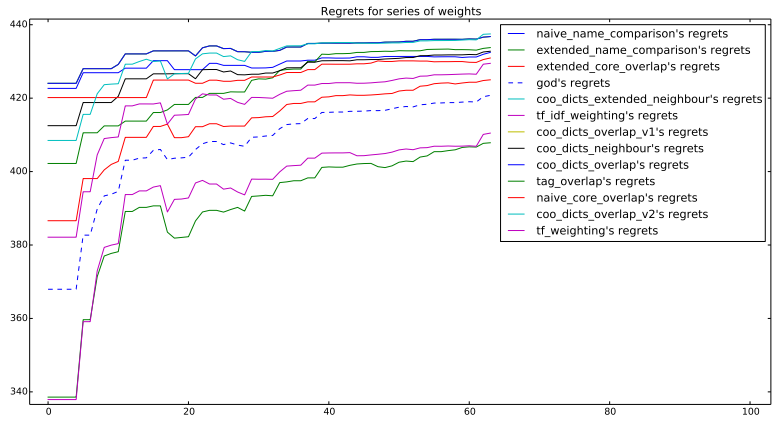


Figure 9: regrets on true links : absolute

This graph represents what happens to a belief set whose weights have been all set to 1 (their initial value), if we assign to it the weights as they have come out of the test we just described. Each step on the x axis then represents the non-cumulative regret at that moment, and depends solely on the weights we have assigned to the system at that step.

As the weights evolve (via feedback) from their initial value to their final one, we can see that the regrets for true links is actually increasing.

Even though, if we don't just look at true links, regrets are falling:

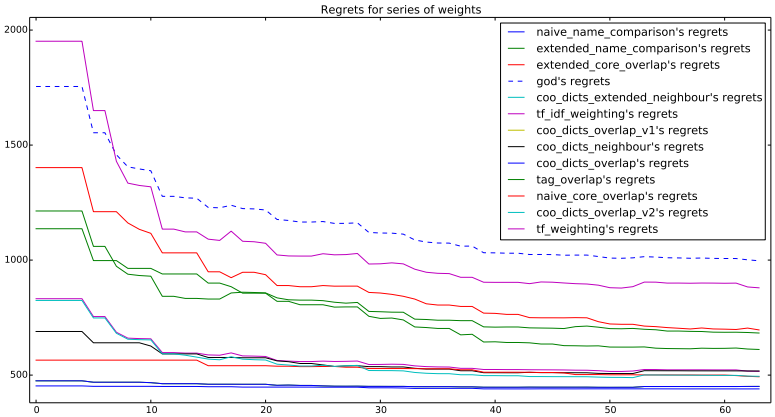


Figure 10: regrets on all links : absolute

So, at an absolute level, it seems that the system is learning the golden rule 'if the 99% of the times the right answer is "no", then I better just say no every time.' To face this situation, it seemed natural to go for some ways of countering sample bias. Some early attempts are described in the appendix.

6.2 What comes next.

• During the development of SemanticSky we had several intuitions of possible things to do, or possible alternative ways of doing things we were already doing. For example, when giving feedback, the standard way to go is to give a feedback closer to 1 the closer my evaluation of x is to the evaluation of whoever I am giving feedback to. A different option however is to give straight what should be, in the opinion of the feedback-giver, the evaluation for that item. While training, this is basically equivalent to having the knower give a 0-1 feedback or a continuum between 0 and 1 (closer to 1 for high confidence correct answers or low confidence false answers, and vice versa). We have run a couple of tests with this alternative algorithm, but due to lack of time we could not double check them and we have no definitive conclusions to draw.

This and a number of other possible alternative settings can and need be discussed. Probably, a bit more conceptual analysis ahead of a 'now sit down and experiment' is preferable, in this case, being there so many parameters at stake whose interplay is in most cases too complex to have a clear balance.

• Closely related to the previous point, the issue on whether to give feedback on false negatives is still open. The main way to find a quick answer to this problem is, to my mind, to run a bunch of test flipping the switch on and off, and see what happens. This has not been done yet due to lack of machine time.

