5

Word-Vectors and Search Engines

So far in this book we have discussed symmetric and antisymmetric relationships between particular words in a graph or a hierarchy, described one way to learn symmetric relationships from text, and shown how to use ideas such as similarity measures and transitivity to find 'nearest neighbours' of a particular word. But ideally we should be able to measure the similarity or distance between *any* pair of words or concepts. To some extent, this is possible in graphs and taxonomies by finding the lengths of paths between concepts, but there are problems with this. First of all, finding shortest paths is often computationally expensive and may take a long time. Secondly, we might not have a reliable taxonomy, and as we've seen already, that there is a short path between two words in a graph doesn't necessarily mean that they're very similar, because the links in this short path may have arisen from very different contexts. Thirdly, the meanings of words we encounter in documents and corpora may be very different from those given by a general taxonomy such as WordNet — for example, WordNet 2.0 only gives the *fruit* and *tree* meanings for the word *apple*, which is a stark contrast with the top 10 pages returned by Google when doing an internet search with the query *apple*, which are all about Apple Computers.

Another limitation of our methods so far is that we have focussed our attention purely on individual concepts, mainly single words. Ideally, we should be able to find the similarity between two arbitrary *collections* of words, and quickly. For this, we need some process for *semantic composition* — working out how to represent the meaning of a sentence or document based on the meaning of the words it contains.

This all sounds like a pretty tall order, but (to some extent) it's actually been possible for years and you will probably have encountered such a system many times — it's precisely what a search engine does.

The traditional ways in which a search engine accomplishes these tasks are almost entirely mathematical rather than linguistic, and are based upon numbers rather than meanings. Based on its distribution in a corpus, the meaning of each word is represented by a characteristic list of numbers, and the numbers representing a whole document are then given simply by averaging the numbers for the words in the document! Bizarre but effective.

This chapter is all about the 'lists of numbers' used to represent words and documents, and these lists are called *vectors*. The discussion of vectors will finally bring us to talk in detail about different *dimensions* and what they mean to a mathematician. This in itself might be of interest to many readers, and quite different from what you supposed.

5.1 What are Vectors

A *vector* is a very useful way of keeping track of several different pieces of information, all of which relate to the same concept or object. For example, suppose I have a drawer at home where I keep my leftover currency from travelling to different countries, and have a small pile of 6 US dollars ($6), 20 UK pounds (£20), 15 Euros from Germany (€15) and 100 Japanese yen (¥100). We could write this information down in a table

$	£	€	¥
6	20	15	100

or if we keep careful track of the 'convention' ($, £, €, ¥) we could write this as the *row vector*

$$Dm = (6, 20, 15, 100),$$

where Dm stands for "Dominic's money". Now, suppose that Maryl has a similar drawer, which contains

$	£	€	¥
50	16	20	0

This information can be encoded in the row vector

$$Mm = (50, 16, 20, 0).$$

If we marry our fortunes together, what will our combined currency drawer contain? The answer is simple — just add together the numbers in the matching positions and you get the combined vector

$$Dm + Mm = (56, 36, 35, 100).$$

Suppose that we decided to put this money into savings and it grew by 20% (which corresponds to multiplying by the number 1.2). Then we'd have

$$1.2(Dm + Mm) = (67.2, 43.2, 42, 120).$$

Not impressed? Maybe it doesn't seem like rocket science to write down two lists of numbers, keep track of which numbers refer to which currencies, and then add and multiply these numbers. But it turns out that the ability to break a situation down into individual numbers, to do separate calculations with those numbers, and then to combine the answers to represent a new situation, can be extremely powerful, because it allows us to break down a potentially complicated process into a number of extremely simple ones. And by the way, you just dealt with points in a *four-dimensional* space without even blinking.

5.2 Journeys in the plane

Another situation that can be described using vectors is one we've already encountered — namely, the 2-dimensional plane can be thought of as a vector space. In Figure 29, the arrows a and b represent two 'journeys' in a plane. The combined journey from going along the a arrow and then the b arrow is called the *sum* of the journeys a and b, and so is (naturally enough) written as $a + b$.

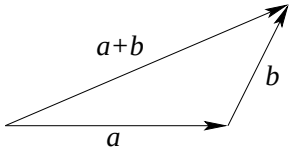


FIGURE 29 The journey $a + b$

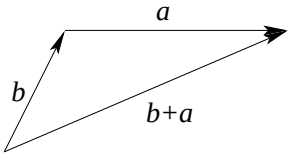
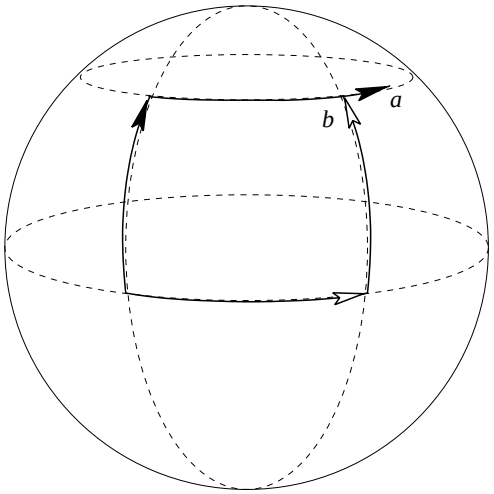


FIGURE 30 The journey $b + a$

On the other hand, we could have gone along the journey b first, followed by the journey a, and this combined journey would be called $b + a$. One fundamental property of the flat plane is that these two journeys have the same destination: $a + b$



If you travel north, then east you'll reach point *a*.

If you travel the same distances east, then north you'll reach point *b*.

FIGURE 31 When combining two journeys on a sphere, you can end up at different places depending on which journey you make first.

and $b + a$ are two ways of writing the same journey. This isn't always true — it depends on the space in which a and b are journeys. For example, imagine your space is a sphere like the earth and you start on the equator. If you go 1000 miles north and then 1000 miles east, you will end up further east that if you go 1000 miles east, *then* 1000 miles north, because the parallel (line of latitiude) 1000 miles north is a smaller circle than the parallel of the equator, as shown in Figure 31.[49] Because a sphere is curved, it turns out that the order in which you make different journeys matter — the convenient identity $a+b = b+a$ ceases to be true.

In fact, the amount to which different journeys can land you in different destinations can be used to define and measure the very concept of the *curvature* of a mathematical space. Vector spaces are special precisely because they *aren't* curved — they are 'flat' or 'linear',

[49] At the extreme, if you go as far as the north pole, the 'line of latitude' collapses from a circle to a single point — the north and south poles do not have a well defined longitude, since *all* lines of longitude intersect at the poles. This 'singularity' is responsible for the old brainteaser

> Suppose you walk ten miles south, ten miles east, ten miles north and find yourself at the point you started from. Where are you?

the answer being that you must be at the north pole.

and because of this the study of vectors is called *linear algebra*. As a result of this linearity, vector spaces obey Euclid's fifth axiom of geometry, which states that parallel lines never meet.

The process of putting two journeys, or vectors, together into a single journey is called *vector addition*. You should convince yourself that this is a wise generalisation of the real number addition operation you learnt as a child. In this context, real numbers behave as '1-dimensional vectors'. Draw yourself a mental picture of how our $a + b$ and $b + a$ journeys appear along a single line, and I hope you'll see what I mean. Another way of describing vector addition is called the 'parallelogram rule'. If you can see why by mentally combining Figures 29 and 30, you're well on the way to understanding what's going on.

The symbol '$+$' which means "add together these two vectors (or numbers)" is called an *operator*. An operator is different from a relationship such as similarity '$\sim$' or hyponymy '$\sqsubseteq$', because whereas $a \sim b$ is just an assertion that the relationship holds, the operation $a + b$ has a result or outcome. Familiar examples of relationships between numbers are "equals ($=$)" and "less than ($<$)": familiar examples of operators are multiplication, addition and subtraction. The similarity

The Parallel Axiom

Euclid's fifth *Axiom of Geometry* states

5. That, if a straight line falling on two straight lines makes the interior angles on the same side less than two right angles, the two straight lines, if produced indefinitely, meet on that side on which are the angles less than the two right angles.

This is equivalent to saying that parallel lines never meet. Because it is so much more cumbersome than the first four axioms, mathematicians tried to prove it as a consequence of these, rather than assuming it as an axiom in its own right, for over two thousand years.

In the 19th century, mathematicians such as Gauss, Bolyai, Labachevsky and Riemann finally realised that the behaviour of parallel lines depended on the nature of the space you were working in: for example, all the lines of longitude on a globe are parallel at the equator and meet at the poles. As often happens when we let go of rules that didn't need to be there, letting go of the parallel axiom led to a whole new field of *non-Euclidean geometry*, which paved the way for the Theory of Relativity.

and distance measures of Chapter 4 are all operators because they produce a number.

Just as a relationship $a \asymp b$ where the order of a and b is irrelevant is called symmetric (Definition 2), an *operator* where the order of the inputs doesn't matter also has a special name.

Definition 11 A mathematical operator '$\star$' is said to be *commutative*[50] if $a \star b = b \star a$ for all possible a and b.

Simple examples of commututive operations are addition and multiplication of real numbers: we know that $a+b = b+a$ and $ab = ba$. Subtraction, on the other hand, is certainly *not* commutative, because $a - b \neq b - a$. In fact, since in general

$$b - a = -(a - b),$$

the result of swapping a and b round is to give us the *opposite* answer, and the for this reason the subtraction operator is *anticommutative*.

The operation of adding the 'currency vectors' in section 5.1 is also commutative — it doesn't matter whether you add my fortune to Maryl's, or the other way round, you get the same answer. Vector addition must always be commutative, and this is one of the reasons why vector addition is a good generalisation of real number addition. In fact, the amalgamation of our 'currency vectors' is commutative precisely because it is a combination of four separate addition operations with numbers, and each of these separate sums is commutative.

So much for adding vectors. The other operation we must be able to perform is *scalar multiplication* — stretching or shrinking of vectors. Any of our 'journey vectors' can be scaled up or down to give you a journey in the same direction but of different length. Similarly, we could scale our currency vector by a factor of 20% (multiplying by 1.2), or any other factor (though in this case we might find ourselves rounding the result to two decimal places or the nearest 'cent', which

[50] The word *commutative* has nothing to do with travelling to work or having a prison sentence reduced: nor is there really any good reason why we couldn't call an operator *symmetric* instead of coining this new technical term. There is no particularly good reason why you have to learn a new four syllable word at this point, except that for those readers who are already familiar with it, it would be too confusing to change things now — because of this, the excessive jargonisation of all contemporary fields is getting out of hand and is apparently impossible to resist.

the bank normally does on our behalf when calculating interest). Just as addition must satisfy a few properties such as being commutative, scalar multiplication must also be well-behaved in certain ways. For example, if Maryl and I invested our money separately, each got a return of 20%, and then married our new fortunes together, we should get exactly the same amount as if we married them together first and then increased the total by 20%. In equations, this would be

$$1.2Dm + 1.2Mm = 1.2(Dm + Mm),$$

which you can easily check is true in this case. This 'scaling operation' can be traced back to Euclid's second *Axiom of Geometry*, which states that it is always possible

2. To extend a finite straight line continuously in a straight line.

A set of points equipped with addition and scalar multiplication operations must satisfy the eight axioms of Figure 32 to be a *vector space*. These axioms were given by Peano in 1888, though this was really a culmination of different strands of mathematical progress throughout the 19th century and before. In the next section we will explain how this culmination came to be, and this process will hopefully make some difficult ideas very clear. This will leave us in a sound position to use the techniques of vector spaces to describe words and their meanings.

