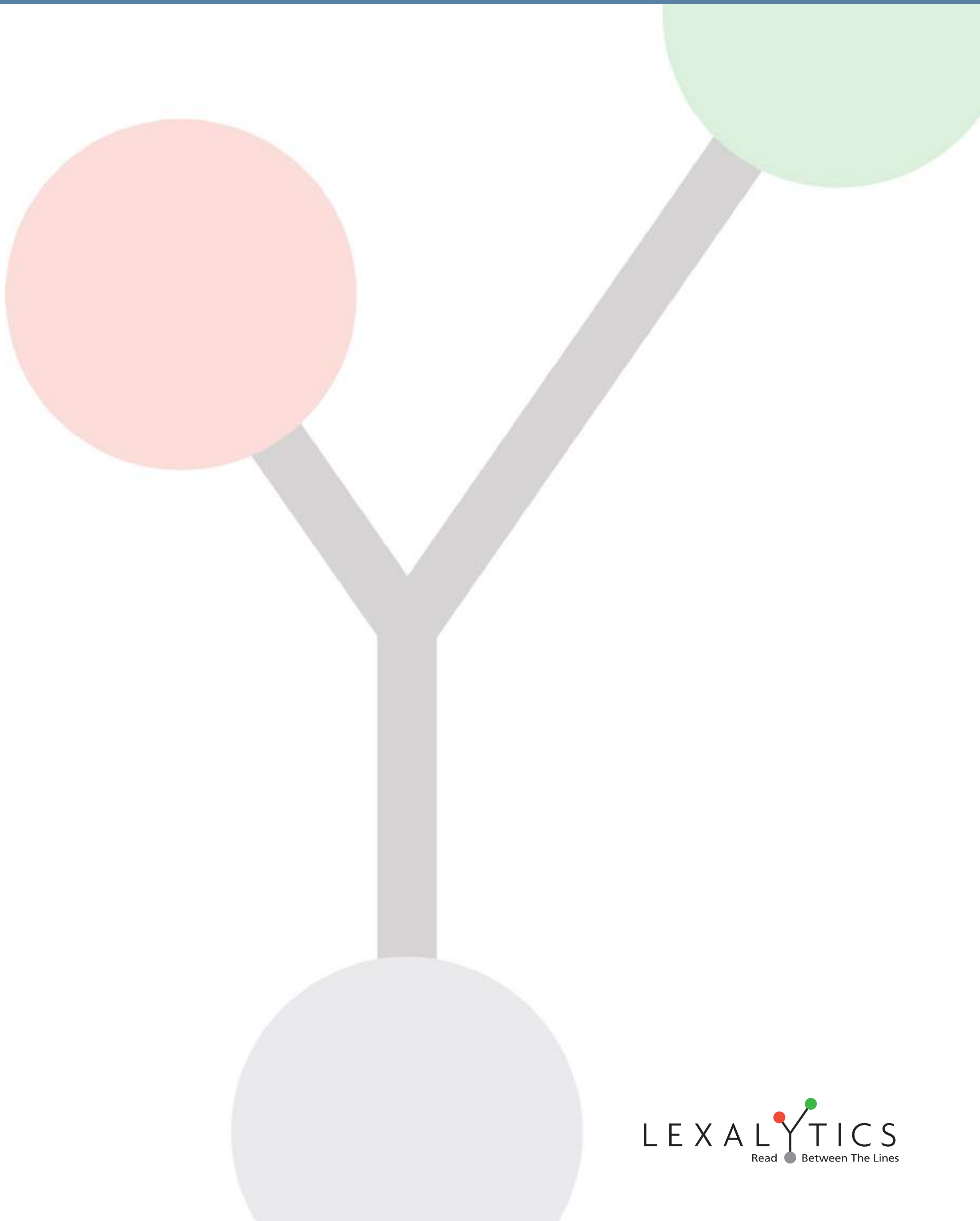


Build Vs. Buy For Text Mining

Why use hand tools when you can get some rockin' power tools?