• Another thing that is missing is some sort of benchmarking: we had some ideas about trying to SemanticSky our way through the immense Wikipedia API or LastFm, IGN, or one of the many databases available.

To extend the framework to deal with that kind of clouds is quite straightforward. The main obstacle, for the moment, seems computational in nature: if we choose to punish false negatives (and the answer to 'should we do it' is still open), an evaluation on Starfish took more than two or three days on my laptop. I wonder how many weeks would it take with (a subsection of) Wikipedia. But, given time and a machine one does not need to Skype his girlfriend, this is clearly something we want to do.

- For the moment we did not have time to generalize (codewise) the cloud-formation process. Some programming effort should be taken into doing that in a wise way.

6.3 On generality and generalization.

How general is SemanticSky, or how long would it take to make it (more) general?

To our mind, SemanticSky is quite general already. What is Starfish-specific are the cloud-formation process (that partly relies on the information we have from our testing platform) and the Angels, that are built to use exactly these features.

For example, Starfish items come with a list of tags, which we store in clouds and some angels use as a basis for their evaluation.

More specifically, the database the system has been designed to work with (Starfish) contains data which is partially annotated: each item has a 'title' field, an 'author', and other metadata: what if it had not?

There are many ways to go with unannotated data which would move SemanticSky closer to the unsupervised side of the semi-supervised class of approaches: we could either have some algorithms to pre-classify the documents or parts of them (for example to extract tags, or titles, to recognize authorship or inter-document relations) or we could have very simple clouds with only a 'text' entry, and our angels will have to be smart and diverse enough to work on that sort of data without any more annotated input.

Here I must add that technically, all the work done by clouds could as well be done by Angels alone. The main idea behind clouds was to do beforehand something that angels would have needed to do themselves, such as tokenizing the text down to words, stemming, and extracting the tags that came along with the items. But there is no higher-order reason for which we should keep the two things separated: we might as well decide to abandon the whole cloud system and have the algorithms evaluate over the raw items, whatever they are, provided they come with the same toplevel datatype/data structure (otherwise, also guardians would need to be modified each time).

Personally, I think the cloud layer is not a bad thing and should be kept, mainly because it allows to standardise the input we receive to a more intuitive form, which all angels (regardless of what kind of data they were built to evaluate on) can accept without crashing.

Regarding the way Angels can be generalized there is probably little to say: as specific algorithms they probably can't. What can be generalized is the framework, and probably angels will need to be written depending on

the form of the items we are trying to evaluate upon. We might for example build a SemanticSky that recognizes similarities between songs. Then clouds would store some metadata, if available, or extract some features which we deem relevant such as bpm, number of layers, compression level, noise, and more. But, most likely, a tf/idf weighting algorithm will do little if this is all we have, and return 0 all the time.

Of course, we could build an algorithm that just evaluates the similarity of two clouds at a higher level, but that would be a SemanticSky itself. Generalization of the single algorithms that evaluate pairwise the clouds in the sky, in other words, is out of the scope here. What we had in mind is to make a general framework, but the supervised part of building the specific algorithms able to crunch the data we give them is not (as far as I can see) going to disappear soon.

This is, if you want, the lower structural bound of the current version of SemanticSky.

Part II
Appendices.

7 Punishing false positives and negatives.

The decision to only give feedback on known truths (thus not penalising false positives [what I say is, but is not]) can be questioned. Also, no (negative) feedback is given on false negatives, and this is even more questionable.

In short, feedback is given to all and only those clues which convey a non-zero opinion about a link which our training set labels as true. Angels which spot similarities where we know there are none (false positives), or angels who don't spot similarities where we know there are (false negatives), are not punished for this for two different reasons.

Reasons for not punishing for false negatives: the main one I can see is that we chose to interpret the decision of the angels not as opinions about the relatedness of items, but about confidence ratios about such relation.

The difference is subtle but important: if an algorithm returns 0.5 when fed with two clouds, you can interpret that output in at least two ways:

relatedness "I believe these two items to be 0.5 related / they have 50% in common / they resemble to each other half of how they would if they were

exactly equal."

confidence "I am not entirely sure, but 5 out of 10 these two items are related / I am 50% confident in the relatedness of these two clouds."