5.3 Coordinates, bases and dimensions

What do the currency vectors of Section 5.1 and the journey vectors of Section 5.2 have in common? We could try to go through and check that the eight axioms of Figure 32 hold for both systems and say that this *is* what they have in common, but this is really a topic for a linear algebra homework assignment, not a book on meaning. What we want to know is how they came to be regarded as similar systems — after all, this similarity was recognised long before the axioms were written down.

The main breakthrough in thinking of journeys in the plane as lists of numbers is made by giving each journey a set of *coordinates*, which break the journey down into different components in different directions. You will almost certainly have come across these in high school, and we have used them already in this book.

Formal Definition of a Vector Space

A (real) Vector Space is a set V equipped with two mappings, called addition (which maps $V \times V$ to V) and scalar multiplication (which maps $\mathbb{R} \times V$ to V).

Addition must obey the following axioms:
- A1. Addition is associative, so that for all $a, b, c \in V$, $(a + b) + c = a + (b + c)$.
- A2. Addition is commutative, so that for all $a, b \in V$, $a + b = b + a$.
- A3. There is an additive identity element $0 \in V$ (called the "zero vector") such that for all $a \in V$, $a + 0 = 0 + a$.
- A4. For each element $a \in V$, there exists an element $-a \in V$ with $a + (-a) = 0$.

Scalar Multiplication must obey the following axioms:
- M1. Scalar multiplication is associative, so that for all $\lambda, \mu \in \mathbb{R}$ and for all $a \in V$, $(\lambda\mu)a = \lambda(\mu a)$.
- M2. $1a = a$ for all $a \in V$.
- M3. Scalar multiplication is distributive over addition in V, so that for all $\lambda \in \mathbb{R}$ and for all $a, b \in V$, $\lambda(a + b) = \lambda a + \lambda b$.
- M4. Scalar multiplication is commutative over addition in $\mathbb{R}$, so that for all $\lambda, \mu \in \mathbb{R}$ and for all $a \in V$, $(\lambda + \mu)a = \lambda a + \mu a$.

FIGURE 32 A vector space must satisfy these eight axioms (Jänich, 1994, p 17)

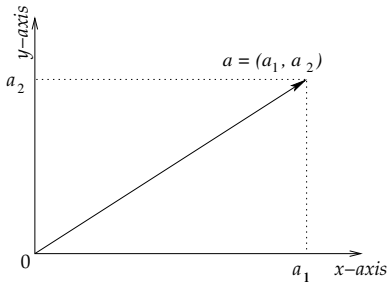


FIGURE 33 The coordinates or components of a vector

The basic picture is in Figure 33, where the vector a is decomposed into 2 parts by *projecting* it onto two fixed vectors or *axes*. If the projection hits the x-axis at distance of a_1 units from the origin, this number is said to be the *x-coordinate* of the point a, and similarly, if the projection intersects the y-axis at a_2, then this is said to be the *y-coordinate* of a. The point a is then said to have coordinates (a_1, a_2), though it is only by convention that first coordinate refers to the horizontal component and the second to the vertical component, the 'right' and 'up' directions regarded as being 'positive'.[51]

One thing that may cause confusion: we've talked about giving coordinates to 'points' in the plane and 'journeys' in the plane — but a journey requires a pair of points (a beginning and an end), not a

[51] That this is only conventional is apparent in computer graphics, where the 'down' direction is normally regarded as positive and the position of a point is determined by its coordinates measured from the top left hand corner of the screen.

single point, so how can they both be represented by the same sorts of coordinates? The answer is that the point a is identified with the journey *from the origin* or 'zero point' to the point a, so we can think of both "the point a" and "the journey from 0 to a" as vectors in the plane.

Giving these numbers or coordinates to a journey vector enables us to carry out the processes of vector addition and scalar multiplication without having to draw the pictures. For example, we can work out the additions in Figures 29 and 30 very easily: if the vector a has coordinates (a_1, a_2) and the vector b has coordinates (b_1, b_2) then their sum $a + b$ will have the coordinates $(a_1 + b_1, a_2 + b_2)$. The method of using coordinates in this fashion was a gradual mathematical development: Appolonius of Perga (*ca.* 260-190BC) and the French bishop Nicole Oresme (*ca.* 1320-1382) both described points in figures using distances from a pair of fixed locations, ideas which contributed to the *Analytic Geometry* of Descartes (1637). The description given by Descartes' of choosing a particular line as a 'unit', and ascribing numerical lengths to other lines according to their ratio with this unit, is one of the first clear-cut definitions of 'measurement' in the sense of Section 1.3.

Cartesian coordinates

The use of a pair of numbers such as (x, y) to represent points in the plane is justly associated with Rene Descartes (1596-1650), though he was not the first to invent it. Descartes realised that many of the relationships between geometric figures such as lines and curves could be represented by numbers in this way — for example, the cosine wave on page 107 is the locus of all points whose Cartesian coordinates (x, y) are related by the equation $y = \cos(x)$.

In this way, many old geometric problems, such as finding the point where two curves intersected, could be solved using the techniques of algebra which, thanks mainly to Islamic mathematicians, had made enormous progress since the Greeks. In algebraic terms, the use of Cartesian coordinates amounts to representing the plane as a set of pairs of real numbers in the product set $\mathbb{R} \times \mathbb{R}$, which is why the plane is often referred to by the symbol $\mathbb{R}^2$ (normally pronounced "R2", like the cute droid from *Star Wars*). To this day, product sets of this sort are called 'Cartesian', though ascribing such a general algebraic notion to the works of Descartes is an anachronism.

This explains how we can use numbers to represent the pictures of Figures 29 and 30. We can also go the other way — suppose we wanted to represent the first two coordinates of the currency vectors in Section 5.1 pictorially. (These are the amounts of dollars and pounds, $Dm = (6, 20)$ and $Mm = (50, 16)$.) We could plot these on a grid and add them together using the parallelogram rule, and finally read off the coordinates of the resulting sum, $Dm + Mm = (56, 36)$, as in Figure 34. This may seem like a long way round, though as we'll see in Chapter 6, representing vectors as points on a plane like this, rather than as a pair of numbers, can give a much more intuitive representation of which groups of vectors are close to one another. Just as points in the plane are often given x and y coordianates, Figure 34 shows that we can just as well talk about "$ and £ coordinates". Ideally, we would have represented the "€ and ¥ coordinates" as well, though it would be difficult to represent three of these coordinates at once on a flat page, and pretty much impossible to represent all four of them in a way that made any sense.

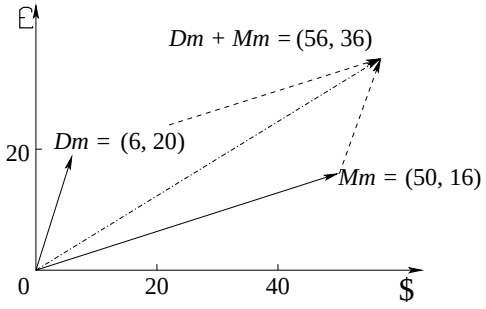


FIGURE 34 Adding together currency vectors

In order to represent a vector as a list of numbers, it is clearly necessary to keep a 'key' telling us which quantity each number refers to. For the currency vectors, we needed to remember that the code (a_1, a_2, a_3, a_4) was shorthand for "$\$a_1$, $£a_2$, $€a_3$ and $¥a_4$". When working in the plane we need to choose coordinate axes, and while the convention is to use 'eastward' and 'northward' pointing axes, there is no *a priori* reason for making this particular choice.[52] This

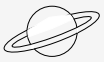
[52]We could use 'south' and 'north-northwest' as our coordinate axes, since you can still specify any journey by saying how far south you have to go, followed by how far north-northwest you have to go — and this representation would be just as unique and unambiguous as the more familiar x and y coordinates. When developing the technique of using coordinates, Descartes in fact used many different pairs of axes, not necessarily at right angles to one another — the convention of using the same pair of perpendicular axes for most problems came later.

key telling us which coordinate is which is called a *basis* for the vector space. For example, the basis for our currency vectors is the set $\{\$1, £1, €1, ¥1\}$.

Now, each coordinate refers back to one of these 'basis elements', so the number of coordinates needed to represent any point is always the same as the number of items in the basis. For example, the vector $Dm = (6, 20, 15, 100)$) has 4 coordinates, one to refer to each of the 4 currencies listed in the basis. This number is an important characteristic property of any vector space, and it's a word you will have come across many times.

Definition 12 The *dimension* of a vector space is the number of coordinates needed to uniquely specify a given point.

The flat plane has two dimensions, since each point needs 2 coordinates to represent it: normally these are the x and y coordinates. The currency vectors of Section 5.1 have *four* dimensions, because each vector is made up of 4 coordinates (or numbers) of Dollars, Pounds, Euros and Yen.



Time: absolute or relative?

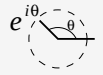
The difference between a vector itself and the coordinates used to write the vector down may be subtle, but it is a vital conceptual difference. For example, (according the the Special Theory of Relativity) the time measured between different events is not a property of the events themselves, but depends on how you choose your coordinates.
Because the speed of light c is *constant*, and speed is a relationship between position and time, if you change the way you measure position you must also change the way you measure time. The relationships between the 'time-coordinates' of different events is therefore different for different observers. This interplay between space and time has strange consequences — for example, two events which are similtaneous for one observer may happen at *different* times for another. If this sounds confusing but you really want to 'get it', see `www-csli.stanford.edu/~dwiddows/train.html`.

Hopefully this will break a few misunderstandings, if they haven't been broken already. We've probably all heard and wondered about the question

Is time the fourth dimension?

Well, clearly the answer for our currency vectors is "no" because in this space, "Japanese Yen" is the fourth dimension! This goes back to the difference between studying physical *space* as opposed to studying mathematical *spaces*. In mathematics, we use however many dimensions we need to represent the information we're working with — if a sitation is complicated enough to require more than than 2, or 3, or 4 coordinates to properly represent it, then we'll use a 'space' with as many coordinates as are necessary. However, if you *are* seeking to model the physical universe, then four coordinates can be a very good choice — three to measure different directions in space, and one to measure the time that elapses between two events (this is exactly what we did when defining the *Minkowski distance* on page 110). So in this sense, the answer is "yes", insofar as there are some very good models of the physical universe where time is a 'fourth dimension'.