Whitepaper

LEXALYTICS
Read Between The Lines

We, at Lexalytics, see a significant number of people who have the same question

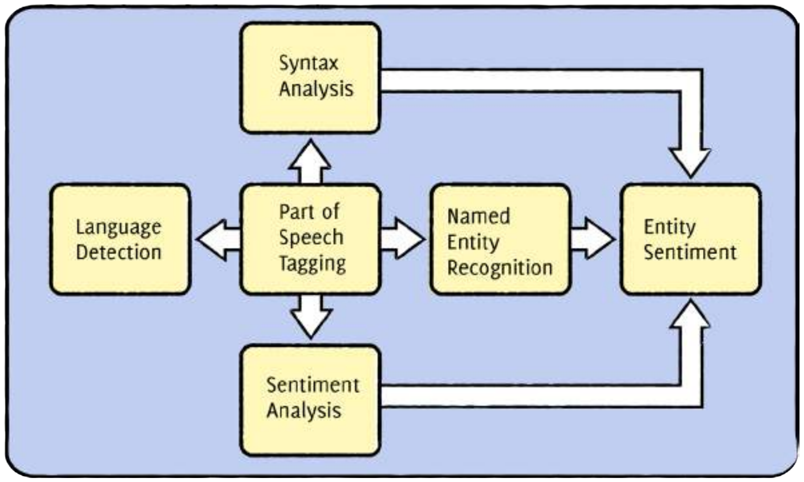
"Why should I buy this software when I can simply use open-source software to roll my own system?"

Since open-source doesn't come wrapped into a "system" per-se, this is effectively a **build vs. buy decision**. This paper discusses the advantages and disadvantages of each approach, and is intended to be vendor-neutral. In other words, the arguments made here apply to many other commercial text mining systems as compared to "rolling your own."

Let's assume that you want to build an entity-based sentiment analysis system to extract companies and associated sentiment from Twitter. This is a relatively common case, and the steps and process are representative of what you'd need to go through for many other cases, including categorization, document-level sentiment, or useful summarization.

Note: Obviously, each of these use-cases will have different details, but the effort is going to be roughly the same, as are the benefits of using off-the-shelf vs. home-growing your own text mining system.

There are several moving parts, as shown below.



Language detection, Part of Speech Tagging, and Named Entity recognition are typically machine learning tasks. Each one requires its own annotated corpus of tens of thousands of documents per-language, and potentially per-document type.

Lexalytics has spent many person-years on just the PoS tagging aspect of the system. Without broad-based language detection and PoS tagging, the rest of the system is going to be fragile and error prone, working only on content that is similar to the training sets.
Each language requires training for those three basic tasks, requiring an investment in the content, annotating the content, then training the system and keeping it up-to-date on changes in the language. Each of these tasks requires months of investment of tagging time, and then high-level expertise to correctly process the training sets into useful models.

These are ongoing tasks, requiring updating on a regular basis.

Language constantly changes – just ask any teenager.

Sentiment analysis can be performed as a machine-learning task, building trained models for sentiment. This approach is limited in terms of being able to associate sentiment back to an entity – as well as requiring new training sets per-language + per-vertical + every time you want to make even a minor update. So, it is more useful to use NLP techniques to build out sentiment and then use syntax analysis to associate the sentiment back to the entities. This requires heavy investment in the core NLP technology for each language supported, as different languages have different requirements. For example, Chinese doesn't have any whitespace for tokenization, and some languages are "subject verb object" (English), whereas other languages are "subject object verb" – (German).

You also need to consider things like anaphora resolution.

For example: "Lexalytics had a great year. They are doing really well."

Who is "they"? How do you apply the sentiment back to "Lexalytics"? , Similarly, another problem is understanding female names vs. male names in order to correctly deal with "her" and "he". Stemming is a third language-specific problem, and if you don't stem correctly, it's really hard to be able to make the correct correlations in the content.

In other words, there's a lot of moving parts that require constant attention. Is that really how you want to be spending your time? Or do you want to be worrying about how to crush your competition and rule the world?

The following are some very specific reasons why using a mature, off-the-shelf text mining system is going to be more profitable and less costly to use.

Let's consider one of the simplest possible cases: Document sentiment for a specific language in a particular vertical.

For this case, you could cheat a little bit (with a cost of precision and recall) by just using a "bag of words" approach to sentiment.

Say you ignore Part of Speech tagging, and you just want to train a model using basic machine learning techniques to give you some analysis of sentiment. You're not going to worry about associating the sentiment with the entities, you're not going to worry about language detection, you're not going to worry about NLP.

Simple, right?