The most relevant outcome of this distinction is that on the relatedness reading, a 0/10 rating means 'my opinion is that these two items are not related', whereas on the confidence rating that would be a milder 'I have no reason to believe they are related, but I just don't know'.

Given that the algorithms we used were relatively naive and relied, as explained in the appropriate section of this report, on a (very) restricted subset of the available evidence, it seemed obvious to use the 'confidence' interpretation of their opinions instead of the other one: they have to be humble about their judgements. Their perceptual field is so narrow that an output of 0 is hardly interpreted as a negative relatedness judgement, but rather as a 'here I can't see anything' statement.

As a consequence of these considerations, we chose not to punish the angels for their false negatives: if they can't see anything, it's not their fault and there's nothing they can do about it: it's not by lowering their (relative) trustworthiness that we will improve the situation.

Even though, it's imaginable that there is an angel whose suggestions are all wrong. In this case, his trustworthiness will stay as high as the initial value (in the tests we run, that defaulted to 1!). By defaulting the initial trustworthiness to 0, this issue would partly solve. But maybe there are just sharper ways to go about this.

Reasons for not punishing for false positives: First of all, the Knower's belief set is assumed to be 'all there is to know', literally, but in a dynamic frame such as Starfish's one, we want to always keep the discussion going. When the state of the discussion (the state of the art, if you wish) is not yet settled, when there is no agreement or just too little opinions to have a definitive decision, then the rationale of 'all there is to know' becomes questionable.

But most crucially, what matters for the purpose of classification is not the unrealistic target of all and only true items being detected as such (zero false negatives and false positives), but that the gap in the confidences is wide enough to allow for a nice separation of them from the false ones (via ranking, for suggestions, or by thresholding, for selection of links.)

And, as it comes out, the angels are already somewhat good to do this as they are. The average confidence in true links against false links is, as it

has been previously shown, slightly in advantage of true links. So, instead of punishing an angel's relative trustworthiness regarding some link type because of its (many) errors in which he nonetheless has very low confidence, thereby lowering the future impact of his (few) right clues in which he has a higher confidence, we could just not give bad feedback on the low-confidence errors.

Clearly if an angel had high confidence on some or many errors, the situation would need more accurate consideration, and it might even be worth considering some thresholded triggering of feedback based precisely on a trustworthiness / confidence function, so that we give feedback on items only when the angel is trustworthy enough in his own suggestion, or that the learning rate for each feedback is some function of these factors.

A (quite faulty) backing experiment. As a partial proof of the usefulness/correctness of this stance, we ran a test in which we set the punish flag to True, thus giving feedback to false positives (to give feedback on false negatives is currently just not possible in the framework, but it will in the future). Graphs of the outcome follows, and are rather self-explanatory. Update rule: median, learning speed 0.2.