The main trouble that most people have when trying to 'think' in more than three dimensions is that they try to visualise all these dimensions at once. We already accepted this limitation in Figure 34: when I was explaining what currency vectors and journey vectors have in common, I only drew a picture of two of the 'currency coordinates' (dollars and pounds) since we could easily represent these on a flat piece of paper. Do not hold yourself back from understanding what dimensions are by trying to visualise more than three dimensions at once: it's a good way of gettting frustrated because it's just not within our physical experience. There are some rare people who



Coordinates and vectors

Since there are many different ways to choose a basis, and thus to assign coordinates to each point in a vector space, we must check that the mathematics is the same whatever choice we make. For example, if the dimension of a vector space is the number of elements in any basis, we had better make sure that this number is the same for *every* possible basis of a given vector space (Jänich, 1994, p. 46) — otherwise 'dimension' would not be a property of the space itself, but only of one particular coordinate system.

Also, the sum of two vectors $a + b$ can be calculated by adding their respective coordinates, but must be the same vector for every possible coordinate system. These important properties can all be proved from the axioms in Figure 32.

say that they can 'picture' four or more dimensions. I can't, I never have been able to, and I found spaces of higher dimensions much easier to live and work with once I accepted that several dimensions could be perfectly valid and consistent, even if I couldn't force them to somehow fit into my physical, visual experience. There's a cheesy moral for life there somewhere, I'm sure, but maybe it's high time we got back to talking about language.

5.4 Search Engines and Vectors

Early search engines relied on simple 'term matching'. If a document contained the keywords in a user's query, it was marked 'relevant' and returned to the user: if not, it was marked 'irrelevant' and left behind. But by the 1970's, as document collections grew bigger, the flaws in such an approach became more and more damaging. One main problem that people encountered is that with large document collections (which would be considered small by today's standards), returning *every* document that matched the keywords would still leave the user with a huge mass of documents to wade through to try and work out if they were relevant. For example, suppose that you wanted to find out about *geometry*. A search of the internet which returned *every* matching document would end up directing you to nearly every course list in every mathematics department in every university, since nearly all of these course lists will probably contain the word *geometry* somewhere. But these pages won't actually be *about* geometry, so they won't really be *relevant* to the user's query. To take extreme examples, a dictionary will probably contain most common words that aren't proper names, but the dictionary isn't relevant to every query containing these words. A telephone directory will contain a huge number of proper names, but it still won't be relevant to many queries containing one of these names. Long documents containing huge numbers of words could be declared relevant to almost any query — but these are often precisely the documents whose information is most diluted. It became very clear that a better measure of relevance was needed than could be achieved by simply dividing the document collection into documents that did and didn't contain the keywords.

One way to do this (and I should stress that there are several)

Document 1 (BNC)

The first **Bass** VI I remember seeing was being used on television by Jack Bruce, during a mimed performance of Strange Brew by **Cream**. Recollections indeed: an extremely fashionable **Cream**, with serious sideburns all round, grossly extended collars on satin shirts, Clapton's Hendrix perm, flared trousers and Les Paul. Being nobbut a sprog, I remember wondering why bassist Bruce was playing a **guitar**. Only he wasn't, of course; he was playing a Fender **Bass** VI. He also became arguably the most famous exponent of the instrument, along with Eric Haydock of The Hollies.

Document 2 (NYT, 1996)

ORLEANS, Mass. — When weekend **fishermen** looking to unwind come to Rock Harbor, they take out their rods and reels and angle for striped **bass** one by one. When **commercial fishermen** like Mike Abdow go out on their boats to earn a living, they catch **bass** the same way. By law, they must reel them in one at time, without nets, traps or long-line gear. But right now, the waters are rough between people who fish **bass** for a living and those who angle for pleasure. In a feud that has divided people along the Massachusetts coast, big-**money** sporting interests are trying to stop small-time **commercial fishermen** from pulling in any more striped **bass**.

Document 3 (NYT, 1995)

Kuala Lumpur, Oct. 26 (Bloomberg) – **bank** Negara will change the way it calculates the cost of lending **money** to **commercial** banks and financial institutions, the central **bank** said in a release. In a statement, the **bank** said that as of Nov. 1, the base lending rate, at which **commercial** banks can borrow from the central **bank**, will become more responsive to movements in **money**-market rates. Several weeks ago, the **bank** said it would change the way it calculates the base rate, said Desmond Ch'ng, a banking analyst OCBC Securities (Malacca) Sdn. Bhd.

FIGURE 35 Three documents about different topics

is to represent words using vectors. The *coordinates* of these vectors are numbers which measure the extent to which a particular word is important to a particular document. To begin with, these coodinates are obtained simply by counting the number of times each word is used in each document.

As a small-scale example of this technique, consider the three documents in Figure 35, in which a handful of different words have been highlighted (*bank*, *bass*, *commercial*, *cream*, *guitar*, *fishermen* and *money*). The number of instances (tokens) of each of these words in each of the three documents is recorded in Table 4.

Such a table, which records the number of times each word occured in each of a collection of documents, is called a *term-document matrix*.[53] Obviously, the snapshot we've taken in Table 4 is a tiny fragment of the term-document matrix used by any real search engine — we've

[53] A *matrix* (plural *matrices*) in this context is simply a table of numbers. There are well-defined algebraic rules for how pairs of matrices can be added and multiplied together, provided that the number of rows and columns in the two matrices is compatible. Matrices grew out of an an 1858 memoir on transformations written by the British mathematician Arthur Cayley (1821-1895), partly as a generalisation of Hamilton's *quaternions* (Boyer and Merzbach, 1991, Ch 26).

	Document 1	Document 2	Document 3
bank	0	0	4
bass	2	4	0
commercial	0	2	2
cream	2	0	0
guitar	1	0	0
fishermen	0	3	0
money	0	1	2

TABLE 4 We count the number of times each of these words is used in each of our documents

only taken three documents and only considered a few of the words they contain. However, it's still possible to get some idea of how the table works, and hopefully the reader will then be able to extrapolate and imagine the huge table we'd produce from a document collection of several million words (or nowadays, several *billion* webpages).

Table 4 can be used as a simple 'inverted file index', like a traditional concordance.[54] That is, if you're interested in *fishermen*, the table will tell you to look in Document 2, if you're interested in a *bank*, the table will tell you to look in Document 3, and so on. But the real advantage of measuring the *number* of times each word occurs in a document comes when you have words occuring in many documents, and you want to know which document is the most relevant. For example, if you want to find out about *money*, Table 4 will tell you that it's mentioned in both Document 2 and Document 3, but since it's mentioned *twice as much* in Document 3, you should start by reading this document, because the term *money* is more concentrated or 'denser' in this document. This would be good advice — while Document 2 mentions tensions between two groups of people which have monetary consequences, Document 3 is directly about finance.

There are many problems and issues with this idea that we haven't even begun to address, some of which are listed below:

- Many possible meanings of the different terms are absent from this

[54]The term 'inverted file index' or 'inverted file format' is used because words are described by a list of documents, whereas it's more usual to think of documents being by a list of words (Salton and McGill, 1983, Ch 2). A concordance, often of the Bible, is a big book where a Biblical scholar can look up a word such as *light* and find the chapter and verse of every place where the word *light* appears, hoping to trace the way a theological concept is used and developed through the centuries.

fragment — *bank* can also refer to a *river bank*, and most of the time *cream* is more likely to mean a *dairy product* than a *1960's rock band*.

- The term *bass* does have more than one meaning in our 3 documents — it means a kind of *musical instrument* in Document 1 and a kind of *fish* in Document 2. If a user is only interested in one of these meanings (and in this case, it's unlikely that they'd be interested in both at the same time), how are we to enable users to search for only the documents containing this meaning of *bass*?
- The term *guitar* only appears once in Document 1 — but it's very possible that a user searching for the term *guitar* would also find articles containing the term *Les Paul* relevant, since *Les Paul* is a make of *guitar*.

Many of these questions will be addressed during the rest of this book — though in truth, none of them can honestly be said to have been completely solved.

Now, here's the point. The rows of numbers in Table 4 can be thought of as vectors, just like the lists of numbers in the 'currency vectors' of Section 5.1. The different rows can be added together component by component, or scaled by any other real number, and it is a simple matter to check that these rows, and the operations of row addition and scalar multiplication, satisfy the axioms of Definition 32. Because of this, some of the theory and techniques of vectors, which are very well developed and well understood parts of mathematics, can be used to calculate and reason with words.

In fact, because they only have 3 coordinates each, it's almost possible to imagine these particular word vectors as points in a 3 dimensional space, as I've tried to depict in Figure 36. However, as we go through this section you'll realise that this diagram is only a visual aid — all of the mathematics we use to work out which words are similar to which other words can be done entirely by working with the coordinates in Table 4.

5.5 Similarity and distance functions for vectors

In this section we describe the most prominent ways to measure similarity and distances between vectors with any number of dimensions, and show how they can be applied to our word vectors.

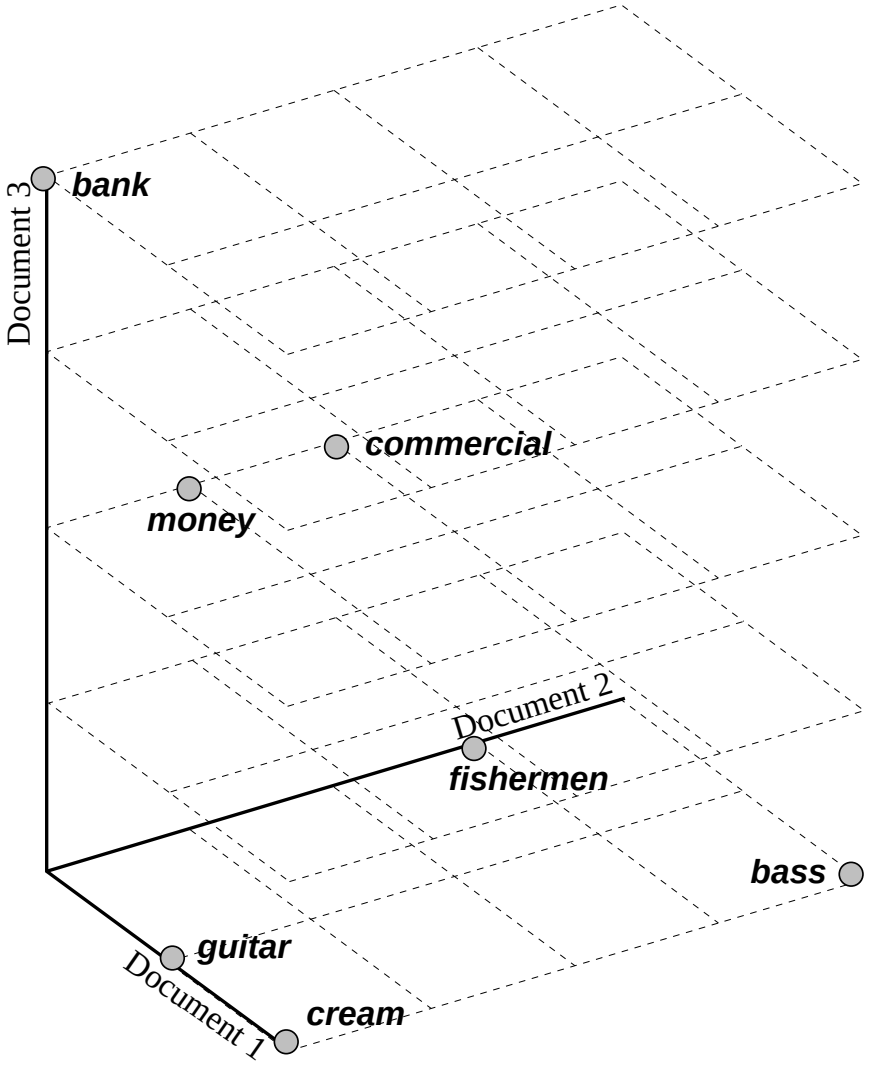


FIGURE 36 Imagining the word vectors in Table 4 as points in the three dimensional space $\mathbb{R}^3$

This should provide you with some robust mathematical tools and terminology (which may seem challenging, but which are easy to use in practice and very easy to program into a computer) and a taste of the possible linguistic applications.