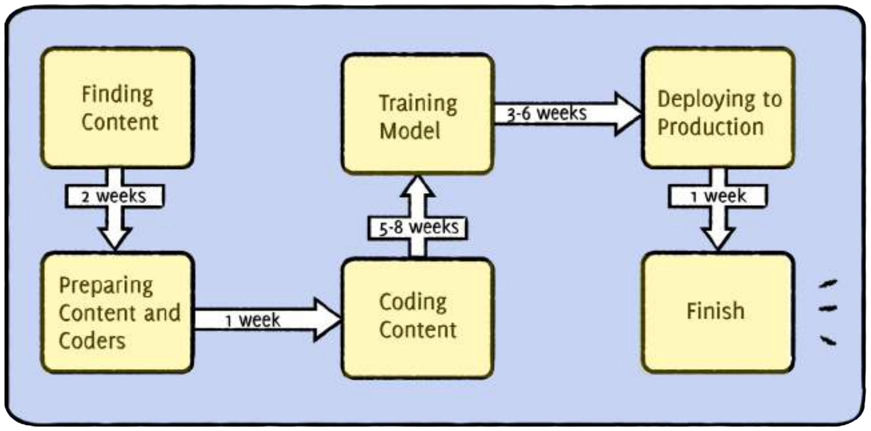
Well, let's walk through the steps.

Finding content
First you have to collect a set of content. Let's choose a low number: 50,000 documents. Assume that this takes you a week to do – find the content, get the correct license for it (a whole discussion in and of itself, you're going to be deploying this for commercial purposes – many corpora do not allow commercial derivative work).

The content has to be cleaned up, because there's going to be some messy content in there. Assume that this takes you another week or so to do correctly. You're now at 2 weeks of time, and that's just assembling your training set.
Time: 2 weeks.

Preparing the content and coders
Now you need to annotate the content for sentiment. This will require multiple raters so that you can calculate inter-rater agreement, and make sure that your coding instructions for your raters are correct. Speaking of which, you have to actually write some coding instructions. You need to make sure that your coding instructions are clear and generic, to get consistent results. You'll need to run a test of your coding instructions so that you can actually get feedback and test for an acceptable inter-rater agreement. That's another week to actually get your coding instructions correct.
Time: 1 week.

Coding the content
You have to get your content coded by a minimum of 3 to 5 coders. Do you have those resources on staff, or are you going to use a service like Mturk? If you have them on staff, you can make sure that everybody scores the same pieces. This is going to take a few weeks to do, weeks of just scoring content.



Instead, most people will crowdsource. But, crowdsourcing means that you're going to have many raters of varying quality. You can approach this in one of two different ways:

1. Assemble a set of coders by running test sets and setting up a custom qualification test to make sure that they understand the questions, and only allow them to code the content,

or,

2. Use a custom quality test, but release it more broadly. You will need to insert gold standard questions, and will have to disallow responses from people who fail the gold standard test.

However you approach this, you will need to allocate around 3 weeks to get the content scored, tested, and verified.

Now you check your inter-rater agreement on the content, and if you're well prepared and a little lucky, you'll get something north of 70% inter-rater agreement. Anything less than that, and you probably have to go back and tune your coding instructions or change your dataset.

Time and cost: you're looking at 5-8 weeks of time, and just for coding, you're looking at $12,500 (assuming .05/coded response for good coders on short content, and 5 coders per piece of content).
Time: 5-8 weeks.