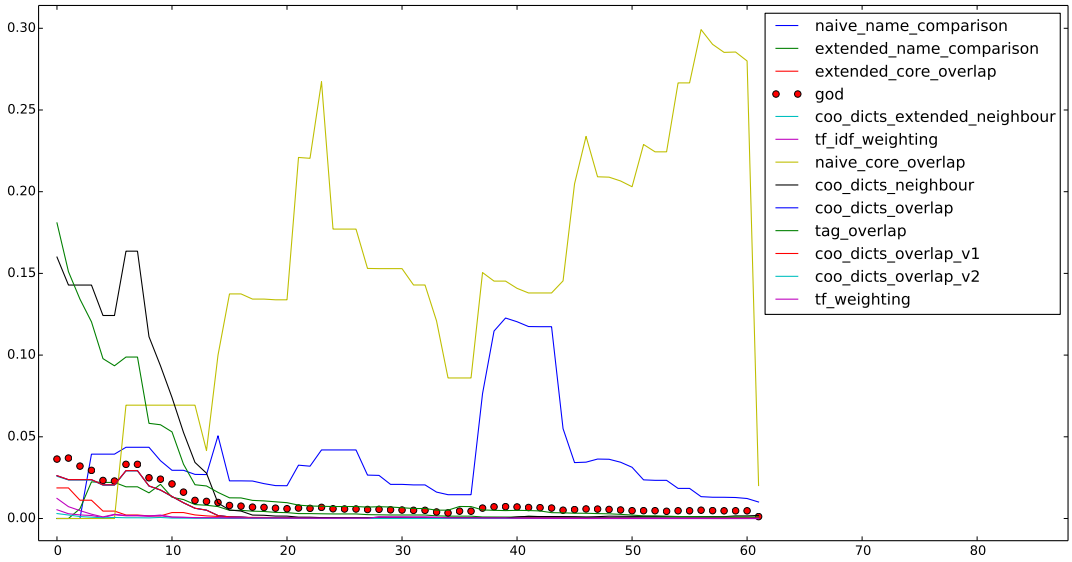


Figure 11: average belief in true links

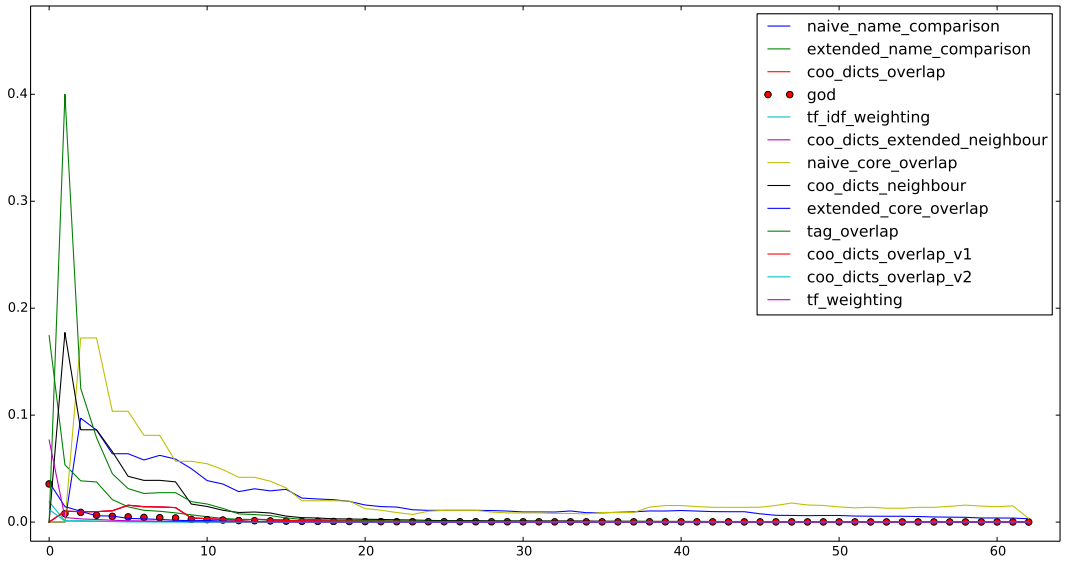


Figure 12: average belief in false links

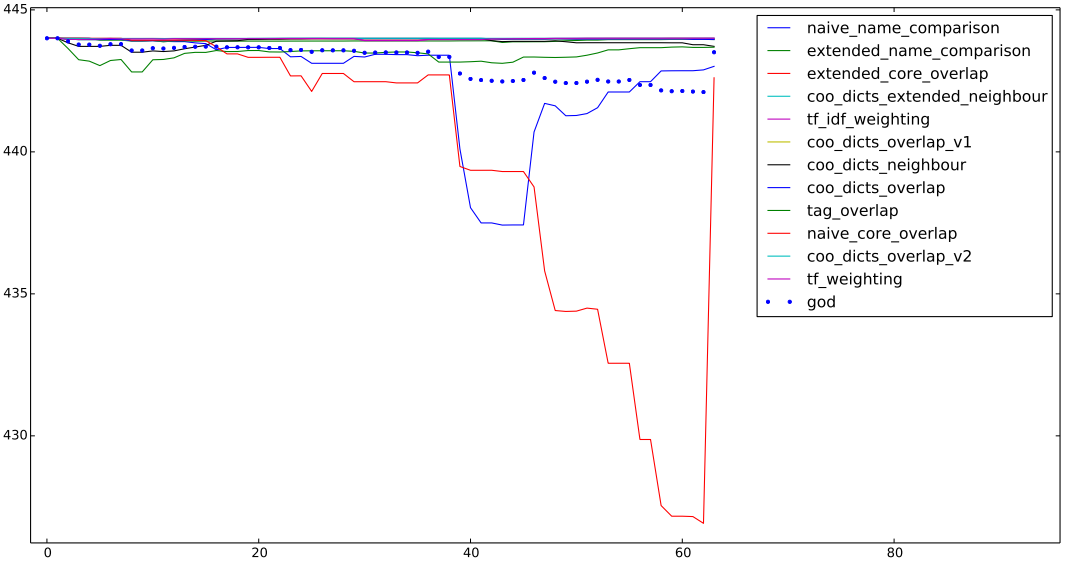


Figure 13: regrets only on true links

As before, the two following, final figures represent the outcome of a different test (on the present test): taking the weight sets as sampled during the evolution of god's beliefs as the present test (median, learning rate 0, punish) proceeded and assigning them to a different god, we have him refresh his belief set according to these weights and then plot the resulting regrets.

As you can see, also by comparing these figures to the previous ('healthy') ones, the regrets in this case touch the top (440) and seem stably hang to the asymptote. In the 'all links' version of the regret calculation, on the other hand, we have an incredibly descending curve which stabilises at a very low value (1250 for God's regrets), against the 2000 top in the non-punish case(s).