Lets start with a few examples from the term-document matrix in Table 4, which gives us word vectors such as

$$\textit{bank} = (0, 0, 4), \quad \textit{bass} = (2, 4, 0) \quad \text{and} \quad \textit{money} = (0, 1, 2).$$

In order to measure a 'similarity' between these word vectors, we could just multiply their first, second and third coordinates together and add the results, exactly as we did when introducing the cosine measure (Definition 8, page 107). For example, to obtain a score for the similarity of *bank* and *money*, we calculate

$$\mathrm{sim}(\textit{bank}, \textit{money}) = (0 \times 0) + (0 \times 1) + (4 \times 2) = 8.$$

Using the same measure, the similarity between *bass* and *money* would be

$$\mathrm{sim}(\textit{bass}, \textit{money}) = (2 \times 0) + (4 \times 1) + (0 \times 2) = 4,$$

and the similarity between *bank* and *bass* would be

$$\mathrm{sim}(\textit{bank}, \textit{bass}) = (0 \times 2) + (0 \times 4) + (4 \times 0) = 0.$$

If this doesn't 'click' straight away, do pause to work out where these numbers have come from and why they've been paired up in this fashion. The resulting 'similarity scores' are not without merit — for these three words, they show us that the two most similar words are *bank* and *money*, that *bass* and *money* are less similar though not unrelated (which in this context appears to make sense — the similarity is drawn entirely from Document 2, which does discuss the financial interests behind bass fishing), whereas *bass* and *bank* have nothing in common at all.

However, there are drawbacks with this similarity score — in particular, it gives higher similarities to more frequent words. For example, for the vectors $\textit{guitar} = (1, 0, 0)$ and $\textit{cream} = (2, 0, 0)$, the similarities for *cream* will always be exactly double those for *guitar*: but just as when we compared composers such as *mozart* and *wagner* with composers such as *hottenterre* and *giamberti* in Section 4.3, it doesn't seem right that *cream* should be twice as 'similar' as *guitar* to every single word, just because it occurs twice as often. (Of course,

the distribution in our 'three document' example is hardly realistic, but it makes the general point that frequent words might get higher 'similarity scores' across the board, and this would not be such a good thing.)

Just as in Section 4.3, we can get round this problem by *normalising* or 'dividing out' our similarity scores. This will be easiest if we introduce some notation and terminology for vectors in general, which will also be frequently used in later chapters.

5.5.1 Introducing the vector space $\mathbb{R}^n$

So far we've seen that a 'journey vector' a in the plane may be given by two coordinates. Because each point is given by 2 real numbers, the plane is given the name '$\mathbb{R}^2$'. Because our word vectors in Table 4 are collected from 3 different documents, each word vector in this example has 3 coordinates. The space of all possible collections of 3 real numbers is given the name '$\mathbb{R}^3$'. This idea can easily be generalised — a vector with n real number coordinates is considered to be a 'point in $\mathbb{R}^n$'.

Definition 13 The vector space $\mathbb{R}^n$ is made up of all lists of the form $(a_1, a_2, \ldots, a_n)$ where each of these entries is a real number.

By definition, each point in $\mathbb{R}^n$ has n coordinates, so the dimension of $\mathbb{R}^n$ is the number n.

Addition of vectors in $\mathbb{R}^n$ is defined just as you would expect — the sum of two vectors $(a_1, a_2, \ldots, a_n)$ and $(b_1, b_2, \ldots, b_n)$ is

$$(a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n).$$

To multiply the vector $(a_1, a_2, \ldots, a_n)$ by a given number or 'scale factor' λ, again multiply each of the coordinates in turn, giving

$$(\lambda a_1, \lambda a_2, \ldots, \lambda a_n).$$

This may look horrible. If you're having difficulty, go back to the calculations with currency vectors in Section 5.1, which are precisely examples of the equations above. Because we're so used to the idea of currencies, it's completely natural to think of adding two collections together by adding the dollars to the dollars, then adding the pounds to the pounds, *etc.* It also makes sense that to scale a whole currency vector by a given scale factor, you have to scale each currency (or each

'coordinate') by this factor in turn. The equations we've just written down are nothing more than general versions of exactly these 'one currency at a time' operations.

Once we start dealing with more than two or three coordinates, we normally stop using different letters such as (x, y, z) for the different coordinates, for two reasons. The first is that sooner or later we'd run out of letters. The second is that it's actually *easier* to use the 'subscript notation' of Definition 13. How it will normally work is that we'll use one letter for each whole vector (such as a), and subscripts such as a_1, a_2, *etc.* for the coordinates. In this way, there are fewer letters to keep track of, and you know straight away that (for example) 'a_3' means "the 3^{rd} coordinate of the vector a". This sort of notation makes it very easy to extend the Euclidean distance and cosine similarity measure of Chapter 4 to cope with any number of coordinates.

Arrays

The concepts and the notation behind the vector space $\mathbb{R}^n$ will probably be familiar to any programmer, even though the terminology may be different. A 'vector' is just the sort of data that can be stored as an *array*, and a vector in $\mathbb{R}^n$ is an array whose elements are n 'real' numbers (in practice, these real numbers are approximations such as the floating point numbers and doubles of the C language).

If `arr` is an array variable then its j^{th} element is usually referred to by `arr[j]`, which is very similar to the 'subscript' notation of Definition 13, where a_j is the j^{th} coordinate of the vector a. The only difference is that in computing the 'first' element of an array is usually called `arr[0]` rather than `arr[1]`.

5.5.2 The scalar product of two vectors

Now that we're gradually becoming familiar with vectors in $\mathbb{R}^n$, and the way of writing the coordinates $(a_1, a_2, \ldots, a_n)$ for the vector a, we can use these techniques to derive equations for computing similarities and distances for general vectors. We've already used a form of similarity score between our word vectors, by multiplying their coordinates and adding the results. This 'product' of two vectors has a special name.

Definition 14 Let $a = (a_1, a_2, \ldots, a_n)$ and $b = (b_1, b_2, \ldots, b_n)$ be

vectors in $\mathbb{R}^n$. Their *scalar product* $a \cdot b$ is given by the formula

$$a \cdot b = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n.$$

We'll see shortly that the scalar product is closely linked to both the Euclidean distance and cosine similarity measures of Chapter 4.

5.5.3 Euclidean distance on $\mathbb{R}^n$ — extending Pythagoras' theorem to n dimensions

The two dimensional version of Pythagoras' theorem (Section 102) will already be familiar to most readers because it is widely taught in schools. This enables you to work out the length of the hypotenuse of a right-angled triangle (alternatively, the length of the diagonal of a rectangle).

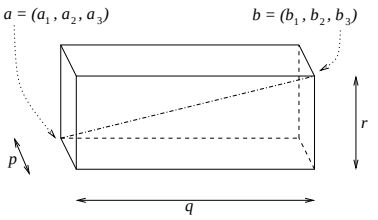


FIGURE 37 The length of the diagonal between the points a and b is the square root of $p^2 + q^2 + r^2$.

A less well-known but equally interesting fact is that exactly the same principle can be used in *three* dimensions to calculate the length of the diagonal of a solid box or 'cuboid' (which is the three dimensional version of a rectangle). For example, if the three perpendicular sides of a box have lengths p, q and r, then the diagonal of the box will have a length of $\sqrt{p^2 + q^2 + r^2}$, which is easy to prove.[55] If the corners of the box are the points a and b, which have coordinates (a_1, a_2, a_3) and (b_1, b_2, b_3) respectively, then the lengths of these sides are $p = b_1 - a_1$, $q = b_2 - a_2$ and $r = b_3 - a_3$. The length of the diagonal, which is the distance $d(a, b)$ between a and b, is therefore given by the formula

$$d(a, b) = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + (b_3 - a_3)^2}. \qquad (5.10)$$

This is essentially the same as the 2 dimensional distance formula given in Equation (4.6), though we've changed some of the notation. For example, we can use the summation sign Σ (the capital Greek

[55] The proof works by applying Pythagoras' theorem twice: first considering the diagonal of the rectangular base of the cuboid, whose length is $\sqrt{p^2 + q^2}$; and then considering the hypotenuse of the triangle made by this diagonal and the upright distance z.

letter sigma) to mean "the sum of all these numbers", in which case Equation (5.10) can be written using the shorthand

$$d(a,b) = \sqrt{\Big(\sum (b_i - a_i)^2\Big)}. \tag{5.11}$$

This means exactly the same thing as Equation (5.10), once you've 'expanded out' the terms in the '$\sum$' expression. (If we wanted to make it particularly explicit that the subscript i takes the values 1, 2 and 3 we'd write '$\sum_{i=1}^{3}$', though normally this will be unnecessary because it should be clear from the context what the range of values the subscript i takes.)

The same 'index and summation' notation we used in Equation (5.11) can be used to express the scalar product of two arbitrary vectors $a = (a_1, a_2, \ldots, a_n)$ and $b = (b_1, b_2, \ldots, b_n)$ as

$$a \cdot b = \sum a_i b_i.$$

(If unsure, you should check this against Definition 14 to convince yourself that they're saying the same thing.) As you can see, this expression is much more succinct than the first version in Definition 14, and in the long run this brevity of expression is a great benefit. (If you like, it's a bit like using acronyms such as *USA* and *UN* for special terms — it takes a while to get used to for a newcomer, but once you're used to it, the idea of writing out the full expression in words every time you need it is far too cumbersome for real life.)