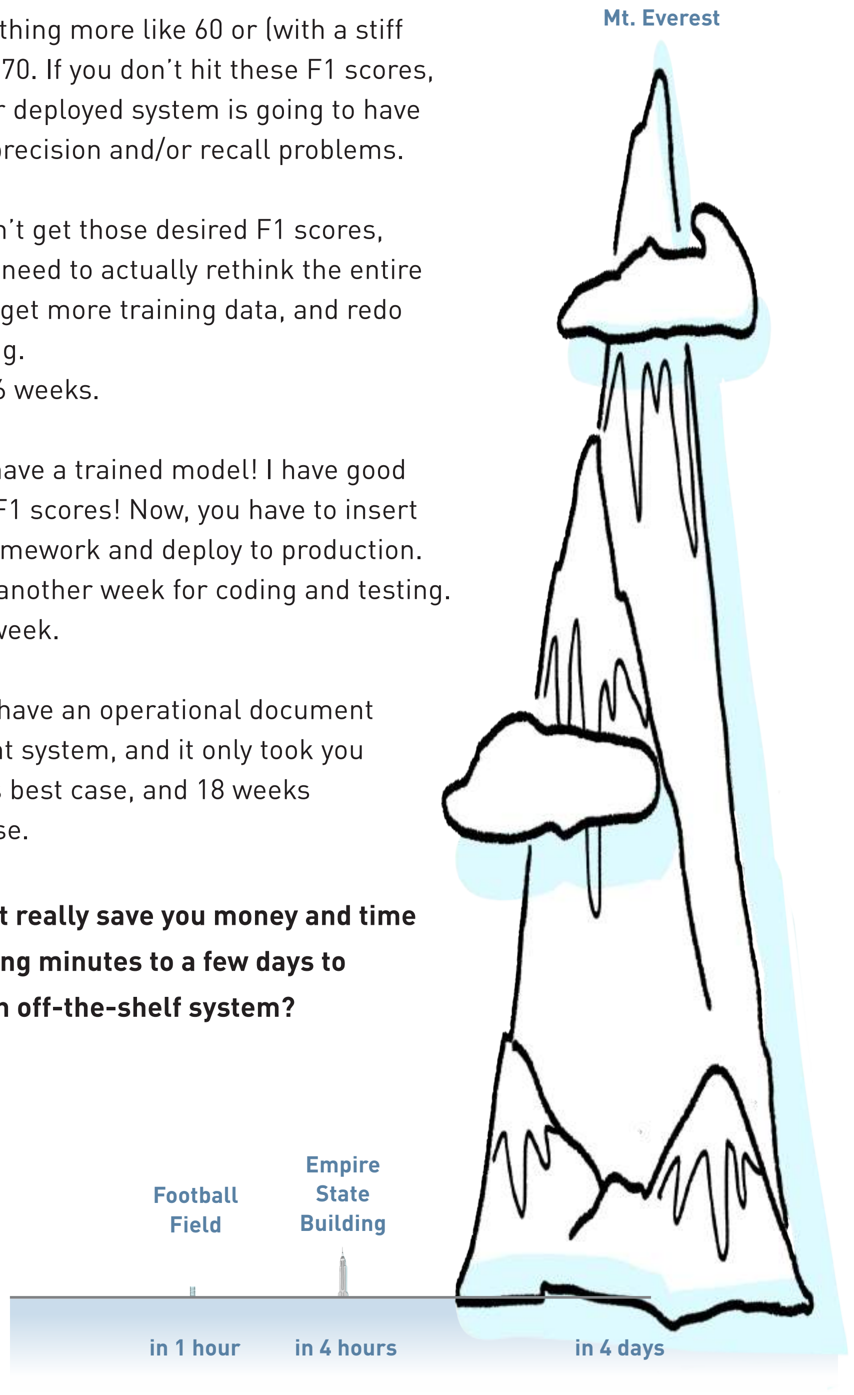
Training your model
Assuming your inter-rater agreement is good, you can now train your model. Yay! Be sure to test against the part of your corpus that you kept aside for testing your new model. Assume another week for training and testing. Again, if you did a good job in the other parts, you should see F1 scores that are at least 50, with hoping for something more like 60 or (with a stiff tailwind) 70. If you don't hit these F1 scores, then your deployed system is going to have serious precision and/or recall problems.

If you don't get those desired F1 scores, then you need to actually rethink the entire process, get more training data, and redo everything.
Time: 3-6 weeks.

Great! I have a trained model! I have good IRA and F1 scores! Now, you have to insert into a framework and deploy to production. Assume another week for coding and testing.
Time: 1 week.

You now have an operational document sentiment system, and it only took you 12 weeks best case, and 18 weeks worst case.

Does that really save you money and time over taking minutes to a few days to deploy an off-the-shelf system?

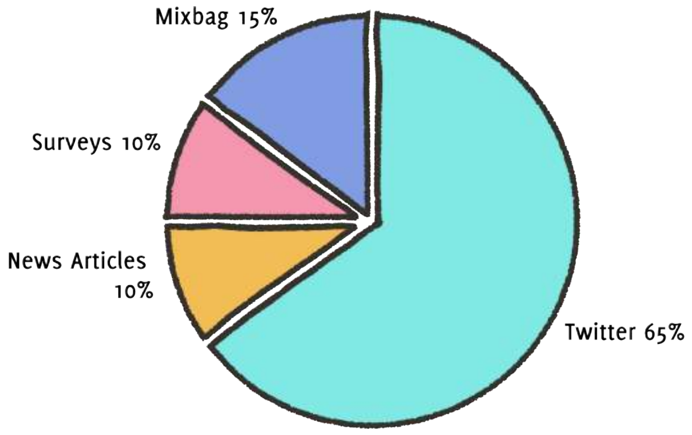


Lexalytics software has been heavily optimized for scalability.