This is due to the fact that all trustworthiness ratios go down, and thus, being most links false (i.e. not in Starfish) the angel's guesses get statistically more correct as their weighted value approaches 0. (This is also known as sample bias.)

Of course we would like regrets to decrease (especially where God is concerned), but as you can see from the graph displaying the evolution of regrets on true links, this is not happening in an altogether desirable way.

Nonetheless, the results of this test are not totally satisfactory and most certainly not conclusive. The sample bias we face in Starfish (with 444 existing links versus circa 74691 possible ones, including tag clouds) has probably had a role in taking down the trustworthiness so much. Maybe employing some sample bias countering technique such as the ones discussed in the next appendix could show us an entirely different picture.

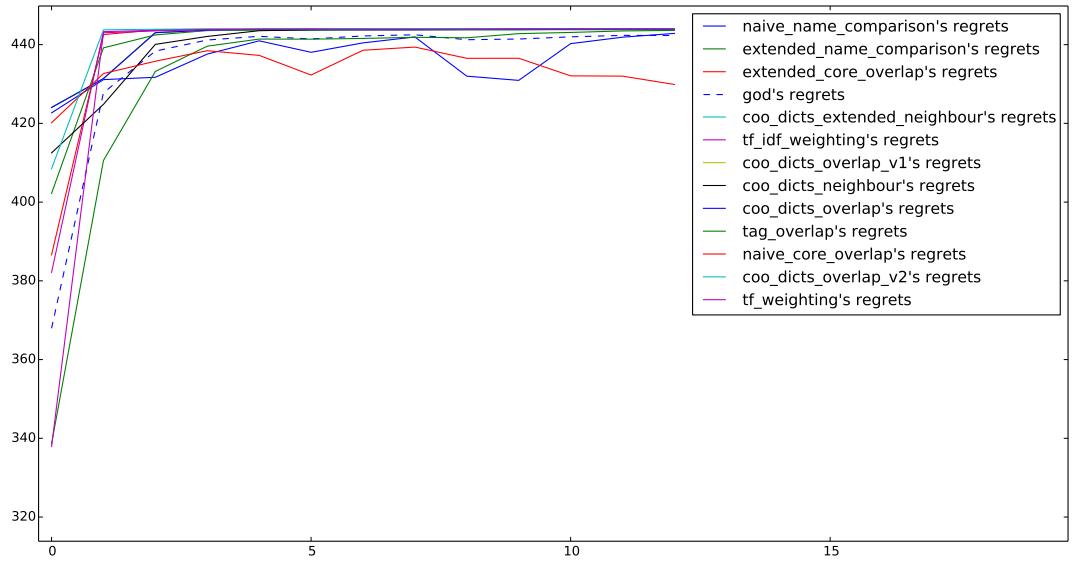


Figure 14: evolution of regrets (on true links)