Example 7 Calculating Distances

To make this a bit easier to grasp, we'll use Equation 5.11 to work out the 'distances' between some of our word vectors in Table 4, which contains the word vectors

$$\textit{bank} = (0,0,4), \quad \textit{guitar} = (1,0,0) \quad \text{and} \quad \textit{money} = (0,1,2).$$

Applying Equation 5.11 gives

$$\begin{aligned} d(\textit{bank}, \textit{money}) &= \sqrt{(0-0)^2 + (0-1)^2 + (4-2)^2} \\ &= \sqrt{0+1+4} \\ &\approx 2.24. \end{aligned}$$

In the same way, we get

$$d(\textit{bank}, \textit{guitar}) = \sqrt{1+16} \approx 4.12.$$

Comparing these two results, we find that *guitar* is 'further' from *bank* than *money* is — a reasonable enough deduction.

However, there are problems with this technique — 'large' vectors tend to be more distant from most other vectors than 'small' vectors. For example, yet more calculations using Equation 5.11 tell us that

$$d(\textit{guitar}, \textit{bass}) \approx 4.12 \quad \text{whereas} \quad d(\textit{guitar}, \textit{fishermen}) \approx 3.16.$$

You can confirm this by looking again at Figure 36, where *guitar* is slightly closer to *fishermen* than to *bass*. But this isn't really because *guitar* and *fishermen* have more in common than *guitar* and *bass* — a brief glance at Table 4 shows that *guitar* and *bass* have some coordinates in common whereas *guitar* and *fishermen* have none. Instead, the smaller distance between *guitar* and *fishermen* is simply because they're both nearer to the origin point. You can almost think of this as word vectors getting thrown out from a sort of 'lexical supernova' — those words which get thrown the furthest end up in deep space, while those which don't get thrown very far end up clustered around the origin, and closer to one another.

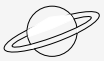
Using subscripts for coodinates and the summation sign Σ takes a bit of mental gymnastics but it's well worthwhile because you can write down much more general formulas and equations without any extra effort. In particular, it Equation (5.11) can be used to define a distance function for a vector space of *any* dimension, without being changed at all. So far in this section, we've presumed that we're working in $\mathbb{R}^3$, where each vector a has 3 coordinates (a_1, a_2, a_3). But Equation 5.11 works in just the same way for longer vectors $(a_1, a_2, \ldots, a_n)$, where the number of coordinates n can be as big as you like. In this way, Pythagoras' theorem can be adapted to give a distance measure on the vector space $\mathbb{R}^n$ for any dimension n. This measure is called the *Euclidean distance* measure on $\mathbb{R}^n$, and since it obeys the three metric space axioms (Definition 7), it is sometimes called the *Euclidean metric*.

This is another example of the fairly 'relaxed' mentality you need to accept the ways vectors and dimensions are used. The analogy by which Pythagoras' theorem is extended from 2 dimensions to 3 dimensions should be pretty clear: the next step of applying the same numerical formulas to more than 3 dimensions isn't complicated.

However, the conceptual step of trying to imagine the Euclidean distance function in $\mathbb{R}^4$ as somehow measuring the length of the diagonal of a 4 dimensional box is challenging, to put it mildly, if not downright crazy. But you don't need to do this to understand word vectors, just as we don't really need the diagram in Figure 36 in order to understand the significance of the word vectors Table 4. The main point behind Descartes' contribution to this sort of mathematics is that *we don't need diagrams for everything* — we can work out the distances algebraically, straight from the numerical coordinates. This is the reason why these techniques can be so easily adapted to spaces with more than 3 dimensions which we can't visualise so easily.

5.5.4 Norms and unit vectors

We now know how to calculate the Euclidean distance and the scalar product between two vectors a and b. However, we've also seen that neither of these measures is an ideal way to work out similarities and distances between word vectors: with the Euclidean distance, frequently occuring words with large word vectors end up *too far* from most other words, and with the scalar product, the same frequently occuring words end up *too similar* to most other words. What we need is a way of factoring out these unfair advantages and disadvantages, just as we wanted to even out our graph similarity scores in Section 4.3.1.



Choosing a norm

Choosing the 'right' norm function for a given vector space will depend on your scientific purpose in building the space. For 'currency vectors', the only sensible way to find the value of a whole currency vector would be to use exchange rates. At the time of writing, £1 = \$1.66, €1 = \$1.18, and ¥1 = \$0.0092, and so the vector $Dm = (6, 20, 15, 100)$ has the total value or 'norm'

$$6 + 20 \times 1.66 + 15 \times 1.18 \\ +100 \times 0.0092 = \$57.82,$$

measured in dollars. In effect, we are using a 'weighted manhattan metric' (page 103) to evaluate the length of a currency vector. (In practice, the 'norm' of a currency vector should allow negative as well as positive values, in case you have more debts than assets — not all financial 'distances' are positive.)

To do this, we measure the 'size' or 'length' of each vector, which is its distance from the zero point or origin. Applying Equation

(5.11) to the vectors $0 = (0, 0, \ldots, 0)$ and $a = (a_1, a_2, \ldots, a_n)$, we have

$$d(0, a) = \sqrt{\sum (a_i)^2}.$$

This can also be expressed in tems of the scalar product, since $\sum (a_i)^2$ is the same as $\sum a_i a_i$ which is just $a \cdot a$. This is summed up in the following definition.

Definition 15 The *norm* or *length* of the vector a is defined to be

$$||a|| = \sqrt{(a_i)^2} = \sqrt{a \cdot a}.$$

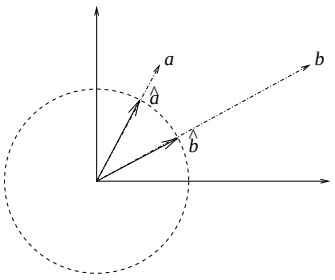


FIGURE 38 Normalised or 'unit' vectors

Just as the norm $||a||$ can be defined as the Euclidean distance $d(0, a)$, so also the Euclidean distance between two vectors a and b can be obtained by the formula $d(a, b) = ||b - a||$, which is the norm of the 'journey vector' from a to b. So in many ways, whether we choose to talk in terms of norms or distances is a matter of convenience, just as whether we think of a vector as a point or as a journey from the origin to that point is a matter of convenience.

This 'norm' is exactly the factor we need to divide by in order to remove any extra preference or penalty given to the word vectors of frequently occuring words. It's easy to check that if you divide every coordinate in a by the norm $||a||$, you are left with a vector whose norm or length is equal to 1. This vector is written as $\frac{a}{||a||}$ or sometimes $\hat{a}$, and is called the *unit vector* of a. In other words, $\hat{a}$ is the vector in the same direction as a whose length is just one unit. One simple way to think of the unit vector $\hat{a}$ is as the projection of the vector a onto a sphere or circle of radius 1, since all the points on this sphere or circle have a distance of one unit from the origin (Figure 38). Since all unit vectors have the same length, the only thing that distinguishes these vectors is their 'directions'.

We can now replace all the word vectors in Table 4 with their unit vectors, giving the 'normalised' version in Table 5. What we've done is found the norm of each row and divided each entry in the table by

	Document 1	Document 2	Document 3
bank	0	0	1
bass	0.447	0.894	0
commercial	0	0.707	0.707
cream	1	0	0
guitar	1	0	0
fishermen	0	1	0
money	0	0.447	0.894

TABLE 5 The vectors form Table 4 after they have been normalised

this number. To check that these normalised vectors *are* unit vectors, simply go along each row in the table, square each individual number and add together the results — you should get the answer 1 (or at least, as nearly as our approximation to 3 decimal place will allow).

5.5.5 Cosine similarity in $\mathbb{R}^n$

The scalar product of two *normalised* vectors corresponds exactly with their *cosine similarity*, as defined in Section 4.2. We are thus in a very useful practical situation: we can work out similarities between words simply by working out the cosine similarities between their vectors in Table 5 — by multiplying together the corresponding coordinates and adding the results. For example, we now have

$$\cos(\textit{guitar}, \textit{cream}) = 1 + 0 + 0 = 1,$$

$$\cos(\textit{guitar}, \textit{bass}) = 0.477 + 0 + 0 = 0.477,$$

and

$$\cos(\textit{guitar}, \textit{fishermen}) = 0 + 0 + 0 = 0.$$

Now we have that *guitar* and *cream* are the most similar pair (which with the meaning of *cream* in Document 1 is what we should have), that *guitar* and *bass* have a certain amount in common (in fact, they have 'one of the meanings of *bass* in common but not both), and *guitar* and *fishermen* are completely unrelated. The skewing of such results because of *bass* being a 'longer' vector, and recurrent problems of this nature, are gone for good.

The cosine similarity of any pair of vectors (not just unit vectors) can easily be obtained by taking their scalar product first and then dividing by their norms (rather than normalising all vectors first and then computing cosine similarities between these normalised vectors).

This is probably the most usual way of defining cosine similarity, and you will often see equations like

$$\cos(a, b) = \frac{a \cdot b}{||a||\ ||b||} \tag{5.12}$$

used in the literature to define the cosine similarity between two vectors a and b. As we said in Section 4.2, this number can also be thought of as the cosine of the angle between the vectors a and b, which is a good way to think of this measure of similarity which makes it intuitively clear that we are only interested in the directions, not the lengths of the vectors (since the angle between two lines doesn't depend in any way on the lengths of those lines).

To see that this way of measuring the similarity between normalised vectors is a reasonable approximation to measuring the similarity between words, have a good look at Table 6. This finally gives the similarity between *each* pair of word vectors. Remember that this is all calculated from the fragment of language contained in Documents 1, 2 and 3 in Figure 35, and in this tiny sample many of the words only occur with a fraction of their possible meanings. Given this, Table 6 does a remarkably good job of modelling which words are similar and dissimilar to one another.

Such a table of similarities between each pair of objects in a collection can be called a *data matrix* or an *adjacency matrix*. Note that the main top-left to bottom-right diagonal is made entirely of 1's — this is because each word has a similarity of 1 with itself,

Cosine similarity and Euclidean distance

The cosine similarity and Euclidean distance between two unit vectors are closely related: if a and b are unit vectors then it follows that

$$\begin{aligned}(d(a,b))^2 &= ||b-a|| \\ &= (b-a)\cdot(b-a) \\ &= b\cdot b + a\cdot a - 2a\cdot b \\ &= 2 - 2a\cdot b.\end{aligned}$$

It follows that the relative ranking of points as being 'more or less similar' according to cosine similarity or 'closer or further away' according to Euclidean distance will be the same for unit vectors.