We have billions of documents a day being processed through our software. Our SaaS service, Semantria, has individual customers that are processing tens of millions of documents per day, every day, day in and day out.

How Much Data is There?

If someone were to strip the text from every tweet as it was published, and then compile and print them, just one minute's worth of Twitter would fill 9,267 pages, or 5 feet.



What We Analyse

To give an idea of the scalability issues, we perform all of the following tasks:

1. Tokenization
2. PoS tagging
3. Chunking
4. Sentence breaking
5. Lexical Chaining
6. Named Entity Recognition
7. Theme Extraction
8. Sentiment Analysis (document, entity, theme)
9. Categorization/Classification
10. Summarization

And we do it fast enough to do 200 tweets/second per processing core. We are able to scale to handle the largest of loads because we're implemented in C for speed, and have learned a lot of tricks for handling language efficiently over the ten years that we've been shipping commercial software.

Low level tricks like having a "chunk parser" or using carefully maintained patterns to glue parts of the system together. Most customers will never even have to be exposed to these aspects of the system because we're simply handling the problem of speed.

Research systems, like many of the open-source systems, are optimized around being able to train on odd use cases and moving science ahead, and they are not optimized for raw processing speed. If you are building to scale, you will need to consider just how to scale your system, and we have lots of experience in bringing new functionality to scale.

Each language requires love. Handling a language natively, and providing the depth of functionality and processing efficiency is a technically complex problem.

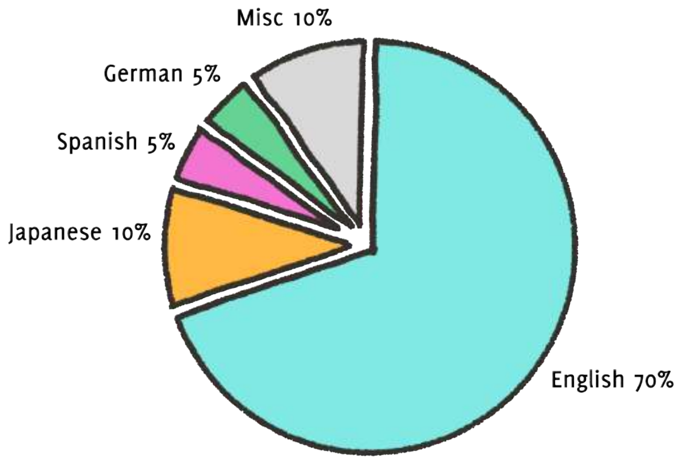
Each language has its own peccadillos – Chinese with its lack of whitespace, accents in languages like Portuguese, different syntax orders.

In order to support more than a single language, every part of the system must be able to "flex" to meet the individual semantic and syntactic needs of that language. To expand on the use case of "sentiment analysis" above – even doing a bag-of-words approach would require you to go through the same process that we just discussed, with months required to train and deploy a simple document-sentiment system. And that is if you can get access to the language expertise necessary to accurately train a system.

In order for us to do a new language, we have to:

1. Recognize the language
2. Train a new tokenizer
3. Train a new PoS tagger
4. Train a new Named Entity Recognition model
5. Process a dataset like Wikipedia that allows us to perform lexical chaining – so that we can understand the semantics of the language
6. Build a new sentiment dictionary

And that's the minimum set of work for us to support the same richness of features across all of the languages. This is months of linear time, and years of person-time per new language.



Languages Breakdown

We have customers that bring our system online in minutes.

Why?

Because our systems come wrapped in a number of popular programming languages, have well defined and documented APIs, and because we produce results right out of the box.

And many customers just stop there. They get reasonable results, and they move on to worry about other things, like selling their product and making tons of cash.