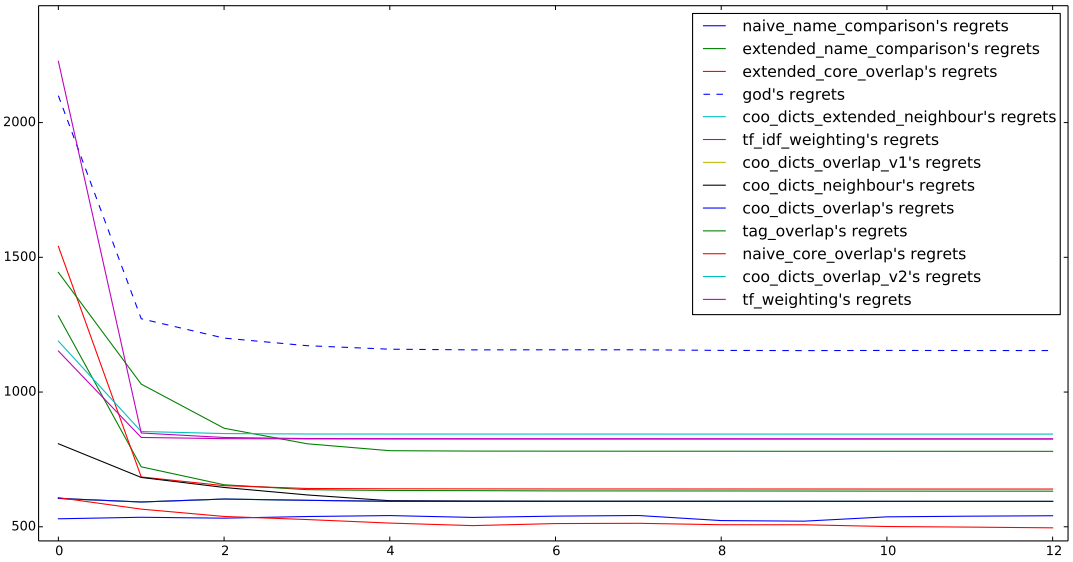


Figure 15: evolution of regrets (on all links)

Even though these graphs seem to argue definitely in disfavour of punishing false negatives, this bad performance might be due to the huge sample

bias we have to face in Starfish.

Besides trying to compensate that in some ways (see next appendix), trying SemanticSky on a more balanced or simply bigger database will probably give more definitive results.

7.1 Further testing: differentiating learning rates to counter sample bias and equalization to counter algorithms output bias.

During the first round of testing, we noticed that there were two important problems that the framework was not able to cope with:

- a great sample bias (true links were 444, versus more than 21.000 false negatives in some cases).
- what I will call **output bias**: most algorithms' normalization of the output produced an uneven distribution of values pushed towards zero. Thus, the distance between the average of the values which received positive feedback (meaning that the guess was correct) and the average of the values which received a false feedback were quite near. If we want to call 'confidence ratios' the raw outputs of the algorithms, we can phrase the problem by saying that the confidence ratios were on average very low (the highest average for all the algorithms being 0.3, and the most of them leaning towards 0.05.).

To counter the first, we introduced differentiation of learning rates for positive and negative feedback.

To counter the latter, we tried to use what I will call an **equalizer**: a map from raw output of the algorithm to a more even distribution of values, evaluated online (or updated periodically) and eventually learned by the algorithms themselves, that tries to push apart the average of values that received a positive feedback and the average of these which received a negative one.

7.2 Differentiation of learning rates.

An option known in machine learning for its effectiveness is to use different learning speeds for positive and negative examples, diminishing the latter by a factor that depends on the size of the sample bias we face.

No testing was made in this direction, even though the code already includes this possibility, disabling it by default, mainly because at the moment,

as you will already know, we just are not giving any feedback on negative examples. Beside taking too much time to run, punishing false negatives had some other cons you can find in the appropriate appendix.