However, because the left hand side in this equation is squared, cosine similarity is 'less transitive' than Euclidean distance. For example, the vectors $a = (1, 0)$, $b = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$ and $c = (0, 1)$ all have length 1, and although $\cos(a, b) = \cos(b, c) = \frac{\sqrt{2}}{2} \approx 0.707$ which is quite high, $\cos(a, c)$ is still 0.

	bank	bass	commercial	cream	guitar	fishermen	money
bank	1	0	0.707	0	0	0	0.894
bass	0	1	0.632	0.447	0.447	0.894	0.400
commercial	0.707	0.632	1	0	0	0.707	0.915
cream	0	0.447	0	1	1	0	0
guitar	0	0.447	0	1	1	0	0
fishermen	0	0.894	0.707	0	0	1	0.447
money	0.894	0.400	0.915	0	0	0.447	1

TABLE 6 The cosine similarities between each pair of words in our model

because the length of each normalised vector is equal to 1. If we reflect the table about this diagonal (thereby interchanging the rows and the columns), notice that we get the *same* table. A table (or 'matrix') with this property is called *symmetric*, because it is symmetrical in the traditional "it looks the same if you reflect it in a mirror" fashion, and because symmetric matrices can be used to represent *relations* which are symmetric in the sense of Definition 2. This will become clearer when we describe ways to transfer information between graphs and vectors spaces in Section 6.7.

5.6 Document vectors and information retrieval

This chapter was meant to tell you how a search engine works, and so far it's been entirely about words and these things called vectors: whereas one thing that all readers will probably know is that search engines are about finding documents. In case you're feeling tricked into reading a whole chapter on mathematics under false pretences, we will finish this chapter by describing how to create *document vectors* so that we finally have a little system which can take a query made out of any combination of the seven terms in Table 6 and rank the three documents in Figure 35 according to their relevance to such a query.

The trick is very simple: we represent the documents as being vectors in *the same space* as the words, and then we can compute similarities between words and documents just as we computed similarities between pairs of words. In a sense, we've already been doing this — our word vectors have had three coordinates, one for their occurence in each of the documents, and in this sense the three documents have been used as a *basis* for our space of word vectors. One way to make this clearer is to reason that if we had 4 documents in our collection, we would need 4 columns in the term-document matrix and each word vector would have 4 coordinates, and so on for n documents: so the number of documents clearly determines the dimension of the space of word vectors. Another way to see this pictorially is to consult the diagram in Figure 36, where the 3 documents are clearly being used as 3 'coordinate axes'. Representing each document as a unit vector in the direction of its coordinate axis, we have the *document vectors*

$$\textit{Doc1} = (1,0,0), \quad \textit{Doc2} = (0,1,0) \quad \text{and} \quad \textit{Doc3} = (0,0,1).$$

It's now a simple matter to work out the cosine similarity between a given query expression and each document in turn. For example, for the query *bass* with vector $(0.477, 0.894, 0)$, we get

$$\cos(\textit{bass}, \textit{Doc1}) = 0.477, \qquad \cos(\textit{bass}, \textit{Doc2}) = 0.894,$$
$$\text{and} \qquad \cos(\textit{bass}, \textit{Doc3}) = 0.$$

If the user was returned the winning document (*Doc2*) and realised that this isn't the sort of *bass* they were looking for, they could add

Programming cosine similarity in higher dimensions

If we use arrays to hold our vectors and a `for-next` loop to work out the cosine, cosine similarity in n dimensions can be implemented using exactly the same code as for 2 dimensions. If your vectors are called `vec1` and `vec2` and their cosine similarity is stored by the variable `cos`, the code would read something like

```
cos=0;
for( i=0; i<dim; i++){
 cos=cos+vec1[i]*vec2[i];
}
```

Norms and Euclidean distances of vectors are just as easy to adapt to n dimensions: in some ways, computers find this easier than people because computers don't care whether or not they can 'visualise' the vectors they're using.

the term *guitar* to give the query *bass guitar*. Now the benefits of using vectors really begin to pay off — we can simply add together the vectors for *bass* and *guitar* to give a new vector,

$$\textit{bass} + \textit{guitar} = (0.477, 0.894, 0) + (1, 0, 0) = (1.477, 0.894, 0).$$

Comparing each document with this new query vector, *Doc1* is the winner with a cosine score of 1.477. In a very simple way, we have combine the 'meanings' of two words to give a 'meaning' for a combination of words, for the first time in this book. This very important process is called *semantic composition*, which we are modelling here using vector addition.

You can carry on playing the game of working out query vectors, comparing them with the three document vectors, and see if the ranking you get coincides with how relevant the documents are to your query.

That is basically how to build an information retrieval system — that and nearly half a century of research into some really fundamental questions about how to assess the importance of different terms and documents (Google's *PageRank* is one example of an answer to such a question), how this vector model compares with other conceptual models, how different terms might depend on one another, how to cope with the engineering task of managing bigger and bigger document collections and term-document matrices. The 'wider reading' section at the end of this chapter will barely be able to scratch the surface of these topics, though will try to give some good leads which you can follow for yourself.

This may seem like a lot of mathematics in order to end up multiplying a few numbers here are there and coming up with a ranking of our three documents. In many ways, this is one of the vector model's great strengths — the mathematics in this chapter may have had some strange names and symbols and far too many dimensions, but at the end of the day this is nothing more than a shorthand for adding and multiplying lots of different numbers. These numbers can be extracted straight from a document collection without any need for deep linguistic analysis, which is one of the main reasons that information retrieval has proved to be such a widespread and successful branch of natural language processing: your system really

doesn't need to know much about *language* at all. (If you just stop for a moment to think about how much more trouble we'd have had in building a simple system that could read out a spoken version of three documents or translate them into a different language getting both the grammar and the meaning correct, I think you'll agree that building an information retrieval system was pretty easy.)

Another vital point is that the toy system we've described in this chapter is *scalable*. This is where the mental gymnastics of working in many dimensions pays off. It may have seemed like unnecessarily hard work to develop all this n dimensional mathematics when most of the time we were just working in 2, 3 and 4 dimensions. But right at the end of the chapter we realised that we were using a 'dimension' for each separate document. Now, if we'd confined ourselves to a comfortable mathematical system that could just cope with 2 or 3 dimensions, we'd have no hope of coping with a bigger document collection. But instead, we put in the hard work to use a mathematical language that can be adapted to *any* number of dimensions: so the same system can be used for a document collection of any size.[56]

I'd like to finish this chapter with two observations which take us back to the concepts of Chapter 1. We said in Section 1.4.4 that the 'measurements' we made of the extent to which different words occured in different relationships or contexts would be *discrete*, but that many of the models we would build from this information would rely on *continuous* mathematical techniques. The vector spaces of this chapter are precisely the sort of continuous model we were referring to. We started by counting discrete whole numbers (for example, in Table 4), but by the time we had normalised these vectors in Table 5, we needed all sorts of numbers in between 0 and 1, and in principle the only limit on the 'granularity' of these numbers is how many

[56] In the presentation in this chapter, I have departed slightly from the approach in most textbooks, which uses the words or *terms* as coordinate axes from which to represent both document and query vectors. This is mainly because it is a better introduction to the way we will use word vectors in the rest of the book, where we are much more interested in computing word-word similarities than (say) document-document similarities. Also, it was much easier to try to draw Figure 36 as 7 word points in 3 'document dimensions' than as 3 document points in 7 'word dimensions'! However, the concepts are very much the same: I honestly don't know whether the difference would have a noticeable practical effect when building a search engine from scratch.

decimal places it's worthwhile keeping for each of them. When we're just comparing the relevance of three different documents, it doesn't matter so much: when we're dealing with a document collection the size of the World Wide Web, we may need more and more 'in between numbers' to keep track of ever finer distinctions of importance and relevance. The old binary distinction into relevant and non-relevant documents could not support a user who needed to know which out of thousands of 'relevant' documents were the *most* relevant, and this challenge has naturally led researchers to study continuous methods such as vector spaces, fuzzy set theory and probability.

In a sense, our models have come all the way from Chapter 2, where we studied relationships between words almost without using any numbers at all, to the work in this chapter where words have been represented using *only* numbers, obtained by measuring how many times a word occurs in different documents. Geometric comparisons such as longer or shorter, closer or further away, have been made purely by doing calculations with these coordinates. This has all been made possible by the 'Cartesian revolution' in mathematics which enabled geometric problems to be described and solved using relationships between collections of numbers.

Wider Reading

The introduction to vectors given here is necessarily brief and very informal. Those who wish to go deeper or more thoroughly into any of the concepts introduced here should consult a proper book on linear algebra. Chapters 2 and 3 of (Jänich, 1994) are particularly appropriate and readable: I also sometimes recommend (Vallejo, 1993).

The rest of this book will describe many important operations with vectors, and the ways they have been used for analysing linguistic information: a lengthy list of 'wider reading' on this topic will gradually emerge. However, readers who are particularly interested in the particular uses of vectors for information retrieval should consult the classic text of (Salton and McGill, 1983, Ch 4), and the summary contained in (Jurafsky and Martin, 2000, §17.3) which gives a good pictorial overview.

The principal mathematical models used for information retrieval systems are the *vector*, *probabilistic* and *fuzzy set* models, all of which

are discussed in Baeza-Yates and Ribiero-Neto (1999). Fuzzy sets for information retrieval are discussed at length in (Miyamoto, 1990), which starts with a good introduction to what fuzzy sets are, and also considers fuzzy-set models for some of the traditional thesaural relationships such as 'Broader Term' which we met in Chapter 2. One of the most interesting probabilistic models is the 'inference network' used in the INQUERY system (Turtle and Croft, 1989, Callan et al., 1992): such networks use probability theory to find important paths in directed graphs of the sort we studied in Chapter 3, in this case the path that leads between a query and relevant documents.

A unifying theme behind these three models is that normalised vectors, probability distributions and fuzzy sets are all mathematical techniques which replace binary values (0 or 1) which signify 'not belonging' or 'belonging' with continuous values (0 or 1 or anything in between) which signify 'partially belonging' or 'probably belonging'. There are some practical differences between the models (for probability distributions, the sum of all the individual probabilities must be equal to 1, whereas as we've seen for normalised vectors, the sum of the *squares* of the individual coordinates must be equal to one, at least with the Euclidean norm), but it seems at least possible that these are really different *implementations* of one underlying *model*.

Other important topics in information retrieval include weighting strategies, user models and interfaces, linguistic operations such as stemming words (such as *fishermen*) to their lexical roots (such as *fisherman* or even *fish*) and of course engineering. Rather than try to give piecemeal and unsatisfactory pointers to each of these topics, I recommend consulting the indexes of general books such as (Salton and McGill, 1983), (Kowalski, 1997) and (Baeza-Yates and Ribiero-Neto, 1999), and also the excellent range of papers collected in (Sparck Jones and Willett, 1997).

Those interested in the mathematical development of vectors should certainly delve into the works of Descartes: there is a particularly beautiful (and as usual, unbeatably priced) version available from Dover publications which contains a facsimile of Descartes' original French manuscript next to the English translation. An excerpt from the vital first book is contained in (Smith, 1959, p

397-403), and a good secondary historical account can be found in (Boyer and Merzbach, 1991, Ch 17).