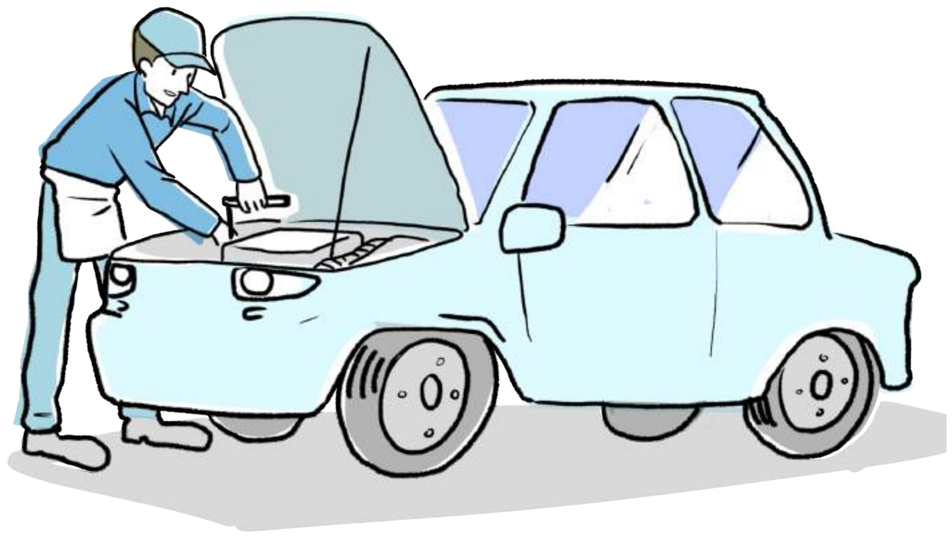
But, things change, and some customers have needs that require more customization and configuration.

Remember that sentiment model we spent a few months building? No configuration possibilities there. You have to get a new training set and go through the whole rigmarole.

By using a combination of deep learning and NLP techniques, Salience and Semantria allows for rich customization through immediately accessible configuration files. Need to change the sentiment on a document? Easy – just change the sentiment phrases. Need a new categorizer? Easy – just put in a search phrase, tweak the automatic document categories, or use our Wikipedia™ based concept topics to catch really broad stuff where you don't know all the keywords.

Do you have a multi-tenancy problem? Do you have multiple customers each with their own desire to make their little bit of the world that much better by customizing? We've already considered that problem, and provide easy to integrate configuration tools. This is very different and much more powerful than deploying a one-size-fits-all solution.

Want to prototype in Python, and deploy in Java? Have a need for PHP, or want to be rocking with C? One advantage of commercial systems like Lexalytics is the broad based support for different programming languages, allowing you the flexibility to use the right one for your installation and needs, rather than jamming a particular language down your throat.



DEPLOYMENT, CONFIGURATION AND MAINTENANCE

This isn't really a build vs. buy decision.

You should be able to make the decision of cloud vs. on-premise based on your needs. You can certainly take your sentiment model that you so laboriously built and deploy it in the cloud. But, then you have all sorts of things to worry about like queuing and auto-scaling.

Do you want to be worrying about the operations of your cloud text mining system, figuring out which Amazon cloud center went down, bringing up a new set of servers when one of your customers suddenly dumps a bazillion documents on you, and handling the irate phone calls when this oh-so-important project is getting held up?

Or do you want to use a system that already has all that stuff built into it, where all you have to do is connect and start sending data, and we have built a clever architecture that simply handles all of these problems for you?

TOTAL COST OF OWNERSHIP

Finally, let's discuss the total cost of ownership.

We already outlined the time commitment for building your own system, which was between 12 and 18 weeks, and this is assuming everything goes swimmingly. In terms of cost, there are many other facts to consider.

By far the most costly step is to score (and code) the content and finding a machine-learning expert to train your model. From our experience, scoring and coding alone typically cost $12,500 while training a model generally was double that. Not considering all the other costs, this alone sets you back $37,500.

However, once we add in the hardware cost and integration cost, you're looking at close to a $40,000 bill once everything is up and running.

On the other hand, integrating a fully featured text analytics platform, i.e. Lexalytics, will cost you less than $12,000, not to mention that you can start using it almost immediately. Furthermore, we are constantly rolling out new features and improving our engine.

CONCLUSION

Between customer reviews, social media, and other unstructured data, there is just too much data to be ignored, or at least analyzed by humans.

As company growth ultimately depends on making sales, which depends on attaining new customers and retaining old ones, being able to extract valuable information from unstructured data has become a necessity.

This ultimately leads to the question "should I build or buy a text mining engine?"

The answer, though, really depends on your goal.

If your goal is to learn as much as you can about natural language processing, or your goal is to research new natural language processing techniques, then there really isn't any substitute for building something yourself.

However, if you goal is simply to get the best, quickest results from your text so that you can worry about other business problems, then you're going to be best served by using a pre-existing cloud service or buying off-the-shelf software.

LEXALYTICS
Read Between The Lines

320 Congress St
Boston, MA 02210

General Inquiries
1-800-377-8036

Sales
sales @lexalytics.com
1-800-377-8036 x1

International
1-617-249-1049