7.3 Equalization.

The first experiments (as discussed in the central sections of this report) showed that between average confidence ratios in true and false items there was a gap, although maybe little. Thus, simple thresholding might have worked in pulling them apart.

We were hoping that through weighting of output / feedback, that gap would increase and be a more accurate division line between true and false links. Nonetheless, what typically occurred was that the weighting, which, you will remember, can only take down trustworthiness[14], would push down all values. And being the evaluation spectrum [0/1], generally the values closer to 0 the positive pole (average of the values which received a positive feedback) would move down more than the negative one.

At the end of this process, the gap had in fact decreased. In this table, you can read to the left the name of the Guardian Angel and on the right the difference between the gaps before and after a full evaluation (and a full round of feedback). A positive value means that the gap has increased, but unfortunately you cannot see any.

[14] And this, once more, is something to question.

Table 2:
Polar gap increase from unweighted to weighted beliefsets.

angel	polar gap increase
coo_dicts_extended_neighbour	-0.00201
coo_dicts_neighbour	-0.012
coo_dicts_overlap	-0.0037
coo_dicts_overlap_v1	-0.00202
coo_dicts_overlap_v2	-0.0037
extended_core_overlap	-0.027
extended_name_comparison	-0.017
naive_core_overlap	-0.21
naive_name_comparison	-0.15
tag_overlap	-0.036
tf_idf_weighting	-0.014
tf_weighting	-0.039
god	-0.032
sum (including god)	-0.548

Overall, the decrease rate of the gap is not so high, but if this is not evidence that something is wrong, then few other things could be.

With equalization turned on (keeping the other parameters still), this is what we have:

Table 2:
Polar gap increase from unweighted to weighted and equalized beliefsets.

angel	equalization	weighting	overall
coo_dicts_extended_neighbour	0.00074	-0.0019	-0.0012
coo_dicts_neighbour	-0.0081	-0.0102	-0.018
coo_dicts_overlap	0.0039	-0.004	-0.000104
coo_dicts_overlap_v1	0.00077	-0.0019	-0.0012
coo_dicts_overlap_v2	0.0039	-0.004	-0.000104
extended_core_overlap	0.0075	-0.033	-0.025
extended_name_comparison	0.032	-0.025	0.0076
naive_core_overlap	0.0012	-0.21	-0.21
naive_name_comparison	0.00069	-0.15	-0.15
tag_overlap	0.018	-0.042	-0.023
tf_idf_weighting	0.0301	-0.018	0.011
tf_weighting	0.038	-0.0506	-0.011
god			-0.032
sum	0.1287	-0.5506	-0.453

The middle column is the step from the raw evaluation to the equalized evaluation, the right column is the step from the equalized evaluation to the final state of the belief set: that is, the weighted (by contextual trustworthiness) and equalized belief set.

As you can see, equalization manages in all but one case (where probably the algorithm is *so* bad that its average confidence in correct suggestions is lower than the that in wrong ones) to widen the gap between the poles. The problem is that then weighting manages to destroy that effort. Maybe this could be countered by making equalization heavier (the curve was very flat close to the centre of gravity of the push, and unfortunately that's precisely where most values are!), but maybe this is just not working.

In most cases, the overall value (gap difference between unweighted, unequalized and weighted, equalized belief sets) is still negative, which means that the gap is decreasing. We can see however that except for a few examples (still too many, maybe) equalization has managed to reduce a bit (still too little, probably) the negative squeezing effect of weighting: if we look at the 'sum' lines, more or less one fifth effect decrease, more or less the sum of the gap gain in the 'equalization' column. Even though, that was not enough to influence God's ideas: his gap decrease is exactly the same at least up to the third decimal digit.

Besides, it's questionable (though it seemed quite natural to us) the order we chose for the pipeline: we might as well equalize up the weighted beliefset, instead of weighting down the equalized beliefset. Who knows what might happen.

More experimentation and research are needed to draw definitive conclusions here, but we think that this equalization business is probably going in the right direction, but just 'not enough' to solve all of our problems, including the weather.