The technique of using vectors to represent points in space grew from the work of Sir William Rowan Hamilton (1805-1865) on quaternions, in which he first describes the addition of a 'fourth dimension' to a mathematical system Hamilton (1847). At the same time, during the 1840's a little known German high school teacher called Hermann Grassman (1809-1877) was developing a system of spaces called *Ausdehnungslehre*, and this work contains our modern notion of vectors in *any* number of dimensions (Smith, 1959, p 684-696). Since his purpose was to represent a general concept with any number of dimensions, our word vectors owe much more to this foundation. Grassmann was a specialist in Sanskrit literature and also made contributions to Indo-European linguistics, raising the question of whether Grassmann's inspirations and our modern treatment of dimensions are rooted in much more ancient sources.

In cognitive science, the use of coordinates and dimensions to model mental the cornerstone of the *Conceptual Spaces* of Gärdenfors (2000). In particular, the first chapter of this book contains an excellent introduction to the notion of dimensions, and the way different stimuli such as taste and colour can be represented as points in such a conceptual space. For example, colours seem to have a clear '3 dimensionality' about them, whether those dimensions are 'red, green and blue' or 'hue, brightness and chromaticity'.

No bibliography about dimensions would be complete without mentioning Edwin Abbott's novel *Flatland*, originally published in 1884. This slim book (under 100 pages) is the darling of every enthusiast who's ever read it and is available for $1.50 (from Dover, of course). In Abbott's story (also a barbed satire against the hierarchical rigidity of Victorian society), the world of a humble square is rocked by the intersection of a solid sphere with his two-dimensional existence. At first resistant, the square becomes convinced that a *third dimension* is possible, and through trying to explain this to the other polygons he becomes a prophet, a heretic and finally a prisoner, clinging forlornly to the hope that his teachings will one day inspire a new generation "who shall refuse to be confined to limited Dimensionality".

References

Agirre, E. and G. Rigau. 1996. Word sense disambiguation using conceptual density. In *Proceedings of COLING'96*, pages 16–22. Copenhagen, Denmark.

Aitchison, Jean. 2002. *Words in the Mind: An Introduction to the Mental Lexicon*. Blackwell, 3rd edn.

Alfonseca, Enrique and Suresh Manandhar. 2001. Improving an ontology refinement method with hyponymy patterns. In *Third International Conference on Language Resources and Evaluation*, pages 235–239. Las Palmas, Spain.

Baeza-Yates, Ricardo and Berthier Ribiero-Neto. 1999. *Modern Information Retrieval*. Addison Wesley / ACM Press.

Bean, Carol A. and Rebecca Green, eds. 2001. *Relationships in the Organization of Knowledge*. Kluwer.

Birkhoff, Garrett. 1967. *Lattice theory*. American Mathematical Society, 3rd edn. First edition 1940.

Birkhoff, Garrett and John von Neumann. 1936. The logic of quantum mechanics. *Annals of Mathematics* 37:823–843.

Bodenreider, Olivier and Rebecca Green. 2001. Relationships among knowledge structures: Vocabulary integration within a subject domain. In Bean and Green (2001), chap. 6, pages 81–98.

Bohm, David. 1951. *Quantum Theory*. Prentice-Hall. Republished by Dover, 1989.

Bohm, David. 1980. *Wholeness and the Implicate Order*. Routledge Classics, republished 2002. Routledge.

Bollobás, Béla. 1998. *Modern Graph Theory*. No. 184 in Graduate Texts in Mathematics. Springer-Verlag.

Boyer, Carl B. and Uta C. Merzbach. 1991. *A History of Mathematics*. Wiley, 2nd edn.

Brin, Sergey and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems* 30(1–7):107–117.

Budanitsky, A. and G. Hirst. 2001. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources*. NAACL, Pittsburgh, PA.

Buitelaar, P. 1998. *CoreLex: Systematic Polysemy and Underspecification*. Ph.D. thesis, Computer Science Department, Brandeis University.

Bush, R. R. and F. Mosteller. 1951. A model for stimulus generalization and discrimination. *Psychological Review* 48:413–423.

Callan, James P., W. Bruce Croft, and Stephen M. Harding. 1992. The INQUERY retrieval system. In *Proceedings of DEXA-92, 3rd International Conference on Database and Expert Systems Applications*, pages 78–83.

Cantor, Georg. 1895,1897. *Contributions to the Founding of the Theory of Transfinite Numbers*. Dover, 1952nd edn.

Caraballo, Sharon. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. In *37th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pages 120–126.

Carroll, Lewis. 1865. *Alice's Adventures in Wonderland*. Penguin Popular Classics (1994).

Carroll, Lewis. 1872. *Through the Looking Glass*. Penguin Popular Classics (1994).

Chang, B. S. W., K. Jönsson, M. A. Kazmi, M. J. Donoghue, and T. P Sakmar. 2002. Recreating a functional ancestral archosaur visual pigment. *Mol Biol Evol* 19:1483–1489.

Chapman, Graham, John Cleese, Terry Gilliam, Eric Idle, Terry Jones, and Michael Palin. 2002. *Monty Python's Life of Brian (Of Nazareth)*. Methuen.

Chartrand, Gary. 1985. *Introductory Graph Theory*. Dover.

Cohen, David. 1989. *An introduction to Hilbert Spaces and Quantum Logic*. Spriger-Verlag.

Collins, A. M. and M. R. Quillian. 1969. Retrieval time from semantic memeory. *Journal of verbal learning and verbal behaviour* 8:240–247.

Cruse, Alan. 2000. *Meaning in Language - An Introduction to Semantics and Pragmatics*. Oxford Textbooks in Linguistics. Oxford University Press.

Cruse, D. A. 2002. Hyponymy and its varieties. In R. Green, C. Bean, and S. H. Myaeng, eds., *The Semantics of Relationships: An interdisciplinary perspective*, chap. 1. Kluwer.

Dagan, Ido, Lilian Lee, and Fernando C. N. Pereira. 1999. Similarity-based models of word cooccurrence probabilities. *Machine Learning* 34(1–3):43–69.

Darwin, Charles. 1859. *The Origin of Species*. Penguin Classics (ed. J. W. Burrows (1982)).

Davey, B. A. and H. A. Priestley. 1990. *Lattices and Order*. Cambridge University Press.

Deerwester, Scott, Susan Dumais, George Furnas, Thomas Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41(6):391–407.

Dolan, W., L. Vanderwende, and S.D. Richardson. 1993. Automatically deriving structured knowledge bases from online dictionaries. In *PACLING 93, Pacific Assoc. for Computational Linguistics*, pages 5–14.

Dumais, S., T. Landauer, and M. Littman. 1996. Automatic cross-linguistic information retrieval using latent semantic indexing. In *SIGIR '96 – Workshop on Cross-Linguistic Information Retrieval*, pages 16–23.

Dunlop, Mark. 1997. The effect of accessing nonmatching documents on relevance feedback. *ACM Transactions on Information Systems* 15(2):137–153.

Eckmann, Jean-Pierre and Elisha Moses. 2002. Curvature of co-links uncovers hidden thematic layers in the world-wide web. *Proceedings of the National Academy of Science* 99:5825–5829.

El-Hoshy, L.M. 2001. Relationships in library of congress subject headings. In Bean and Green (2001), chap. 6, pages 81–98.

Evens, Martha Walton, ed. 1988. *Relational Model of the Lexicon: Representing Knowledge in Semantic Networks*. Cambridge University Press.

Faatz, Andreas, Stefan Hoermann, Cornelia Seeberg, and Ralf Steinmetz. 2001. Conceptual enrichment of ontologies by means of a generic and configurable approach. In *Proceedings of the ESSLLI 2001 Workshop on Semantic Knowledge Acquisition and Categorisation*.

Fellbaum, Christiane. 1998a. A semantic network of english verbs. In Fellbaum (1998b), chap. 3, pages 69–104.

Fellbaum, Christiane, ed. 1998b. *WordNet: An Electronic Lexical Database*. Cambridge MA: MIT Press.

Foskett, D. J. 1997. Thesaurus. In K. S. Jones and P. Willett, eds., *Readings in Information Retrieval*, pages 111–134. Morgan Kaufmann.

Frege, Gottlob. 1884. *The Foundations of Arithmetic (1884)*. Blackwell, 1974th edn.

Fulton, William and Joe Harris. 1991. *Representation theory - a first course*. No. 129 in Graduate Texts in Mathematics. Springer-Verlag.

Gamut, L.T.F. 1991. *Logic, Language, and Meaning*. University of Chicago Press.

Ganter, Bernhard and Rudolf Wille. 1999. *Formal Concept Analysis: Mathematical Foundations*. Springer.

Gärdenfors, Peter. 2000. *Conceptual Spaces: The Geometry of Thought*. Bradford Books MIT Press.

Goldstein, Jade and Jaime Carbonell. 1998. The use of mmr, diversity-based reranking in document reranking and summarization. In *Twente Workshop on Language Technology*, vol. 14, pages 153–164.

Goldstein, Jade, Vibhu O. Mittal, Jaime G. Carbonell, and James P. Callan. 2000. Creating and evaluating multi-document sentence extract summaries. In *Ninth International Conference on Information Knowledge Management (CIKM)*, pages 165–172.

Goscinny, Rene and Albert Uderzo. 1974. *Asterix et les Goths*. Editeur Dargaud.

Grefenstette, Gregory. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer.

Guthrie, L, J Pustejovsky, Y Wilks, and B Slator. 1996. The role of lexicons in natural language processing. *Communications of the ACM* 39(1):63–72.

Hahn, Udo and Klemens Schnattinger. 1998. Towards text knowledge engineering. In *AAAI/IAAI*, pages 524–531.

Hamilton, A. G. 1982. *Numbers, Sets and Axioms*. Cambridge University Press.

Hamilton, Sir William Rowan. 1847. On quaternions. *Proc. Royal Irish Acad.* 3:1–16.

Hausdorff, Felix. 1914. *Grundzüge der Mengenlehre*. von Veit (Germany), 1914. Republished as *Set Theory*, 2nd ed. Chelsea (New York), 1962.

Hearst, Marti and Hinrich Schütze. 1993. Customizing a lexicon to better suit a computational task. In *ACL SIGLEX Workshop*. Columbus, Ohio.

Hearst, Marti A. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING*. Nantes, France.

Hearst, Marti A. 1998. Wordnet: An electronic lexical database. In Fellbaum (1998b), chap. 5, Automated discovery of WordNet relations, pages 131–152.

Heath, Thomas L., ed. 1956. *The Thirteen Books of Euclid's Elements*, vol. I-III. Dover.

Hersh, William, Chris Buckley, T. J. Leone, and David Hickam. 1994. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual ACM SIGIR Conference*, pages 192–201.

Herskovits, Annette. 1986. *Language and Spatial Cognition*. Cambridge University Press.

Hirst, G. and D. St-Onge. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. In Fellbaum (1998b), chap. 13, pages 305–332.

Horn, Laurence. 2001. *A Natural history of Negation*. Stanford: CSLI publications.

Huddleston, Rodney and Geoffrey K. Pullum. 2002. *The Cambridge Grammar of the English Language*. Cambridge, UK: Cambridge University Press.

Ide, Nancy and Jean Véronis. 1998. Introduction to the special issue on word sense disambiguation: The state of the art. *Computational Linguistics* 24(1):1–40.

Jammer, Max. 1993. *Concepts of Space: The History of Theories of Space in Physics*. Dover, 3rd edn.

Jänich, Klaus. 1994. *Linear algebra*. Undergraduate Texts in Mathematics. Springer-Verlag.

Jiang, J. and D. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *In Proceedings of International Conference on Research in Computational Linguistics*. Taiwan.

Jones, H. F. 1990. *Groups, Representations and Physics*. Adam Hilger/IOP Publishing.

Jurafsky, Daniel and James H. Martin. 2000. *Speech and Language Processing*. New Jersey: Prentice Hall.

Kilgarriff, Adam. 1993. Dictionary word sense distinctions: An enquiry into their nature. *Computers and the Humanities* 26(1–2):365–387.

Kilgarriff, A. and J. Rosenzweig. 2000. Framework and results for english senseval. *Computers and the Humanities* 34(1-2):15–48.

Kleinberg, J., S.R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. 1999. The web as a graph: Measurements, models and methods. In *Invited survey at the International Conference on Combinatorics and Computing*.

Kleinberg, J. and S. Lawrence. 2001. The structure of the web. *Science* 294:1849–1850.

Kowalski, Gerald. 1997. *Information retrieval systems: theory and implementation*. Norwell, MA: Kluwer academic publishers.

Leacock, Claudia and Martin Chodorow. 1998. Wordnet: An electronic lexical database. In Fellbaum (1998b), chap. 11, pages 265–283.

Leech, G., R. Garside, and M. Bryant. 1994. Claws4: The tagging of the british national corpus. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING 94)*, pages 622–628. Kyoto, Japan.

Lehmann, Fritz, ed. 1992. *Semantic Networks in Artificial Intelligence*. Pergamon Press.

Lenat, Douglas B. and R. V. Guha. 1990. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley.

Lesk, M. E. 1986. Automated sense disambiguation using machine-readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the SIGDOC conference*. ACM.

Li, Hang and Naoki Abe. 1998. Generalizing case frames using a thesaurus and the mdl principle. *Computational Linguistics* 24(2):217–244.

Lin, Dekang. 1998a. Automatic retrieval and clustering of similar words. In *COLING-ACL*. Montreal.

Lin, Dekang. 1998b. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*. Madison, WI.

Manning, Christopher D. and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: The MIT Press.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics* 19(2):313–30.

McKeon, Richard, ed. 1941. *The Basic Works of Aristotle*. Random House.

Melamed, I. Dan. 2000. Pettern recognition for mapping bitext correspondence. In J. Véronis, ed., *Parallel Text Processing*, pages 25–48. Kluwer.

Mervis, C. and E. Rosch. 1981. Categorization of natural objects. *Annual Review of Psychology* 32:89–115.

Miller, George A. 1998a. Nouns in wordnet. In Fellbaum (1998b), chap. 1, pages 23–46.

Miller, George A. and William G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes* 6(1):1–28.

Miller, Katherine J. 1998b. Modifiers in wordnet. In Fellbaum (1998b), chap. 2, pages 47–68.

Minnen, Guido, John Carroll, and Darren Pearce. 2001. Applied morphological processing of english. *Natural Language Engineering* 7(3):207–223.

Mitchell, Tom. 1997. *Machine Learning*. McGraw-Hill.

Miyamoto, Sadaaki. 1990. *Fuzzy sets in information retrieval and cluster analysis*. Kluwer.

Moore, Robert C. 2001. Towards a simple and accurate statistical approach to learning translational relationships among words. In *Proceedings of the workshop on data-driven machine translation*. 39th annual meeting of the Associate for Computational Linguistics, Toulouse.

Partee, Barbara H., Alice ter Meulen, and Robert E. Wall. 1993. *Mathematical Methods in Linguistics*. Kluwer.

Patwardhan, S., S. Banerjee, and T. Pedersen. 2002. Using semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*. Mexico City.

Pereira, F., N. Tishby, and L. Lee. 1993. Distributional clustering of english words. In *30th Annual Meeting of the Association for Computational Linguistics*, pages 183–190. Columbus, Ohio.

Peters, Carol, Martin Braschler, Julio Gonzalo, and Michael Kluck, eds. 2002. *Evaluation of Cross-Language Information Retrieval Systems, Second Workshop of the Cross-Language Evaluation Forum, CLEF 2001, Darmstadt, Germany, September 3-4, 2001, Revised Papers*, vol. 2406 of *Lecture Notes in Computer Science*. Springer. ISBN 3-540-44042-9.

Proctor, P., ed. 1978. *The Longman Dictionary of Contemporary English (LDOCE)*. London: Longman.

Pustejovsky, James. 1995. *The Generative Lexicon*. Cambridge, MA: MIT press.

Putnam, Hilary. 1976. The logic of quantum maechanics. In *Mathematics, Matter and Method*, pages 174–197. Cambridge University Press.

Quillian, M. R. 1968. Semantic memory. In M. Minsky, ed., *Semantic Information Processing*, chap. 4, pages 227–270. MIT press.

Resnik, Philip. 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of artificial intelligence research* 11:93–130.

Riloff, Ellen and Rosie Jones. 1999. Learning dictionaries for infomation extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 472–479. AAAI.

Riloff, Ellen and Jessica Shepherd. 1997. A corpus-based approach for building semantic lexicons. In C. Cardie and R. Weischedel, eds., *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 117–124. Somerset, New Jersey: Association for Computational Linguistics.

Roark, Brian and Eugene Charniak. 1998. Noun-phrase co-occurence statistics for semi-automatic semantic lexicon construction. In *COLING-ACL*, pages 1110–1116.

Rooth, Mats, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via em-based clustering. In *In Proceeding of the 37th Annual Meeting of the Association for Computational Linguistics*. College Park, Maryland.

Rosch, Eleanor. 1975. Cognitive representations of semantic categories. *Journal of Experimental Psychology: General* 104:192–233.

Roweis, Sam and Lawrence Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science* 290(5500):2323–2326.

Salton, Gerard and Chris Buckley. 1990. Improving retrieval performance by relevance feedback. *Journal of the American society for information science* 41(4):288–297.

Salton, Gerard, Edward A. Fox, and Harry Wu. 1983. Extended boolean information retrieval. *Communications of the ACM* 26(11):1022–1036.

Salton, Gerard and Michael McGill. 1983. *Introduction to modern information retrieval*. New York, NY: McGraw-Hill.

Schütze, Hinrich. 1997. *Ambiguity resolution in language learning*. Stanford CA: CSLI Publications.

Schütze, Hinrich. 1998. Automatic word sense discrimination. *Computational Linguistics* 24(1):97–124.

Schutze, H. and J. Pedersen. 1995. Information retrieval based on word senses. In *Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175. Las Vegas.

Shannon, Claude. 1948. A mathematical theory of communication. *Bell system technical journal* 27:379–423, 623–656.

Smith, David Eugene, ed. 1959. *A Source Book in Mathematics*. Dover. First published in 1929, McGraw Hill.

Sowa, John F. 2000. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Pacific Grove, CA: Brooks Cole Publishing.

Sparck Jones, Karen. 1986. *Synonymy and Semantic Classification*. Edinburgh University Press. (Originally Cambridge PhD thesis, 1964).

Sparck Jones, K. and P. Willett, eds. 1997. *Readings in Information Retrieval*. Morgan Kaufmann.

Stevenson, Mark. 2003. *Word Sense Disambiguation: The Case for Combining Knowledge Sources*. Stanford, CA: CSLI Publications.

Stevenson, Mark and Yorick Wilks. 2001. The interaction of knowledge sources for word sense disambiguation. *Computational Linguistics* 27(3):321–349.

Trudeau, Richard J. 1994. *Introduction to Graph Theory*. Dover.

Turtle, Howard and W. Bruce Croft. 1989. Inference networks for document retrieval. In *Proceedings of the 13th Annual ACM SIGIR Conference*, pages 1–24.

Tversky, A. 1977. Features of similarity. *Psychological Review* 84(4):327–352.

Tversky, A. 1982. Similarity, separability and the triangle inequality. *Psychological Review* 89(2):123–154.

Vallejo, Robert J. 1993. *Linear algebra: an introduction to abstract mathematics*. Undergraduate Texts in Mathematics. Springer-Verlag.

Van der Waerden, B. L. 1985. *A History of Algebra*. Springer-Verlag.

Vossen, Piek. 1998. Introduction to eurowordnet. *Computers and the Humanities* 32(2-3):73–89.

Špela Vintar, Paul Buitelaar, Bärbel Ripplinger, Bogdan Sacaleanu, Diana Raileanu, and Detlef Prescher. 2002. An efficient and flexible format for linguistic and semantic annotation. In *Third International Language Resources and Evaluation Conference*. Las Palmas, Spain.

Wall, Larry, Tom Christiansen, and Jon Orwant. 2000. *Prgramming Perl*. O'Reilly, 3rd edn.

Widdows, Dominic. 2003a. A mathematical model for context and word-meaning. In *Fourth International and Interdisciplinary Conference on Modeling and Using Context*. Stanford, California.

Widdows, Dominic. 2003b. Orthogonal negation in vector spaces for modelling word-meanings and document retrieval. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*. Sapporo, Japan.

Widdows, Dominic. 2003c. Unsupervised methods for developing taxonomies by combining syntactic and statistical information. In *Proceedings of Human Langauge Technology / North American Chapter of the Association for Computational Linguistics*. Edmonton, Canada.

Widdows, Dominic, Scott Cederberg, and Beate Dorow. 2002. Visualisation techniques for analysing meaning. In *Fifth International Conference on Text, Speech and Dialogue*, Lecture Notes in Artificial Intelligence 2448, pages 107–115. Brno, Czech Republic: Springer.

Widdows, Dominic and Beate Dorow. 2002. A graph model for unsupervised lexical acquisition. In *19th International Conference on Computational Linguistics*, pages 1093–1099. Taipei, Taiwan.

Wierzbicka, Anna. 1996. *Semantics: Primes and Universals*. Oxford University Press.

Wilce, Alexander. 2003. Quantum logic and probability theory. In E. N. Zalta, ed., *The Stanford Encyclopedia of Philosophy (Spring 2003 Edition)*. Stanford University.

Yang, Yiming, Jaime Carbonell, Ralf Brown, and Robert Frederking. 1998. Translingual information retrieval: Learning from bilingual corpora. *Artificial Intelligence Journal special issue: Best of IJCAI-97* pages 323–345.

Yarowsky, David. 1993. One sense per collocation. In *ARPA Human Language Technology Workshop*, pages 266–271. Princeton, NJ.

Yarowsky, David. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196.

Zalta, Edward N. 2003. Frege's logic, theorem, and foundations for arithmetic. In E. N. Zalta, ed., *The Stanford Encyclopedia of Philosophy (Spring 2003 Edition)*.